Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Chancellerie fédérale ChF

Transformation numérique et gouvernance de l'informatique TNI

14.09.2026

Em002-2 Instructions pour la publication d’OSS

Version 2.0-31789e3

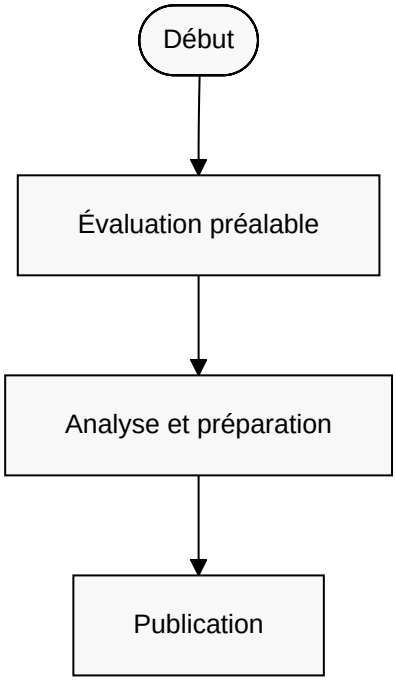
Table des matières

Clause de non-responsabilité : Le présent document est un projet en cours d'élaboration et fait partie des outils destinés à soutenir l'administration fédérale dans la publication de code open source. [1][2][3][4] Pour de plus amples informations, voir le README principal.

1. Résumé à l'intention de la direction

Les présentes instructions décrivent la procédure de publication du code source des logiciels que les autorités fédérales développent ou font développer pour accomplir leurs tâches, conformément à l'art. 9 LMETA.[5] Des exceptions sont prévues lorsque des droits de tiers ou des motifs relevant de la sécurité excluent ou restreignent cette publication. Publier signifie que quiconque peut utiliser, développer et diffuser le logiciel. Aucune redevance de licence n'est perçue.

Les présentes instructions s'adressent aux personnes responsables de la publication du code source et chargées de sa mise en œuvre.



La matière est divisée en trois parties principales :

- La première partie (chapitre 3) traite des **évaluations préalables** et des exceptions prévues par la LMETA.
- La deuxième partie (chapitre 4), **Analyse et préparation**, examine le code source et le prépare selon les besoins. Le choix de la licence y est également effectué.

1. Recommandation pour l'informatique de l'administration fédérale conformément à [P035] *chapitre 4.6*
2. Pour les définitions des classifications INTERNE et CONFIDENTIEL, voir l'*ordonnance du 8 novembre 2023 sur la sécurité de l'information au sein de la Confédération et de l'armée (OSI ; RS 128.1)*
3. Voir note de bas de page 1
4. Domaines de planification conformément à la *stratégie informatique de la Confédération 2020-2023 du 3 avril 2020 (SB000)*
5. Loi fédérale *[EMOTA2023]*

- La troisième partie (chapitre 5), **Publication et annonce**, décrit la publication proprement dite ainsi que d'autres mesures telles que la communication et la constitution d'une communauté appropriée.

Trois listes de contrôle accompagnent et documentent le processus.

Des compléments techniques figurent en annexe.

2. Introduction

Lors de la publication de logiciels open source, il convient de distinguer entre la **contribution de code source et de documentation à un logiciel open source existant** et la **publication en tant que projet open source indépendant**.

Le premier cas comprend typiquement des corrections d'erreurs et des extensions fonctionnelles. Selon le type de licence et l'utilisation du logiciel, le code source doit ou peut être publié sous la licence existante du projet open source. Un accord de publication peut devoir être respecté.

Dans le second cas, le lancement d'un nouveau projet open source, la gouvernance et la licence peuvent en principe être choisies librement. Le seul élément à prendre en compte est la licence sous laquelle sont publiés les éléments logiciels éventuellement intégrés au nouveau projet. De plus amples détails sur le choix de la licence figurent dans le document Em002-3 Guide de licences OSS. Si une plus-value peut être générée pour l'administration fédérale grâce à une communauté ou si l'administration fédérale souhaite créer un écosystème, la liste de contrôle Em002-4.1 Communauté OSS doit également être remplie conformément au Em002-4 Guide des communautés OSS.

La publication de logiciels open source est plus qu'une simple mesure technique. Elle suppose aussi d'établir une culture correspondante d'ouverture et d'embarquer les personnes concernées. Une culture open source réussie est un mélange d'ouverture, d'excellence technique, de forte interaction sociale et d'engagement partagé.

Le schéma suivant décrit les instructions et les différentes configurations.

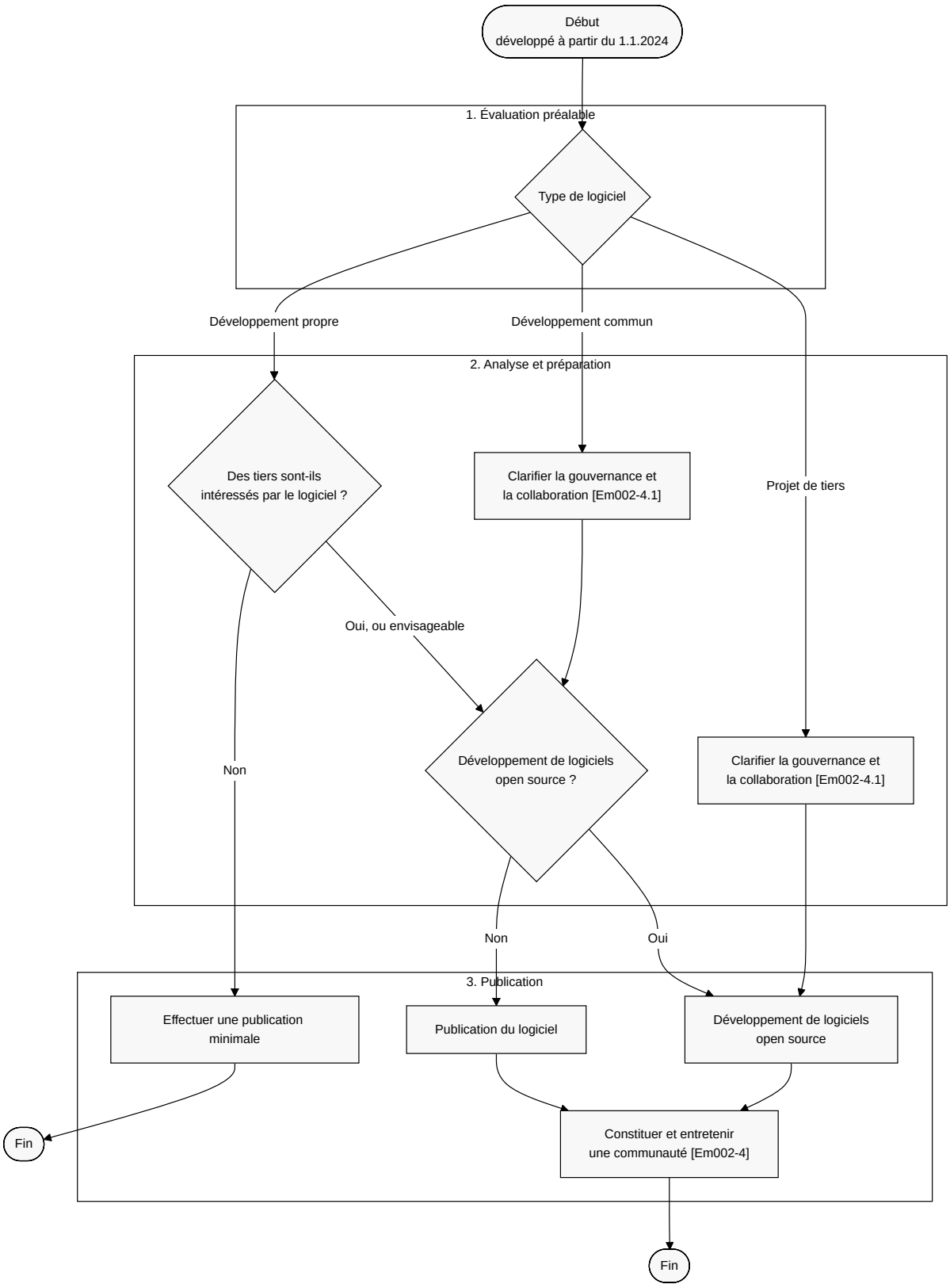


Figure 1 – Arbre de décision pour la publication de logiciels

3. Processus d'évaluation préalable

Objectif : comprendre les avantages et les conséquences possibles d'une publication du logiciel. Décider si le logiciel peut être publié et, dans l'affirmative, si la constitution d'une communauté active en vaut la peine. Remarque : la liste de contrôle Em002-2.1 Évaluation préalable OSS devrait être

remplie le plus tôt possible dans le projet, car cela peut influencer la manière dont le logiciel est développé.

3.1. Publication selon la LMETA

Conformément à l'art. 9 de la loi fédérale du 17 mars 2023 [EMOTA2023] sur l'utilisation des moyens électroniques pour l'exécution des tâches des autorités (LMETA), l'administration fédérale (concrètement l'art. 2, al. 1) doit publier le code source des logiciels qu'elle développe ou fait développer pour accomplir ses tâches. Des exceptions s'appliquent lorsque des droits de tiers ou des motifs relevant de la sécurité l'excluent ou le restreignent.

Pour les logiciels sur mesure créés sur mandat de la Confédération, les conditions générales de la Confédération [BBL-AGB] s'appliquent en principe. Selon leur chiffre 25.1, la propriété du code source et de la documentation passe au service acquéreur (la Confédération). Si le logiciel a été acquis conjointement par plusieurs organisations ou d'autres institutions, il faut vérifier qui détient les droits sur le code. La propriété peut par exemple appartenir à une association fondée à cet effet.

La simple utilisation de logiciels standard n'entre pas dans le champ de l'art. 9 LMETA et une publication ne serait de toute façon généralement pas possible, les droits sur les logiciels standard restant souvent chez le fournisseur (CG de la Confédération pour l'acquisition de logiciels standard). Les extensions sur mesure de logiciels standard se prêtent en revanche tout à fait à une publication. Elles appartiennent normalement à la Confédération. De simples adaptations de configuration se prêtent moins à une publication. Le sujet est traité dans Em002-7 Aspects stratégiques des achats et des logiciels open source ainsi que dans les documents [KKB-MB] et [BBL-WL]. En principe, les extensions relèvent toujours de l'art. 9 LMETA.

3.1.1. Le logiciel au regard de la LMETA

La norme ISO/IEC 24765 actuelle contient trois définitions du logiciel :

1. un programme ou un ensemble de programmes servant à faire fonctionner des ordinateurs
2. les programmes et la documentation qui s'y rapporte
3. les programmes et, le cas échéant, la documentation associée ainsi que les autres données nécessaires au fonctionnement de l'ordinateur.

Sur la base de cette définition, les scripts, macros ou Infrastructure as Code (IaC) de moindre importance sont également considérés comme des logiciels. La LMETA se réfère au code source à l'article 9, alinéa 1. Selon cette interprétation, une publication au sens de la définition 1 suffirait. Une publication sans la documentation associée n'a toutefois que peu de valeur, le logiciel ne pouvant pas être utilisé au sens de la définition 2.

Pour réaliser des synergies avec des tiers, il faut se fonder sur la définition 3. S'agissant des données, cela signifie concrètement les données de base et les configurations de base (par exemple les valeurs d'énumération et les données de base qui devraient également être mises à disposition en open source).

Les petits projets, scripts ou exemples de code peuvent aussi être publiés ensemble dans un seul dépôt si l'autorité fédérale le juge approprié. Dans ce cas, les présentes instructions et les listes de contrôle correspondantes peuvent être remplies une seule fois.

La proportionnalité et la charge de travail doivent être prises en compte pour décider de ce qui est publié et comment.

3.1.2. Type de logiciel

Comme le montre la figure 1, on distingue entre développement propre, développement commun et projets de tiers. Chaque constellation a des implications différentes. Quelle que soit la manière dont le logiciel est développé, il relève de l'art. 9 LMETA.

Dans le cas du développement propre, la Confédération crée elle-même le logiciel ou le fait créer en conséquence. Elle choisit la gouvernance et conserve les droits sur le code source.

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

Si la Confédération contribue directement à un logiciel de tiers, ce code source relève également de l'art. 9 LMETA. Il faut s'assurer que cela se fasse dans l'intérêt des autorités fédérales, que la participation réponde aux exigences du projet et que l'administration fédérale respecte la gouvernance.

Cela se fait au moyen de la liste de contrôle Em002-2.1 Évaluation préalable OSS et de la liste de contrôle Em002-4.1 Communauté OSS.

3.1.3. Code existant

Pour les anciens logiciels (**logiciels existants**), la charge rétroactive liée à une publication est plus élevée que si celle-ci avait été planifiée dès le départ. Pour les logiciels existants, il ne vaut la peine de consentir cette charge que si un utilisateur potentiel souhaite utiliser le logiciel. Indépendamment de cela, la liste de contrôle Em002-2.1 Évaluation préalable OSS devrait être remplie afin de documenter les décisions déterminantes.

Les applications dont les autorités fédérales ont commencé le développement le 1er janvier 2024 ou après cette date, ou qui sont développées sur mandat de la Confédération sur la base d'un contrat conclu après le 1er janvier 2024, ne sont pas considérées comme existantes et relèvent dans tous les cas de l'obligation de publication prévue à l'art. 9, al. 1, LMETA.

L'obligation de publication prévue par la LMETA ne dépend pas de l'utilité pour des tiers.

3.1.4. Bibliothèques, plug-ins et add-ons

Dans certains cas, le logiciel n'est pas une application autonome, mais seulement des parties de celle-ci. La LMETA et l'ordonnance ne définissent pas plus précisément le code source. Les instructions s'appliquent également aux bibliothèques, plug-ins et add-ons découplés, qui relèvent de la LMETA.

Tâches :

- Recueillir les informations figurant dans Em002-5 Aide-mémoire Outils OSS. Celui-ci décrit tant des informations générales sur les logiciels open source que des informations spécifiques sur l'application de la LMETA.
- Vérifier si les conditions prévues par la LMETA sont remplies.
- Remplir la liste de contrôle Em002-2.1 Évaluation préalable OSS. En règle générale, il est judicieux d'associer directement les personnes de contact du fournisseur ou de l'équipe de développement.
- Au besoin, remplir la liste de contrôle Em002-4.1 Communauté OSS.

Décision : le logiciel doit-il être publié et comment cela doit-il se faire ?

3.2. Exception : droits de tiers

Il convient de renoncer à une publication si celle-ci violerait des droits de tiers. Si le logiciel a été développé par des employés de la Confédération, les droits appartiennent à l'employeur.[6] Dans les contrats de location de services et de prestations informatiques, les droits sont généralement transférés à la Confédération et devraient être invoqués.

Si le logiciel a été ou est développé sur la base des conditions générales de la Confédération suisse (état 2024),[7] les droits de propriété intellectuelle appartiennent au mandant, sauf convention contractuelle contraire.

Une application se compose typiquement de nombreux composants et éléments individuels. Pour certains types, cela devient problématique lorsqu'ils ne sont eux-mêmes pas open source ou ne peuvent pas être publiés sous une licence open source dans le cadre de l'application.

Bibliothèques

Selon le langage de programmation, les bibliothèques sont soit compilées directement avec l'application, soit liées, soit livrées sous forme de paquets. Elles font partie intégrante de l'application.

Si l'application contient des bibliothèques propriétaires ou soumises à licence, l'open sourcing devient plus difficile. En principe, le code source du reste de l'application peut être publié, mais les utilisateurs ou développeurs potentiels devraient acquérir la bibliothèque concernée avant que l'application ne devienne utilisable ou extensible.

Si la propriété de la bibliothèque problématique appartient au fournisseur de l'application, il vaut la peine de demander une clarification juridique afin de déterminer si la Confédération dispose de droits transférables sur ces bibliothèques.

Dans la mesure du possible, les bibliothèques propriétaires et soumises à licence devraient être évitées dans les développements propres ou remplacées si possible avant la publication. Au regard de l'objectif de l'article 9 LMETA, il convient de publier autant de fonctionnalités que possible utilisées par les administrations publiques. Les bibliothèques isolées qui empêchent la publication du code représentent donc une dette technique qui devrait être documentée et, le cas échéant, éliminée dès que l'occasion se présente.

Bases de données et autres systèmes de stockage

Si l'application utilise un système de stockage propriétaire et soumis à licence tel que Microsoft SQL Server, cela ne constitue pas un obstacle à une publication open source.

Il convient toutefois d'examiner si des bases de données ouvertes supplémentaires telles que PostgreSQL peuvent être prises en charge. Cela peut réduire les coûts d'exploitation et abaisser les barrières à l'entrée pour les utilisateurs potentiels.

Serveurs d'applications et systèmes d'exploitation

Si l'application nécessite des systèmes d'exploitation ou des serveurs d'applications sous licence (p. ex. Microsoft Windows Server ou RedHat JBoss), cela ne constitue pas un obstacle à une publication open source.

Nous recommandons de demander au fournisseur du logiciel si des bibliothèques tierces sous licence sont utilisées comme composants de l'application et, le cas échéant, lesquelles.

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

7. Conditions générales de la Confédération suisse, en particulier le chiffre 25.1 – https://www.bkb.admin.ch/bkb/de/home/themen/agb.html#accordion_21321111141713247349255

Si la création du logiciel a été acquise conjointement par plusieurs organisations ou d’autres institutions, la propriété appartient généralement à une association ou à une autre personne morale constituée 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). Les brevets n’empêchent pas fondamentalement qu’un logiciel soit publié en open source, mais ils peuvent en empêcher l’utilisation ou l’extension. Bien que les brevets soient plutôt rares en Suisse, il vaut généralement la peine de demander brièvement au fournisseur du logiciel si un examen a déjà eu lieu.

Acquisition des droits nécessaires

Pour satisfaire aux exigences légales de la LMETA, l’autorité fédérale a, en tant qu’adjudicateur, la possibilité de s’assurer les droits sur les résultats des travaux. Cette condition d’une publication peut être définie comme critère lors de l’acquisition. L’acquisition ultérieure des droits nécessaires sur le logiciel peut s’avérer plus complexe.

Si l’autorité fédérale ne souhaite pas publier elle-même le logiciel, elle peut par exemple déléguer cette tâche au mandataire ou transférer les droits de publication en conséquence. Dans ce cas, elle transfère les droits (et obligations) nécessaires au fournisseur ou à des tiers et les charge de la publication. Le droit est ainsi effectivement transféré à un tiers sous condition de publication ou de support du logiciel correspondant.

Ici aussi, [KKB-MB] contient des indications importantes.

Procédure

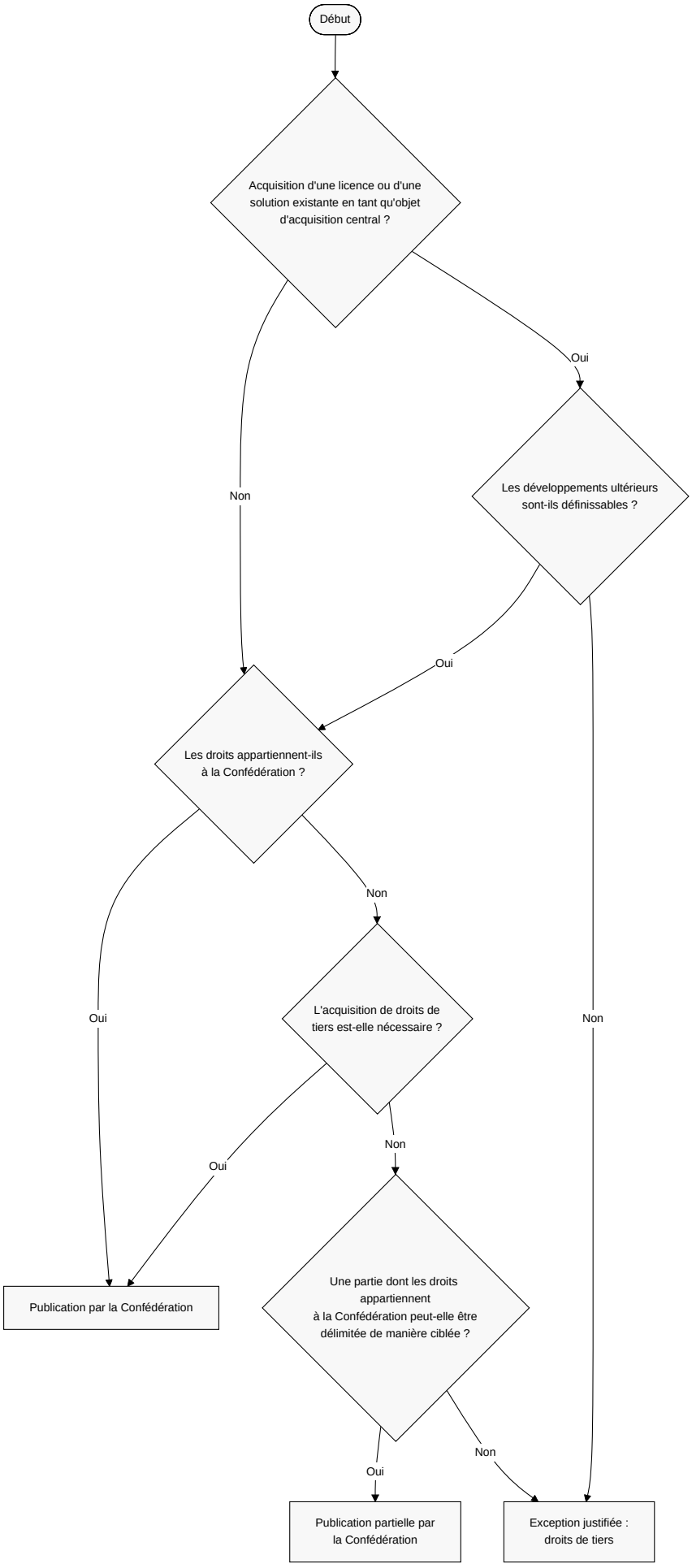

Début
Acquisition d'une licence ou d'une solution existante en tant qu'objet d'acquisition central ?
Oui
Non
Les développements ultérieurs sont-ils définissables ?
Oui
Non
Les droits appartiennent-ils à la Confédération ?
Oui
Non
L'acquisition de droits de tiers est-elle nécessaire ?
Oui
Non
Une partie dont les droits appartiennent à la Confédération peut-elle être délimitée de manière ciblée ?
Oui
Non
Publication par la Confédération
Publication partielle par la Confédération
Exception justifiée : droits de tiers

Figure 2 – Exception : droits de tiers (toujours tenir compte également de [KKB-MB] et de Em002-7 Aspects stratégiques des achats et des logiciels open source)

Tâches :

- Vérifier si l'organisation détient les droits de protection déterminants sur le logiciel.
- Vérifier si les droits peuvent être acquis.
- Au besoin, obtenir l'autorisation écrite des titulaires des droits.
- S'assurer que l'application ne contient aucune partie propriétaire ou protégée.
- Vérifier qu'aucun obstacle ne résulte d'une protection par brevet (dans la mesure du possible et du raisonnable).
- Vérifier si la publication doit être effectuée par un tiers.
- Vérifier si le développement doit reposer sur le développement de logiciels open source (au moyen du Em002-4 Guide des communautés OSS).

Décision : le logiciel peut-il être publié sans violer de droits de tiers ?

3.3. Exception : motifs relevant de la sécurité

Les logiciels qui ne peuvent pas être publiés pour des motifs relevant de la sécurité sont exemptés de publication.

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'outre les données de contenu, certains algorithmes et procédés visibles dans le code source ne doivent pas devenir publics (p. ex. capacités cyber offensives, détails de la détection des fraudes). La sécurité peut porter sur la confidentialité, l'intégrité, la disponibilité ou la traçabilité.

Recommandation :

Les logiciels devraient être délibérément développés de manière qu'aucun risque supplémentaire ne survienne même en cas de publication du code source. L'hypothèse selon laquelle la non-publication protège contre des attaques réussies est trompeuse (« security through obscurity »).

Procédure :

> **NOTE**
>
> Ce qui est déterminant est toujours le logiciel et non sa finalité. Le traitement de données critiques ne constitue pas a priori un aspect relevant de la sécurité.

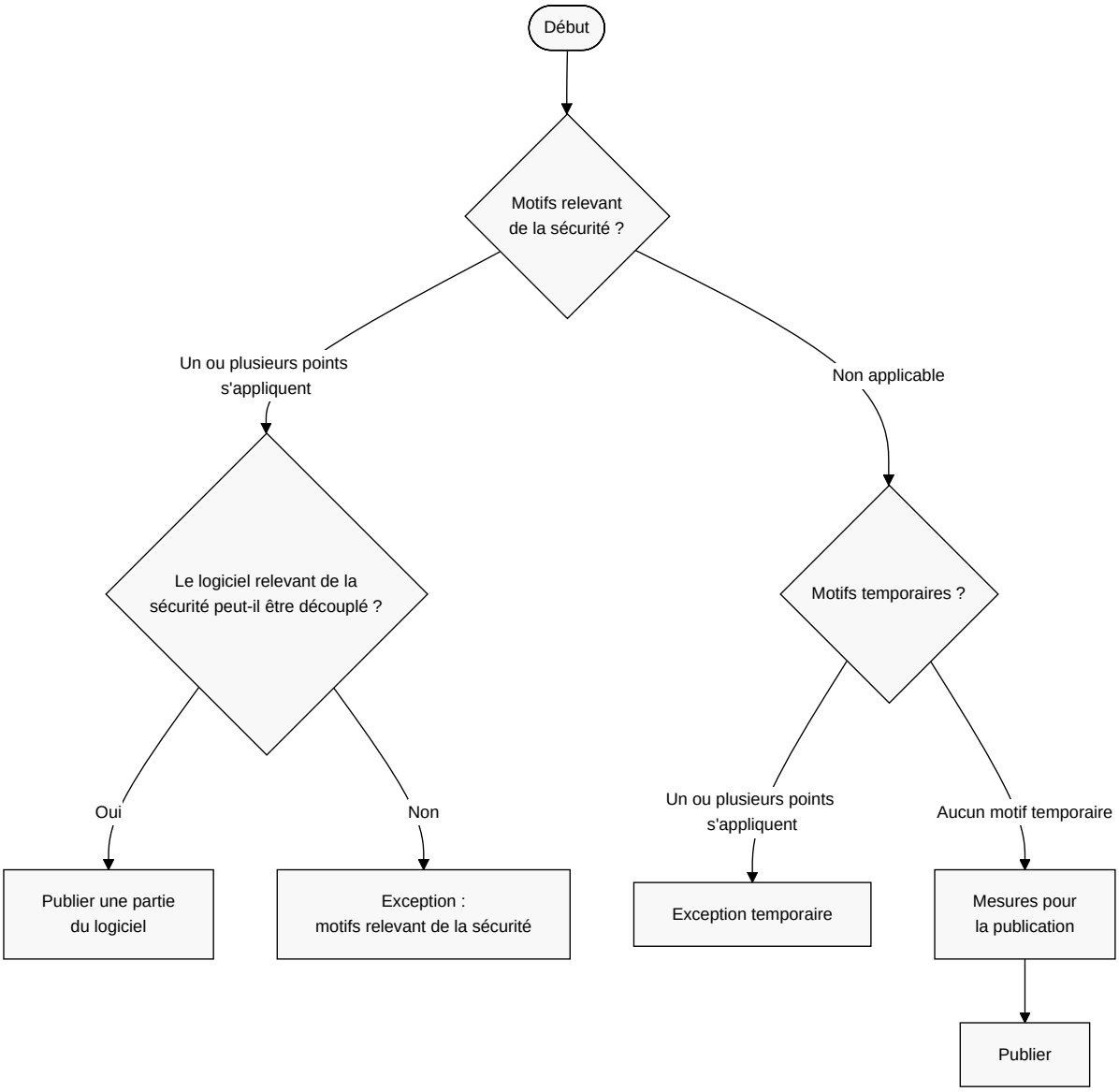


Figure 3 – Exception : motifs relevant de la sécurité

Les motifs suivants peuvent être considérés comme relevant de la sécurité selon le projet :

- La publication du logiciel augmenterait sensiblement un risque en matière de sécurité de l'information.[8]
- Pour les algorithmes de prévention de la fraude et les algorithmes analogues, il peut le cas échéant être renoncé à une publication.
- Technologies de cybersécurité particulièrement sensibles.
- Logiciels essentiels à l'exploitation d'infrastructures critiques.
- Lorsque la publication violerait d'autres bases légales.

Les motifs temporaires suivants peuvent encore empêcher une publication. Celle-ci sera toutefois rendue possible ultérieurement par des mesures appropriées.

8. Loi sur la sécurité de l'information, LSI. Art. 6 Sécurité de l'information. https://www.fedlex.admin.ch/eli/cc/2022/232/de#art_6

- Le code existant n'a pas encore pu être nettoyé et contient éventuellement des problèmes relevant de la sécurité.
- Les audits de sécurité du code n'ont pas encore été réalisés.
- Le processus de développement n'est pas encore assez mature pour permettre une publication sûre. Si le processus de développement n'est pas suffisamment établi, il est possible que des parties non destinées à des tiers soient publiées. Par exemple, les directives de commentaire du code et la génération de données de test doivent être adaptées au nouveau scénario de publication et fonctionner de manière fiable.

Les mesures suivantes permettent de publier un logiciel malgré des motifs relevant de la sécurité :

- Séparer du logiciel les configurations (p. ex. type de chiffrement utilisé).
- Externaliser dans des configurations ou des paramètres les informations techniquement sensibles et non publiques (p. ex. processus, calculs).
- Planifier, mettre en œuvre et exercer régulièrement des mesures organisationnelles, personnelles et techniques appropriées.
- Planifier et réaliser des activités de qualité appropriées lors de la publication. Voir le chapitre suivant « Analyse et préparation ».

Tâches :

- Examiner le logiciel quant aux motifs relevant de la sécurité qui empêcheraient une publication.
- Estimer la charge nécessaire pour éliminer les motifs temporaires.
- Planifier des mesures visant à séparer les informations relevant de la sécurité du code source du logiciel.
- S'assurer que l'application ne contient aucun procédé qui ne doit pas devenir public.

Décision :

Existe-t-il des motifs impératifs relevant de la sécurité qui rendent la publication impossible ?

3.4. Clarification de la publication

Cette étape permet de clarifier qui publiera le logiciel. Si le logiciel a été créé par un fournisseur externe, la publication peut aussi être confiée à ce fournisseur. Lors de cette étape, il importe de déterminer où le code source sera publié et d'établir une gouvernance correspondante.

Si le logiciel a été développé en interne à la Confédération et que celle-ci ne peut ou ne souhaite pas le publier elle-même, un mandat de publication peut être confié à un fournisseur ou les droits peuvent être cédés (transfert des obligations à un tiers).

Publication minimale et support

La LMETA ne prévoit aucune exigence concernant la publication. Une publication minimale du code source satisfait à l'exigence légale. Aucune autre activité n'est requise. La question de savoir si et dans quelle mesure un support est offert et avec quelle intensité le projet est entretenu peut être traitée au moyen du Em002-4 Guide des communautés OSS.

Développement de logiciels open source (OSSD)

Dans le développement de logiciels open source (OSSD), outre la publication du code source, l'ensemble du processus de développement se déroule publiquement. Tout, des exigences (issues) au code source, est transparent. L'OSSD s'appuie sur des outils de communication publics (listes de diffusion, forums, etc.), un système de gestion de versions (git), des listes d'erreurs et de fonctionnalités, une feuille de route et des outils de développement. Grâce à cette méthode de travail transparente, toute la communauté en profite et la collaboration au-delà des frontières organisationnelles devient possible. Avec cette approche, aucune activité de publication supplémentaire n'est nécessaire à l'achèvement du projet, celle-ci ayant déjà été prise en compte à son démarrage.

Tâches

- Établir une estimation sommaire de la charge pour la phase d'analyse. Selon la taille et la complexité de l'application, la charge varie entre trois jours et deux semaines.
- Libérer des ressources pour les étapes suivantes, en particulier pour l'analyse.
- Donner un accord de principe pour publier l'application en open source.

Décision :

L'accord de principe et la faisabilité pour publier l'application en open source sont donnés.

3.5. Minimiser la charge

Les listes de contrôle et les instructions sont conçues de manière à réduire au minimum la charge liée aux publications. Une planification anticipée et cohérente de la publication dès le début du projet ainsi qu'un développement orienté vers la publication minimisent la charge. Une certaine charge administrative et certains coûts sont inévitables [Le2023]. Ceux-ci peuvent le cas échéant être récupérés grâce à des communautés partageant les coûts de développement (voir Em002-4 Guide des communautés OSS).

3.6. Choix de la langue de publication

Le choix de la langue doit tenir compte à la fois du public cible potentiel et de la méthode de travail de l'équipe. Heureusement, les traductions sont devenues bien plus simples grâce aux nouveaux outils. Les langues officielles de la Confédération[9] et l'anglais entrent en ligne de compte.

Si le projet présente une pertinence internationale, l'anglais devrait être la langue cible. Pour une publication ultérieure, un changement de langue n'a naturellement guère de sens.

Au minimum, le fichier README.md devrait être disponible en plusieurs langues afin que les personnes intéressées puissent voir immédiatement si le projet est pertinent pour elles.

La communication publique devrait tenir compte en premier lieu du public cible. Il convient également de considérer qu'il s'agit d'une communication publique de la Confédération.

Remarque : avec la publication, l'administration fédérale emprunte un nouveau canal de communication publique. Si cela n'est pas fait avec soin et professionnalisme, l'image publique de la Confédération peut rapidement en pâtir.

4. Analyse et préparation

Objectif : les travaux nécessaires à une publication open source sont achevés. Les décisions relatives à la licence et au type de communauté ont été prises. Cela se fait au moyen de la liste de

9. https://www.fedlex.admin.ch/eli/cc/1999/404/fr#art_70

contrôle Em002-2.2 Analyse et préparation OSS.

4.1. Analyse du code source

Le code source à publier et les autres documents qui s'y rapportent sont analysés avant la publication. Cette analyse garantit qu'aucune information confidentielle n'est publiée.

Si un volume important de code source est prévu pour la publication, il est recommandé d'utiliser des outils automatisés appropriés pour les activités suivantes.

Les détails sont définis à la section E.

Tâches

- Vérifier comment la qualité du code existante et les directives ont été mises en œuvre (p. ex. ISO/IEC 25010:2023[10]).
- S'assurer qu'aucune donnée (de test) sensible n'est incluse, p. ex. des données fondées sur des personnes ou des cas réels.
- Mettre à jour la documentation du logiciel (voir annexe).
- Supprimer les fichiers inutiles ou les documents et données obsolètes.
- Vérifier le code source à la recherche de secrets ou d'identifiants.[11]
- Effectuer des tests de sécurité ciblés et mettre ensuite en place un programme de bug bounty (public).[12]
- Vérifier toutes les bibliothèques utilisées quant aux licences et établir les listes correspondantes (attributions). Si une bibliothèque est disponible sous plusieurs licences, cela doit être indiqué en conséquence dans les listes.
- Décrire le déploiement (pipeline CI/CD) et, le cas échéant, mettre en place une instance de démonstration.

Ces activités sont indépendantes de la publication, mais constituent des conditions préalables à une bonne qualité logicielle et à la garantie de la conformité. Le respect de ces mesures simplifie la collaboration, accroît la qualité des contributions externes et améliore l'image du logiciel ou de l'autorité.

4.2. Choix de la licence

Voir Em002-3 Guide de licences OSS.

Des textes de licence établis à l'échelle internationale devraient être utilisés lorsque cela est possible et judicieux. Les prétentions en responsabilité des preneurs de licence doivent être exclues dans la mesure permise par le droit.

Tâches

- Vérifier les éventuelles dépendances à des licences existantes.

10. ISO/IEC 25010:2023 - Systems and software Quality Requirements and Evaluation (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

- Pour le code de tiers, vérifier la compatibilité des conditions d'utilisation avec la licence prévue et remplacer le code si nécessaire.[13]
- Choisir la licence appropriée.

La compatibilité des licences peut être vérifiée au moyen d'outils tels qu'ORT Toolkit,[14] Black Duck,[15] FOSSA[16] ou FOSSology.[17] Le ToDo Group [18] propose une vue d'ensemble actualisée des outils.

Décision :

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

4.3. Documentation open source

Un minimum de documentation est attendu lors de la publication. Plusieurs documents se sont établis dans l'écosystème open source. Ils donnent aux visiteurs un aperçu rapide du logiciel, y compris de sa licence.

Les documents suivants sont attendus :

Document	Description / contenu :
README(.md)	Sert de document d'entrée et donne un aperçu rapide de l'objectif, du périmètre, du statut, de la licence et des groupes cibles du projet.
LICENSE	Description de la licence choisie dans le projet
CONTRIBUTING(.md)	Décrit le processus permettant de contribuer au projet.
CODE_OF_CONDUCT(.md)	Le code de conduite fixe les normes sociales attendues au sein du projet.
CHANGELOG(.md)	Tient une liste des modifications apportées aux versions du logiciel.

13. Voir Em002-3 concernant le choix de code ou de bibliothèques de tiers pour une pile technologique et concernant 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/

THIRD-PARTY-LICENSES(.md)	Description des licences de tiers ou des composants utilisés.
Getting started	Brève marche à suivre pour installer et utiliser le logiciel.
Create Gitignore	Décrit les documents qui ne seront pas publiés, p. ex. configurations, données de test ou contenus générés. Les informations protégées et sensibles devraient être gérées en dehors du projet, dans des outils et des espaces de stockage appropriés et dédiés.
publiccode.yml	Métadonnées structurées décrivant le logiciel, sur la base du standard https://yml.publiccode.tools/. Le format défini rend la publication plus facile à trouver et à réutiliser. De plus amples informations figurent à l'annexe F.

Tableau 1 – Documents open source

D'autres détails techniques sur ces documents sont décrits en annexe.

Tâches

- Créer la documentation standard.
- Faire figurer le copyright, les mentions de licence et la clause de non-responsabilité dans tous les fichiers.
- Selon la plateforme de publication, des descriptions supplémentaires peuvent être ajoutées.

Décision :

- Décision définitive de publier l'application en open source.

5. Publication et annonce

Objectif : l'application est publiée en open source et facilement repérable par des tiers. Tous les composants sont prêts pour la publication et la communauté est constituée ou établie. Cela se fait au moyen de la liste de contrôle Em002-2.3 Validation et publication OSS. Dans une certaine mesure, il s'agit de récapituler des décisions déjà prises.

5.1. Choix de la plateforme

Il existe plusieurs manières de publier du code source. Le tableau suivant donne une vue d'ensemble des systèmes de gestion de code source (SCM) et des plateformes possibles.

La plateforme appropriée doit être choisie avant la publication proprement dite. Certaines autorités fédérales sont déjà présentes sur GitHub.[19] Dans ce cas, il est judicieux d'utiliser les plateformes déjà établies.

Recommandation :

19. Le site https://ossbenchmark.com donne un aperçu des organisations et autorités actives sur GitHub.

Nous recommandons actuellement de mettre en place une organisation sur GitHub par autorité fédérale pour la publication du code source. Au sein de cette organisation, des projets (dépôts) correspondants peuvent être créés et les utilisateurs gérés avec les autorisations appropriées. À l'avenir, une plateforme fédérale centrale devrait être mise en place. Afin de garantir une gestion optimale des utilisateurs et de préserver la réputation de la Confédération suisse, il est important d'utiliser moins de dépôts, mais mieux gérés.

Plateforme	Caractéristiques
GitHub[20]	Plateforme en ligne de Microsoft, très répandue. Une grande quantité de logiciels open source est hébergée sur GitHub. Offre de nombreux outils supplémentaires en plus du SCM. Disponible en version Free et Enterprise (abonnement).
GitLab[21]	Plateforme en ligne de l'entreprise du même nom GitLab, elle-même publiée sous une licence open source. Offre de nombreux outils supplémentaires en plus du SCM. Peut aussi être exploitée localement, offrant ainsi un certain degré de souveraineté. Disponible en version Free et Enterprise (abonnement).
Bitbucket[22]	Plateforme en ligne d'Atlassian, spécifiquement intégrée à son écosystème. Disponible en version Free et Enterprise (abonnement).
Système SCM interne	Si une plateforme SCM propre est mise à disposition, elle peut aussi publier et rendre disponibles certains dépôts ou projets. L'exploitation doit être assurée.
Plateforme fédérale	Il n'existe actuellement aucune plateforme fédérale centrale pour la publication de logiciels. Une mesure en ce sens est proposée dans Em002 Lignes directrices stratégiques pour les logiciels open source dans l'administration fédérale.

20. https://github.com/about/
21. https://about.gitlab.com/
22. https://bitbucket.org/product/en

Site web, serveur FTP public, etc.	La LMETA ne définit pas comment un logiciel doit être publié. Une publication minimale[23] peut aussi se faire sur un site web ou d'autres ressources accessibles au public. Cela ne convient toutefois pas à une collaboration durable au sens de l'open source.

Tableau 2 – Plateformes de publication de logiciels open source

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

Tâches

- Évaluer la plateforme appropriée, y compris l'organisation ou le projet correspondant.
- Si ce n'est pas encore clarifié, déterminer la dénomination du projet ou du dépôt.
- Définir et vérifier les autorisations sur le dépôt.
- Assurer une gouvernance appropriée et une éventuelle réplication du code source.

Décision :

- La plateforme de publication a été choisie de manière appropriée.

5.2. Publication du logiciel

Lorsque les conditions sont clarifiées, le logiciel peut être publié.

Tâches

- Vérifier les conditions au moyen de la liste de contrôle Em002-2.3 Validation et publication OSS.
- L'accès à la plateforme est disponible et la gouvernance est clarifiée.
- Publication du code source et de la documentation par l'équipe de développement.

5.3. Communication

Après la publication proprement dite, d'autres activités de communication peuvent être lancées par et au sein de l'autorité fédérale concernée. Elles favorisent la diffusion et apportent la plus-value souhaitée de la publication.

Tâches

- Description dans l'hébergement du dépôt, paramètres de sécurité, etc.
- Communication interne et, le cas échéant, externe
- Rédiger un message à l'intention de la communauté.
- Faire connaître l'application au sein de l'organisation et dans les organes spécialisés, en promouvoir l'utilisation et les contributions.

23. Une publication minimale ne comprend que la publication du code source et d'autres artefacts minimaux. Ce type de publication satisfait aux exigences de la LMETA. Aucune plus-value n'est toutefois obtenue par cette publication.

- Créer un fichier publiccode.yml[24] dans le dépôt pour le catalogue OSS de l'administration fédérale.[25]

5.4. Constitution et entretien de la communauté

Lors de la publication d'un logiciel, une communauté peut se développer et contribuer au logiciel par des contributions (pull requests), des questions, des propositions de documentation, etc. Pour disposer d'une communauté active, une charge considérable est nécessaire tant pour la constitution que pour l'entretien continu. Une communauté qui fonctionne bien réduit au minimum les forks indésirables du logiciel. L'objectif est ici d'activer la plus-value de la communauté par des activités ciblées. Même en cas de publication minimale, la manière de traiter les retours et les questions sur le logiciel devrait être définie.

Tâches

- Clarifier qui sont les utilisateurs potentiels ou les personnes intéressées.
- Déterminer la forme appropriée pour l'application concrète au moyen du Em002-4 Guide des communautés OSS.
- Constituer la communauté.
- Activités de constitution de la communauté.

6. Annexe

6.1. Modifications par rapport à la version précédente

- Chapitres 4.3 et 5.3 : publication complétée par publicode.yml
- L'annexe F « Public Code » a été ajoutée
- Diverses modifications et améliorations rédactionnelles mineures

6.2. Références

Voir *Em002 Lignes directrices stratégiques pour les logiciels open source dans l'administration fédérale*.

6.3. Abréviations

Voir Em002 Lignes directrices stratégiques pour les logiciels open source dans l'administration fédérale et Em002-6 FAQ sur l'OSS.

6.4. Documentation d'accompagnement **Em002-2.2 Liste de contrôle Analyse et préparation OSS – Documentation**

6.5. Documentation du code source

La documentation devrait faire partie du code source du projet afin que les modifications de la documentation soient enregistrées de manière infalsifiable et que la documentation soit versionnée parallèlement au projet. Cela garantit automatiquement que la documentation, si elle est correctement

24. Le standard de métadonnées https://yml.publiccode.tools/ décrit le logiciel. Le format défini rend la publication plus facile à trouver et à réutiliser. De plus amples informations figurent à l'annexe F.
25. Catalogue OSS : https://www.opensource.admin.ch

entretenue, ne diverge pas de l'état de développement du projet et que la documentation des versions antérieures du projet peut être consultée à tout moment.

En outre, cela donne à des tiers la possibilité de contribuer facilement à des modifications de la documentation.

Markdown devrait être utilisé comme langage de balisage pour la mise en forme du texte, car il est facile à écrire, largement répandu et lisible par machine. De plus, les documents Markdown sont affichés dans un format cible bien lisible par toutes les plateformes courantes. Les documents Markdown peuvent être rendus dans n'importe quel format cible indépendamment de la plateforme utilisée, au moyen de générateurs de sites statiques tels que Jekyll, MkDocs, Gatsby ou Sphinx, par exemple avec les prescriptions d'identité visuelle d'une administration.

6.6. Destinataires et structure de la documentation

La documentation de projet devrait s'adresser tant aux utilisateurs finaux qu'aux spécialistes techniques et ne pas se concentrer uniquement sur la technique. Pour le choix de la langue, voir le chapitre 3.6. Cela favorise la diffusion du logiciel.

Un fichier README.md sert de document d'entrée. Les fichiers portant ce nom sont reconnus par toutes les plateformes courantes et affichés comme point d'entrée de la documentation.[26]

README.md devrait donner un aperçu rapide de l'objectif, du périmètre, du statut, de la licence et des groupes cibles du projet. Concrètement, README.md devrait contenir les points suivants :

- Nom du logiciel
- Brève description de l'objectif et de la fonction du logiciel. Périmètre et limites possibles
- Instructions d'installation
- Utilisation du logiciel
- Renvoi à une éventuelle instance de démonstration
- Coordonnées de support et de contact
- Comment contribuer au logiciel open source (Contributing)
- Licence utilisée (Licence)
- Statut du projet / état de développement

6.7. Mettre en avant l'open source et déposer la licence

Directement après l'énoncé de mission, il convient d'indiquer clairement et sans ambiguïté que le projet est un logiciel open source. La licence utilisée par le projet devrait être indiquée avec l'identifiant SPDX correspondant[27] et liée au texte de licence concret figurant dans le fichier LICENSE. Les modèles de licence comportent généralement une partie variable (p. ex. nom du projet, année de copyright, nom de l'auteur) qui doit être adaptée dans le fichier LICENSE.

La plupart des plateformes reconnaissent les licences figurant dans un fichier LICENSE et offrent des informations claires sur la licence, p. ex. un bref résumé des conditions de licence.

26. Pour plus d'informations sur la création d'un fichier readme : https://www.makeareadme.com/
27. Identifiant SPDX – https://spdx.org/about

Il est recommandé d'ajouter un en-tête minimal de licence et de copyright à chaque fichier source créé. Les en-têtes existants dans les fichiers existants ne devraient pas être modifiés. En cas de contribution à un projet disposant de règles claires sur les en-têtes, celles-ci doivent être respectées.

Pour plus de détails, voir : Producing Open Source Software: State That the Project is Free.[28]

6.8. État de développement actuel

Afin de montrer rapidement l'état du projet, l'état de développement actuel devrait être documenté dans README.md. Il est important pour les lecteurs de savoir si le projet est dans un état mature ou au début de son développement, s'il est activement entretenu et à quelle fréquence de nouvelles versions sont publiées.

Cette section peut décrire quel type de soutien de tiers est actuellement le plus important pour le projet. Il pourrait s'agir par exemple d'un développeur disposant de connaissances technologiques spécifiques ou d'une personne chargée de remanier la documentation du projet.

Pour plus de détails, voir Producing Open Source Software: Development Status.[29]

6.9. Instance de démonstration

Pour les applications, il est recommandé de mettre à disposition une instance de démonstration afin que l'application puisse être essayée sans effort d'installation. Selon le type d'application, l'instance de démonstration peut être l'installation productive ou un environnement de test. Il importe que les lecteurs puissent accéder à l'instance correspondante de la manière la plus autonome et la plus directe possible.

En alternative, de nombreux projets open source fournissent des outils permettant de démarrer facilement l'application. Cela peut se faire par exemple au moyen de conteneurs, d'applications portables ou de scripts d'installation.

6.10. Personne de contact spécialisée et propriété du projet

Le fichier README.md devrait nommer la principale personne de contact spécialisée afin que d'autres administrations et organisations intéressées puissent prendre contact le plus directement possible.

Les adresses électroniques personnelles ne devraient pas être utilisées.

Le fichier README.md devrait aussi décrire brièvement quelle organisation est propriétaire du projet, en particulier si une association a été constituée pour celui-ci.

7. Documentation d'installation

Pour les applications, README.md devrait renvoyer à une documentation d'installation décrivant les exigences matérielles et logicielles, les composants d'infrastructure utilisés (par exemple les bases de données) ainsi que la manière d'installer et d'exploiter l'application.

7.1. Developer Guidelines

Le point d'entrée des Developer Guidelines devrait être déposé dans un fichier Markdown CONTRIBUTING.md, lié dans README.md. Les Developer Guidelines font partie de la documentation étendue destinée aux développeurs qui souhaitent contribuer au projet et se concentrent avant tout sur la collaboration et la communication au sein du projet plutôt que sur la technique.

---

28. https://producingoss.com/en/getting-started.html#state-freedom
29. https://producingoss.com/en/getting-started.html#state-freedom

Les Developer Guidelines devraient contenir les points suivants et renvoyer le cas échéant aux documents correspondants :

- Un renvoi au code de conduite du projet. Le code de conduite fixe les normes sociales attendues au sein du projet.
- Il est recommandé de choisir comme code de conduite le Contributor Covenant Code of Conduct, placé sous licence CC-BY-4.0,[30] ou de s'en inspirer.[31]
- Une indication précisant que le projet effectue des revues de code sous forme de pull requests GitHub, avec un renvoi à la documentation GitHub sur les pull requests.

Enfin, il convient de renvoyer à la Developer Documentation.

7.2. Developer Documentation

Alors que les Developer Guidelines décrivent la collaboration et les normes sociales du projet, la Developer Documentation se concentre sur les aspects techniques de la participation au développement du projet. Elle devrait décrire au moins les points suivants :

- Quelles dépendances techniques le projet présente et quels outils sont nécessaires à son développement.
- Quelles règles formelles s'appliquent au code source du projet, par exemple en matière de formatage du code source.
- Comment le code source du projet est organisé et comment les fonctions peuvent être testées techniquement.
- Une esquisse d'architecture avec une description sommaire de l'architecture, de préférence avec délimitation du contexte et vue sommaire des composants.[32]
- Pour les applications : comment l'application peut être démarrée à des fins de développement ou de débogage et quels composants d'infrastructure sont nécessaires à cet effet.

8. Documentation d'accompagnement de la liste de contrôle Em002-2.2 Analyse et préparation OSS – Code source

8.1. Historique du code source

Le code source d'un projet est généralement conservé dans un système de gestion de code source (SCM) tel que GitHub. Ces systèmes enregistrent non seulement l'état actuel du code source, mais aussi les modifications qui y ont été apportées au fil du temps. Cela comprend en particulier les suppressions dans le code source, y compris les fichiers supprimés.

En outre, le SCM contient des métainformations sur les modifications du code source, telles que le nom et l'adresse électronique de l'auteur, ou des signatures PGP pour les commits et tags signés.

8.2. Données sensibles, protégées ou confidentielles

Le code source et son historique peuvent contenir des données accessibles au public, telles qu'un répertoire de codes postaux, des données générées aléatoirement ou des données synthétiques.

30. https://creativecommons.org/licenses/by/4.0/
31. https://www.contributor-covenant.org/version/1/4/code-of-conduct.html
32. Cadre d'architecture MMB (méthode de modélisation de la Confédération) ou arc42. http://arc42.org/

Le code source et son historique ne doivent contenir aucune donnée ou information (personnelle) sensible ou confidentielle au sens de la loi fédérale sur la protection des données (LPD) et de la loi sur la sécurité de l'information (LSI). Cela comprend en particulier :

- Les noms d'utilisateur et mots de passe ou autres secrets tels que clés privées, jetons d'accès ou certificats.
- Les noms DNS internes, URL, noms d'hôte, adresses IP, structures de réseau ou noms de lecteurs.
- Les noms, adresses, images ou données personnelles similaires sans le consentement de la personne concernée
- Les données anonymisées ou pseudonymisées (car il est possible d'annuler l'anonymisation ou la pseudonymisation)

Même si l'état actuel du code source est exempt de données ou d'informations sensibles, protégées ou confidentielles, de telles données ou informations peuvent encore être extraites de l'historique du système SCM si elles ont un jour fait partie du code source ou des métainformations du SCM. Le nettoyage de l'état actuel du code source ne supprime pas complètement ces informations.[33]

En cas de doute, il est donc recommandé de supprimer entièrement l'historique après le nettoyage du code source, avant de publier celui-ci.

Les noms et adresses électroniques des auteurs du code source sont généralement disponibles directement dans le code source ou dans les métadonnées du SCM, par exemple dans les commits GitHub ou les tags GitHub annotés. D'un point de vue juridique, cela ne pose pas de problème, l'entreprise de développement étant chargée de régler la publication de telles informations. Il est toutefois recommandé de sensibiliser l'entreprise de développement ou les employés internes de la Confédération au fait que les noms et adresses électroniques des auteurs du code source peuvent être divulgués lors de la publication du logiciel.

8.3. Suppression des informations sur les auteurs

L'historique du logiciel peut être modifié afin de masquer les informations sur les auteurs. Des outils[34] permettent d'automatiser cette étape dans une large mesure.

La suppression des informations sur les auteurs supprime l'attribution, la traçabilité et les coordonnées de contact. La charge que cela représente ne doit pas être sous-estimée, selon le volume de code source. Pour ces raisons, la suppression des auteurs n'est pas recommandée. Les noms peuvent être publiés si a) le consentement des personnes concernées a été obtenu et b) aucun motif relevant de la sécurité ne s'y oppose.

8.4. Prise en compte des licences des bibliothèques

Pratiquement chaque projet utilise des bibliothèques de tiers (bibliothèques logicielles, dépendances) afin d'accéder à des fonctions couramment utilisées sans devoir les programmer. Ces bibliothèques dépendent généralement d'autres bibliothèques, ce qui engendre des dépendances à plusieurs niveaux (dépendances transitives). Selon le langage de programmation, même de petits projets peuvent compter des centaines de dépendances.

Les résultats de cette clarification doivent être pris en compte lors du choix de la licence conformément au Em002-3 Guide de licences OSS et dans la liste de contrôle Em002-2.3 Validation

33. L'outil BFG Repo-Cleaner offre une possibilité de nettoyer un dépôt
34. https://www.adamdehaven.com/blog/update-commit-history-author-information-for-git-repository/

et publication OSS.

Pour chacune de ces dépendances, il convient de s'assurer que sa licence est compatible avec celle du projet. Sont particulièrement problématiques les dépendances qui ne sont pas sous licence open source, qui n'ont déclaré aucune licence ou qui utilisent une licence virale ne correspondant pas à celle du projet. Certaines bibliothèques sont placées sous plus d'une licence et laissent à l'utilisateur le choix de l'une d'elles.

- Dans tous les principaux langages de programmation, les dépendances du projet (y compris les dépendances transitives) et leurs licences peuvent être listées automatiquement. Il s'agit notamment de :
- Le plugin Project Info Reports pour le gestionnaire de paquets Apache Maven pour les projets Java, qui génère au moyen du rapport de dépendances une sortie HTML de toutes les bibliothèques utilisées, y compris la licence déclarée.[35]
- Le NPM Licence Checker pour le gestionnaire de paquets NPM pour les projets JavaScript, qui peut restituer toutes les bibliothèques utilisées, y compris la licence déclarée, dans différents formats, p. ex. CSV.[36]
- Le Licence Finder de Pivotal, qui prend en charge divers gestionnaires de paquets et langages de programmation, dont Ruby Gems, Python Eggs et Godeps.[37]

9. Utilisation de bibliothèques sous licence propriétaire

Les bibliothèques sous licence propriétaire sont en principe problématiques dans les projets open source et ne peuvent généralement pas être utilisées, en particulier si le code source a été publié sous une licence virale telle que l'AGPL.

Veuillez vous adresser au fournisseur et au service juridique de l'office concerné afin de vérifier si et à quelles conditions la bibliothèque propriétaire peut être utilisée dans votre projet.

9.1. Utilisation de gestionnaires de paquets

Pratiquement tous les langages de programmation répandus proposent des gestionnaires de paquets pour gérer leurs dépendances, tels qu'Apache Maven pour les projets Java, NPM pour les projets JavaScript ou NuGet pour les projets .NET.

Lors de l'utilisation d'un gestionnaire de paquets, les dépendances utilisées par le projet ne font plus directement partie du code source de celui-ci, par exemple par la copie du code source de la dépendance dans le projet ou par la copie des dépendances sous forme binaire. Le projet obtient au contraire toutes ses dépendances via les déclarations du gestionnaire de paquets.

Cela permet de lister et de référencer correctement toutes les dépendances. Cela empêche en outre la modification involontaire du code des dépendances. Si des modifications sont nécessaires, elles doivent être effectuées directement à la source de la dépendance.

9.2. Déclaration de la licence du projet dans le descripteur du gestionnaire de paquets

Le projet devrait déclarer sa licence dans le descripteur approprié du gestionnaire de paquets, au moyen de l'identifiant SPDX,[38] afin que la licence du projet puisse être restituée par le gestionnaire

35. Infos sur le plugin https://maven.apache.org/plugins/maven-project-info-reports-plugin/plugin-info.html
36. NPM Licence Checker https://github.com/davglass/license-checker
37. Licence Finder de Pivotal https://github.com/pivotal/LicenseFinder
38. Identifiant SPDX – https://spdx.org/

de paquets. Cela est particulièrement important pour les bibliothèques, afin que leurs utilisateurs puissent interroger automatiquement leur licence via le gestionnaire de paquets.

9.3. Attribution et mentions de copyright

De nombreuses bibliothèques sont soumises à une licence qui exige de leur utilisateur une mention de copyright d'origine ou une attribution.

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 les autres, voir la liste des licences reconnues par l'OSI.[39]

Dans la pratique, cela concerne presque toutes les bibliothèques utilisées. Comme le nombre de bibliothèques utilisées est déjà très élevé dans les petits projets, une mention de copyright juridiquement correcte ou l'attribution nécessaire n'est possible qu'au prix d'efforts considérables.

Il est donc recommandé de procéder comme suit :

- Établir une liste des bibliothèques utilisées, au moyen des mécanismes du gestionnaire de paquets ou d'outils tels que le NPM Licence Checker ou le Licence Finder de Pivotal.
- Préparer la liste manuellement afin qu'elle contienne les informations suivantes : nom officiel du projet de la bibliothèque, site web du projet de la bibliothèque, licence utilisée pour la bibliothèque sous forme d'identifiant SPDX avec un lien vers le texte de la licence.
- Déposer la liste dans le code source du projet, p. ex. dans un fichier THIRD-PARTY-LICENSES.md.
- Pour les applications : intégrer un lien pointant vers THIRD-PARTY-LICENSES.md dans une boîte de dialogue « À propos » ou « À propos de cette application ».
- La boîte de dialogue « À propos » est largement utilisée à cette fin, par exemple dans Google Chrome (dans la boîte de dialogue « À propos de Chrome »).
- Pour les bibliothèques : renvoyer dans README.md à THIRD-PARTY-LICENSES.md.
- Dans le cadre du processus de revue de code, s'assurer que les modifications des bibliothèques sont reportées dans THIRD-PARTY-LICENSES.md.

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

Ce projet utilise des logiciels open source :

- Spring Boot, projects.spring.io/spring-boot sous licence Apache-2.0
- caniuse-db, GitHub sous licence CC-BY-4.0

10.1. Documentation d'accompagnement de la liste de contrôle Em002-2.3 Validation et publication OSS

39. Les licences open source sont des licences conformes à l'Open Source Definition — en bref, elles permettent d'utiliser, de modifier et de partager librement un logiciel. https://opensource.org/licenses

10.1.1. Publication avec publiccode.yml

Publiccode.yml est un standard de métadonnées pour les dépôts de logiciels contenant des logiciels développés ou acquis par des administrations publiques. L'objectif est de rendre ces logiciels plus faciles à trouver et à réutiliser. Le standard de métadonnées est établi à l'échelle internationale et décrit publiquement à l'adresse https://yml.publiccode.tools/. Un fichier publiccode.yml contient typiquement des informations telles que le titre et la description (possible en plusieurs langues), l'état de développement, les coordonnées de contact, les informations de licence, etc. Des extensions spécifiques au pays sont prévues pour la Suisse.

Un fichier yml doit être créé dans le répertoire principal du dépôt pour chaque logiciel. Si la solution se compose de plusieurs dépôts (auxiliaires), un seul fichier suffit, celui-ci décrivant le cas d'application de la solution.

À partir des données structurées contenues dans les fichiers publiccode.yml, un catalogue OSS (opensource.admin.ch) est généré pour l'ensemble de l'administration fédérale. Le catalogue offre une vue d'ensemble des logiciels publiés et facilite la recherche et la réutilisation de solutions existantes.

Si un nouveau dépôt ou une nouvelle organisation est créé, cela doit être annoncé par courriel à opensource@bk.admin.ch en vue de l'inscription au catalogue OSS.

Outre le catalogue OSS, un éditeur graphique et un validateur pour les fichiers publiccode.yml sont également disponibles.

Catalogue OSS et éditeur : https://www.opensource.admin.ch/

10.2. Autres sources et licence

La liste de contrôle, la documentation d'accompagnement et les modèles qui s'y rapportent se fondent en partie sur les sources suivantes :

- Google Open Source Docs de Google LLC, sous licence CC-BY-4.0[40]
- Producing Open Source Software, How to Run a Successful Free Software Project de http://www.red-bean.com/kfogel C-BY-SA-4.0[41]
- GitHub Healthy Contributions[42]
- Standard for Public Code[43]

40. Google Open Source Docs – https://opensource.google.com/docs/
41. Producing Open Source Software, How to run a Successful Free Software Project - https://producingoss.com/
42. GitHub Healthy Contributions
43. Standard for Public Code – https://standard.publiccode.net/