Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Bundeskanzlei BK

Digitale Transformation und IKT-Lenkung DTI

14.09.2026

Em002-2 Anleitung zur Veröffentlichung von OSS

Version 2.0-b565cab

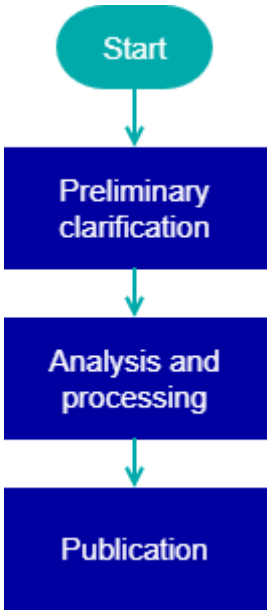
Inhaltsverzeichnis

Haftungsausschluss: Dieses Dokument ist ein laufend weiterentwickelter Entwurf und Teil der Hilfsmittel, welche die Bundesverwaltung bei der Veröffentlichung von Open-Source-Code unterstützen.[1][2][3][4] Weitere Informationen finden Sie im README.

1. Management Summary

Diese Anleitung beschreibt das Vorgehen für die Offenlegung von Quellcode von Software, die Bundesbehörden zur Erfüllung ihrer Aufgaben entwickeln oder entwickeln lassen, gemäss Art. 9 EMBAG.[5] Ausnahmen bestehen, wenn Rechte Dritter oder sicherheitsrelevante Gründe die Offenlegung ausschliessen oder einschränken. Veröffentlichung bedeutet, dass alle die Software nutzen, weiterentwickeln und verbreiten dürfen. Es werden keine Lizenzgebühren erhoben.

Diese Anleitung richtet sich an Personen, die für die Veröffentlichung von Quellcode verantwortlich sind und diese umsetzen.



Der Gegenstand gliedert sich in drei Hauptteile:

- Der erste Teil (Kapitel 3) behandelt die **Vorabklärungen** und die Ausnahmen nach EMBAG.
- Der zweite Teil (Kapitel 4), **Analyse und Vorbereitung**, untersucht den Quellcode und bereitet ihn nach Bedarf auf. Zudem wird die Lizenzwahl getroffen.
- Der dritte Teil (Kapitel 5), **Veröffentlichung und Bekanntmachung**, beschreibt die eigentliche Veröffentlichung und weitere Massnahmen wie Kommunikation und Aufbau einer geeigneten Community.

Drei Checklisten begleiten und dokumentieren den Prozess.

1. Empfehlung für die Informatik der Bundesverwaltung gemäss [P035] *Kapitel 4.6*
2. Zu den Definitionen der Klassifizierungen INTERN und VERTRAULICH siehe die *Verordnung vom 8. November 2023 über die Informationssicherheit bei Bund und Armee (ISV; SR 128.1)*
3. Siehe Fussnote 1
4. Planungsbereiche gemäss der *IKT-Strategie des Bundes 2020–2023 vom 3. April 2020 (SB000)*
5. Bundesgesetz *[EMOTA2023]*

Technische Ergänzungen sind im Anhang aufgeführt.

2. Einleitung

Bei der Freigabe von Open-Source-Software ist zu unterscheiden zwischen dem **Beitragen von Quellcode und Dokumentation zu bestehender Open-Source-Software** und der **Veröffentlichung als eigenständiges Open-Source-Projekt**.

Ersteres umfasst typischerweise Fehlerbehebungen und Funktionserweiterungen. Je nach Lizenztyp und Softwareeinsatz muss oder kann der Quellcode unter der bestehenden Lizenz des Open-Source-Projekts freigegeben werden. Möglicherweise ist eine Freigabevereinbarung einzuhalten.

Im zweiten Fall, dem Start eines neuen Open-Source-Projekts, können Governance und Lizenz grundsätzlich frei gewählt werden. Zu berücksichtigen ist einzig die Lizenz, unter der allfällige in das neue Projekt integrierte Softwareelemente veröffentlicht sind. Weitere Einzelheiten zur Lizenzwahl finden sich im Dokument Em002-3 Leitfaden OSS-Lizenzierung. Lässt sich für die Bundesverwaltung aus einer Community ein Mehrwert erzielen oder will die Bundesverwaltung ein Ökosystem schaffen, so ist zusätzlich die Em002-4.1 Checkliste OSS-Community gemäss Em002-4 Leitfaden OSS-Community auszufüllen.

Die Freigabe von Open-Source-Software ist mehr als eine rein technische Massnahme. Sie bedeutet auch, eine entsprechend offene Kultur zu etablieren und die Menschen mitzunehmen. Eine erfolgreiche Open-Source-Kultur ist eine Mischung aus Offenheit, technischer Exzellenz, starker sozialer Interaktion und gemeinsamem Engagement.

Die folgende Grafik beschreibt die Anleitung und die verschiedenen Ausprägungen.

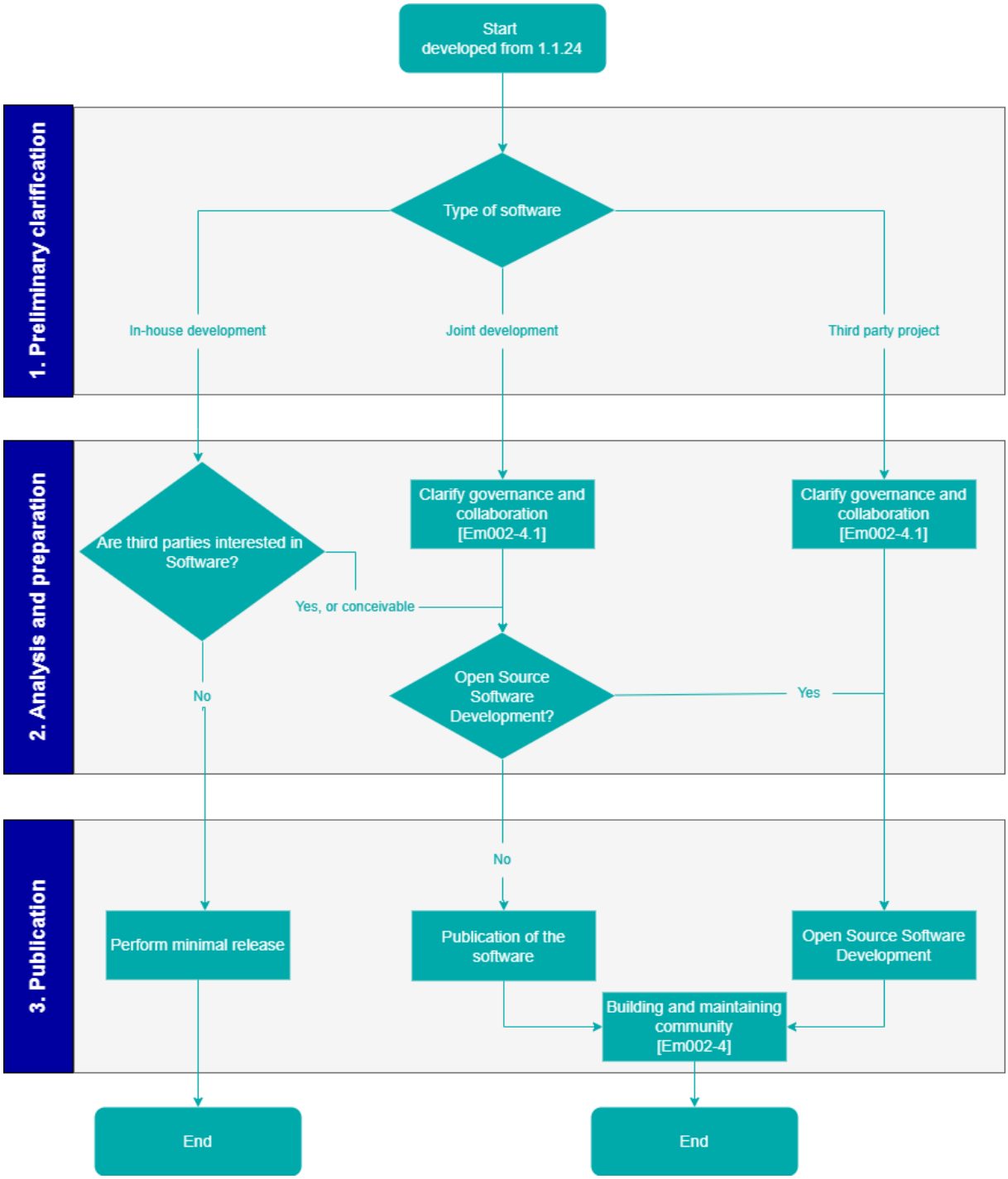


Abbildung 1 – Entscheidungsbaum für die Softwarefreigabe

3. Prozess der Vorabklärung

Ziel: Die Vorteile und möglichen Folgen einer Veröffentlichung der Software verstehen. Entscheiden, ob die Software veröffentlicht werden kann und, falls ja, ob sich der Aufbau einer aktiven Community lohnt. Hinweis: Die Em002-2.1 Checkliste OSS-Vorabklärung sollte möglichst früh im Projekt ausgefüllt werden, da dies die Art der Softwareentwicklung beeinflussen kann.

3.1. Freigabe nach EMBAG

Nach Art. 9 des Bundesgesetzes vom 17. März 2023 [EMOTA2023] über den Einsatz elektronischer Mittel zur Erfüllung von Behördenaufgaben (EMBAG) muss die Bundesverwaltung (konkret Art. 2 Abs.

1) den Quellcode von Software offenlegen, die sie zur Erfüllung ihrer Aufgaben entwickelt oder entwickeln lässt. Ausnahmen gelten, wenn Rechte Dritter oder sicherheitsrelevante Gründe dies ausschliessen oder einschränken.

Für im Auftrag des Bundes erstellte Individualsoftware gelten grundsätzlich die Allgemeinen Geschäftsbedingungen des Bundes [BBL-AGB]. Nach Ziffer 25.1 sehen diese vor, dass das Eigentum an Quellcode und Dokumentation an den Leistungsbezüger (den Bund) übergeht. Wurde die Software gemeinsam von mehreren Organisationen oder anderen Institutionen beschafft, so ist zu prüfen, wer die Rechte am Code hat. Beispielsweise kann das Eigentum bei einem zu diesem Zweck gegründeten Verein liegen.

Die blosse Nutzung von Standardsoftware fällt nicht unter Art. 9 EMBAG, und eine Freigabe wäre in der Regel ohnehin nicht möglich, da die Rechte an Standardsoftware häufig beim Anbieter verbleiben (AGB Bund für die Beschaffung von Standardsoftware). Individuelle Erweiterungen von Standardsoftware eignen sich hingegen durchaus für eine Veröffentlichung. Normalerweise stehen diese im Eigentum des Bundes. Reine Konfigurationsanpassungen eignen sich weniger für eine Veröffentlichung. Das Thema wird in Em002-7 Strategische Aspekte der Beschaffung und Open-Source-Software sowie in den Dokumenten [KKB-MB] und [BBL-WL] behandelt. Grundsätzlich unterstehen Erweiterungen stets Art. 9 EMBAG.

3.1.1. Software im Verhältnis zum EMBAG

Die aktuelle Norm ISO/IEC 24765 enthält drei Definitionen für Software:

1. ein Programm oder eine Menge von Programmen zum Betrieb von Computern
2. Programme und die zugehörige Dokumentation
3. Programme und, sofern zutreffend, zugehörige Dokumentation sowie weitere für den Computerbetrieb notwendige Daten.

Auf Basis dieser Definition gelten auch kleinere Skripte, Makros oder Infrastructure as Code (IaC) als Software. Das EMBAG bezieht sich in Artikel 9 Absatz 1 auf den Quellcode. Nach dieser Auslegung würde eine Veröffentlichung nach Definition 1 genügen. Eine Veröffentlichung ohne zugehörige Dokumentation ist jedoch von geringem Wert, da die Software nach Definition 2 nicht genutzt werden kann.

Um Synergien mit Dritten zu erzielen, ist Definition 3 zugrunde zu legen. Bezüglich Daten bedeutet dies konkret Stammdaten und Grundkonfigurationen (beispielsweise Aufzählungswerte und Grunddaten, die ebenfalls als Open Source verfügbar gemacht werden sollten).

Kleinere Projekte, Skripte oder Codebeispiele können auch gemeinsam in einem Repository veröffentlicht werden, wenn die Bundesbehörde dies als sinnvoll erachtet. In diesem Fall können diese Anleitung und die entsprechenden Checklisten einmalig ausgefüllt werden.

Bei der Entscheidung, was und wie veröffentlicht wird, sind Verhältnismässigkeit und Aufwand zu berücksichtigen.

3.1.2. Art der Software

Wie in Abbildung 1 dargestellt, wird zwischen Eigenentwicklung, gemeinsamer Entwicklung und Drittprojekten unterschieden. Jede Konstellation hat andere Auswirkungen. Unabhängig davon, wie die Software entwickelt wird, fällt sie unter Art. 9 EMBAG.

Bei der Eigenentwicklung erstellt der Bund die Software selbst oder lässt sie entsprechend erstellen. Er wählt die Governance und behält die Rechte am Quellcode.

Bei der gemeinsamen Entwicklung erstellt der Bund die Software zusammen mit anderen Organisationen. Governance und Rechte an der Software sind über die gewählte Organisationsform zu regeln.

Trägt der Bund direkt zu Drittsoftware bei, so fällt auch dieser Quellcode unter Art. 9 EMBAG. Es ist sicherzustellen, dass dies im Interesse der Bundesbehörden geschieht, dass die Beteiligung den Projektanforderungen entspricht und dass sich die Bundesverwaltung an die Governance hält.

Dies erfolgt anhand der Em002-2.1 Checkliste OSS-Vorabklärung und der Em002-4.1 Checkliste OSS-Community.

3.1.3. Legacy-Code

Bei alter Software (**Legacy-Software**) ist der nachträgliche Aufwand für eine Veröffentlichung höher, als wenn die Freigabe von Anfang an geplant war. Bei Legacy-Software lohnt sich dieser Aufwand nur, wenn eine potenzielle Nutzerin oder ein potenzieller Nutzer die Software einsetzen will. Unabhängig davon sollte die Em002-2.1 Checkliste OSS-Vorabklärung ausgefüllt werden, um die massgebenden Entscheide zu dokumentieren.

Anwendungen, mit deren Entwicklung Bundesbehörden am oder nach dem 1. Januar 2024 begonnen haben oder die im Auftrag des Bundes auf der Grundlage eines nach dem 1. Januar 2024 abgeschlossenen Vertrags entwickelt werden, gelten nicht als Legacy und unterstehen in jedem Fall der Veröffentlichungspflicht nach Art. 9 Abs. 1 EMBAG.

Die Veröffentlichungspflicht nach EMBAG hängt nicht vom Nutzen für Dritte ab.

3.1.4. Bibliotheken, Plug-ins und Add-ons

In manchen Fällen handelt es sich bei der Software nicht um eine eigenständige Anwendung, sondern nur um Teile davon. Das EMBAG und die Verordnung definieren den Quellcode nicht näher. Die Anleitung gilt auch für entkoppelte Bibliotheken, Plug-ins und Add-ons; diese fallen unter das EMBAG.

Aufgaben:

- Informationen aus Em002-5 Merkblatt OSS-Hilfsmittel beschaffen. Dieses beschreibt sowohl allgemeine Informationen zu Open-Source-Software als auch spezifische Informationen zur Anwendung des EMBAG.
- Prüfen, ob die Voraussetzungen nach EMBAG erfüllt sind.
- Die Em002-2.1 Checkliste OSS-Vorabklärung ausfüllen. In der Regel ist es sinnvoll, die Ansprechpersonen des Softwarelieferanten bzw. der Entwicklung direkt beizuziehen.
- Bei Bedarf die Em002-4.1 Checkliste OSS-Community ausfüllen.

Entscheid: Muss die Software veröffentlicht werden und wie soll dies geschehen?

3.2. Ausnahme: Rechte Dritter

Auf eine Veröffentlichung ist zu verzichten, wenn dadurch Rechte Dritter verletzt würden. Wurde die Software von Bundesangestellten entwickelt, so stehen die Rechte der Arbeitgeberin zu.[6] In Verträgen über Personalverleih und IT-Dienstleistungen werden die Rechte üblicherweise an den Bund übertragen und sollten geltend gemacht werden.

6. Art. 332 OR i. V. m. Art. 6 Abs. 2 BPG

Wurde oder wird die Software auf der Grundlage der Allgemeinen Geschäftsbedingungen des Bundes (Stand 2024) entwickelt,[7] so stehen die Immaterialgüterrechte dem Auftraggeber zu, sofern vertraglich nichts anderes vereinbart wurde.

Eine Anwendung besteht typischerweise aus zahlreichen Einzelkomponenten und -teilen. Bei manchen Arten wird es problematisch, wenn diese selbst nicht Open Source sind oder nicht als Teil der Anwendung unter einer Open-Source-Lizenz veröffentlicht werden können.

Bibliotheken

Je nach Programmiersprache werden Bibliotheken entweder direkt mit der Anwendung kompiliert, gelinkt oder als Pakete ausgeliefert. Sie bilden einen integralen Bestandteil der Anwendung.

Enthält die Anwendung proprietäre oder lizenzpflichtige Bibliotheken, so wird das Open Sourcing schwieriger. Grundsätzlich kann der Quellcode des übrigen Teils der Anwendung veröffentlicht werden, doch müssten potenzielle Nutzende oder Entwickelnde die betreffende Bibliothek erwerben, bevor die Anwendung nutzbar oder erweiterbar wird.

Liegt das Eigentum an der problematischen Bibliothek beim Softwarelieferanten, so lohnt sich eine rechtliche Abklärung, ob der Bund übertragbare Rechte an diesen Bibliotheken hat.

Wo möglich sollten proprietäre und lizenzpflichtige Bibliotheken bei Eigenentwicklungen vermieden oder vor der Freigabe nach Möglichkeit ersetzt werden. Mit Blick auf das Ziel von Artikel 9 EMBAG sollte möglichst viel von der öffentlichen Verwaltung genutzte Funktionalität veröffentlicht werden. Einzelne Bibliotheken, welche die Veröffentlichung des Codes verhindern, stellen daher technische Schulden dar, die dokumentiert und bei Gelegenheit beseitigt werden sollten.

Datenbanken und andere Datenspeicher

Nutzt die Anwendung proprietäre, lizenzpflichtige Datenspeicher wie Microsoft SQL Server, so steht dies einer Open-Source-Veröffentlichung nicht entgegen.

Es sollte jedoch geprüft werden, ob zusätzlich offene Datenbanken wie PostgreSQL unterstützt werden können. Dies kann Betriebskosten sparen und die Einstiegshürden für potenzielle Nutzende senken.

Applikationsserver und Betriebssysteme

Benötigt die Anwendung lizenzierte Betriebssysteme oder Applikationsserver (z. B. Microsoft Windows Server oder RedHat JBoss), so steht dies einer Open-Source-Veröffentlichung nicht entgegen.

Wir empfehlen, beim Softwareanbieter nachzufragen, ob und gegebenenfalls welche lizenzierten Drittbibliotheken als Komponenten der Anwendung verwendet werden.

Wurde die Softwareerstellung gemeinsam von mehreren Organisationen oder anderen Institutionen beschafft, so liegt das Eigentum in der Regel bei einem zu diesem Zweck gegründeten Verein oder einer anderen juristischen Person.

Weiteres geistiges Eigentum

Zu den Rechten Dritter zählen nicht nur Urheberrechte, sondern auch weiteres geistiges Eigentum (Marken, Patente). Patente verhindern grundsätzlich nicht, dass Software als Open Source veröffentlicht wird, können aber deren Nutzung oder Erweiterung verhindern. Patente sind in der

7. Allgemeine Geschäftsbedingungen des Bundes, insbesondere Ziffer 25.1 – https://www.bkb.admin.ch/bkb/de/home/themen/agb.html#accordion_21321111141713247349255

Schweiz zwar eher unüblich, doch lohnt sich in der Regel eine kurze Nachfrage beim Softwareanbieter, ob bereits eine Prüfung stattgefunden hat.

Erwerb der erforderlichen Rechte

Um die gesetzlichen Vorgaben des EMBAG zu erfüllen, hat die Bundesbehörde als Auftraggeberin die Möglichkeit, sich die Rechte an den Arbeitsergebnissen zu sichern. Diese Voraussetzung für eine Veröffentlichung kann bei der Beschaffung als Kriterium festgelegt werden. Der nachträgliche Erwerb notwendiger Rechte an der Software kann aufwendiger werden.

Will die Bundesbehörde die Software nicht selbst veröffentlichen, so kann sie diese Aufgabe beispielsweise an den Auftragnehmer delegieren oder die Veröffentlichungsrechte entsprechend übertragen. In diesem Fall überträgt sie die notwendigen Rechte (und Pflichten) an den Lieferanten und/oder Dritte und beauftragt sie mit der Veröffentlichung. Damit wird das Recht faktisch unter der Bedingung der Freigabe bzw. des Supports der entsprechenden Software an Dritte übertragen.

Auch hier enthält [KKB-MB] wichtige Hinweise.

Vorgehen

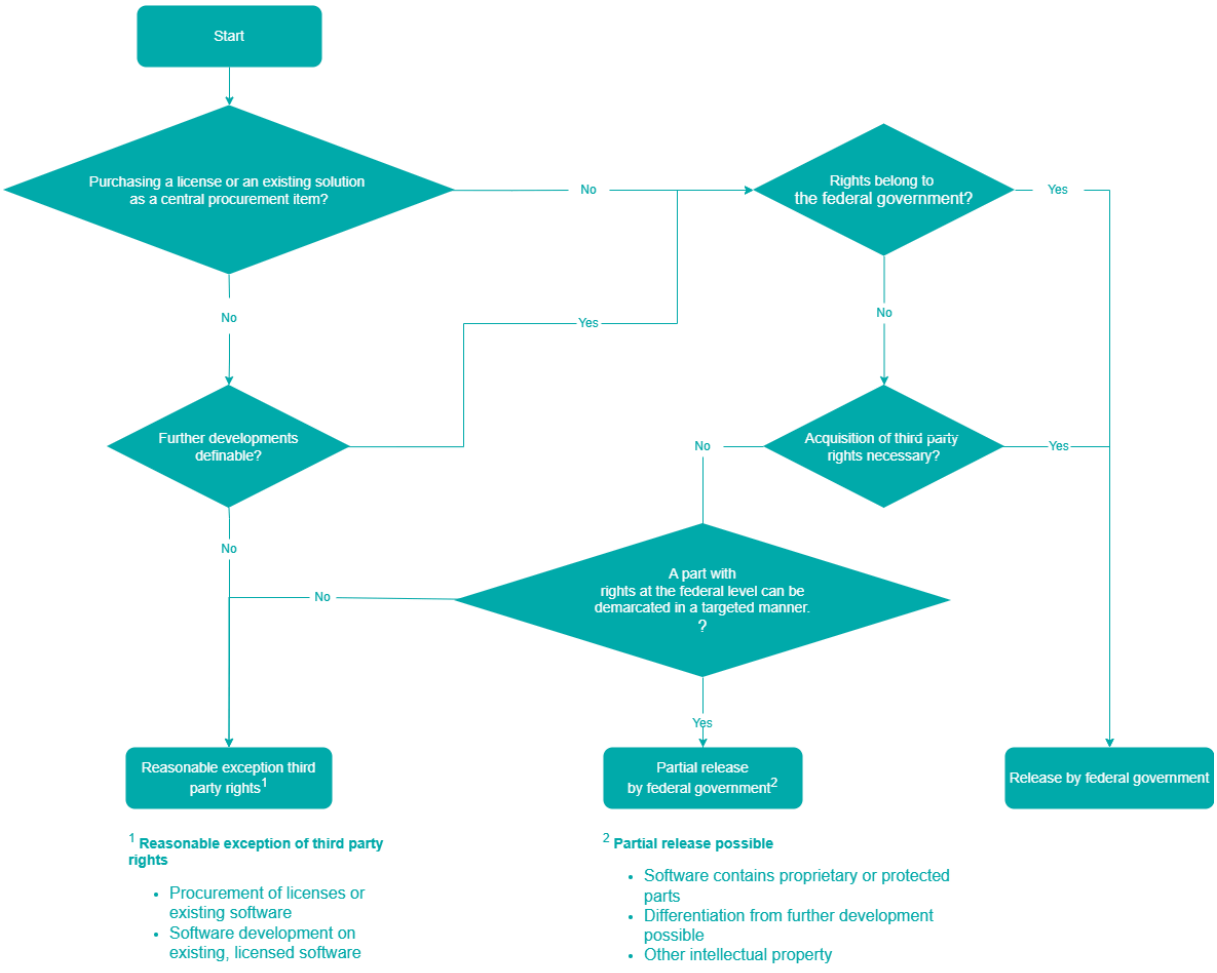


Abbildung 2 – Ausnahme: Rechte Dritter (stets auch [KKB-MB] und Em002-7 Strategische Aspekte der Beschaffung und Open-Source-Software berücksichtigen)

Aufgaben:

- Prüfen, ob die Organisation die massgebenden Schutzrechte an der Software besitzt.

- Prüfen, ob die Rechte erworben werden können.
- Nötigenfalls die schriftliche Erlaubnis der Rechteinhaber einholen.
- Sicherstellen, dass die Anwendung keine proprietären oder geschützten Teile enthält.
- Prüfen, dass keine Hindernisse aufgrund von Patentschutz bestehen (soweit möglich und zumutbar).
- Prüfen, ob die Freigabe durch Dritte erfolgen soll.
- Prüfen, ob die Entwicklung auf Open-Source-Softwareentwicklung beruhen soll (anhand des Em002-4 Leitfadens OSS-Community).

Entscheid: Kann die Software veröffentlicht werden, ohne Rechte Dritter zu verletzen?

3.3. Ausnahme: Sicherheitsrelevante Gründe

Software, die aus sicherheitsrelevanten Gründen nicht veröffentlicht werden kann, ist von der Veröffentlichung ausgenommen.

Die eigentlichen Daten einer Software sind von der Veröffentlichung des Quellcodes nicht betroffen. Diese Daten sind in der Regel sensibel und werden nicht veröffentlicht. Es kann jedoch sein, dass neben den Inhaltsdaten bestimmte im Quellcode sichtbare Algorithmen und Verfahren nicht öffentlich werden sollen (z. B. offensive Cyberfähigkeiten, Einzelheiten der Betrugserkennung). Sicherheit kann sich auf Vertraulichkeit, Integrität, Verfügbarkeit oder Nachvollziehbarkeit beziehen.

Empfehlung:

Software sollte bewusst so entwickelt werden, dass auch bei einer Veröffentlichung des Quellcodes kein zusätzliches Risiko entsteht. Die Annahme, eine Nichtveröffentlichung schütze vor erfolgreichen Angriffen, ist trügerisch («security through obscurity»).

Vorgehen:

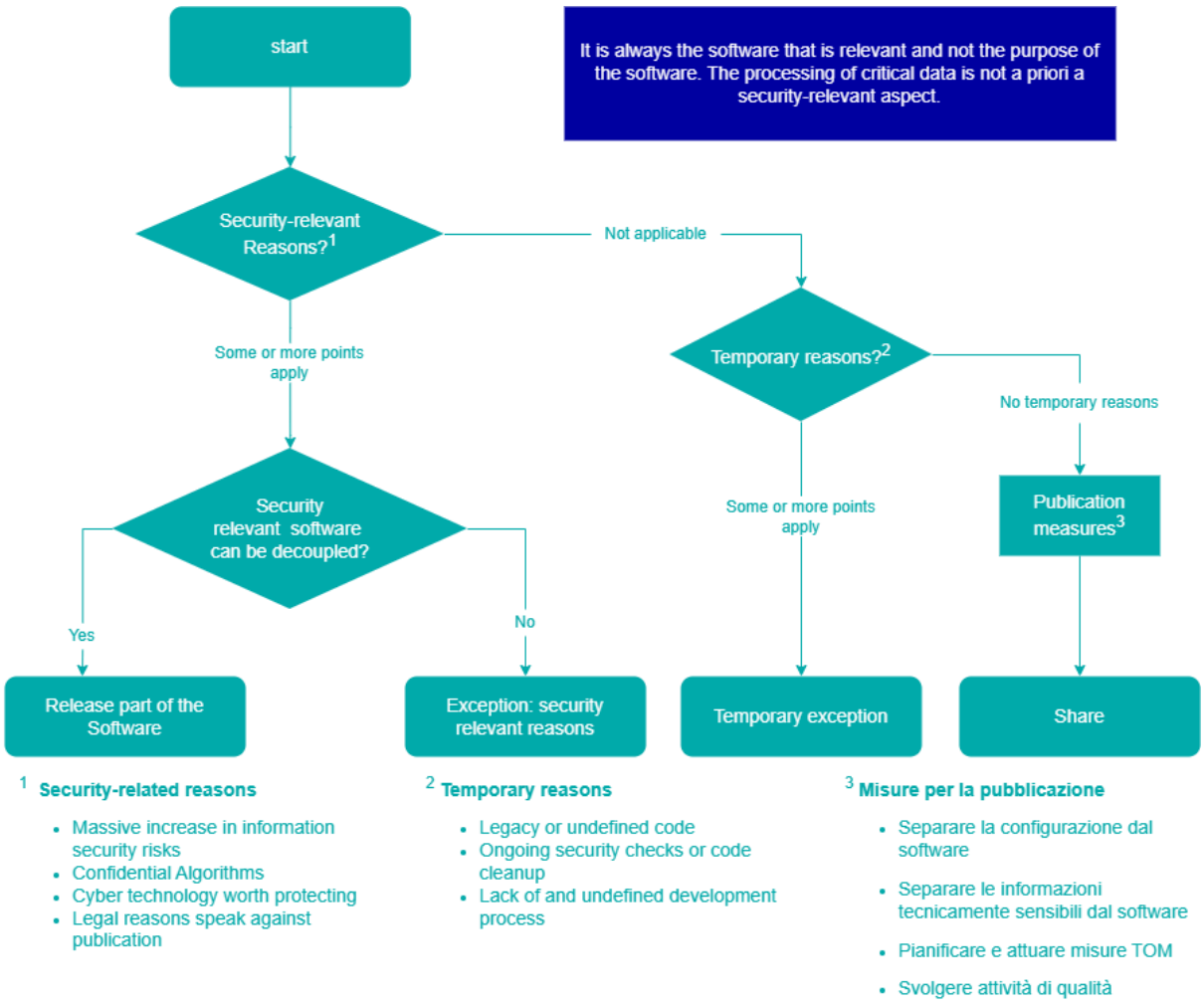


Abbildung 3 – Ausnahme: Sicherheitsrelevante Gründe

Folgende Gründe können je Projekt als sicherheitsrelevant gelten:

- Die Veröffentlichung der Software würde ein Informationssicherheitsrisiko wesentlich erhöhen.[8]
- Bei Algorithmen zur Betrugsprävention und ähnlichen Algorithmen kann gegebenenfalls auf eine Veröffentlichung verzichtet werden.
- Besonders sensible Cybersicherheitstechnologien.
- Kernsoftware für den Betrieb kritischer Infrastrukturen.
- Wenn die Veröffentlichung andere Rechtsgrundlagen verletzen würde.

Die folgenden vorübergehenden Gründe können eine Veröffentlichung noch verhindern. Eine Veröffentlichung wird jedoch später durch geeignete Massnahmen ermöglicht.

- Legacy-Code konnte noch nicht bereinigt werden und enthält möglicherweise sicherheitsrelevante Probleme.
- Sicherheitsaudits des Codes wurden noch nicht durchgeführt.

8. Informationssicherheitsgesetz, ISG. Art. 6 Informationssicherheit. https://www.fedlex.admin.ch/eli/cc/2022/232/de#art_6

- Der Entwicklungsprozess ist noch nicht reif genug für eine sichere Veröffentlichung. Ist der Entwicklungsprozess nicht ausreichend etabliert, so besteht die Möglichkeit, dass Teile veröffentlicht werden, die nicht für Dritte bestimmt sind. Beispielsweise müssen Richtlinien zur Codekommentierung und die Erzeugung von Testdaten an das neue Veröffentlichungsszenario angepasst werden und zuverlässig funktionieren.

Durch folgende Massnahmen kann Software trotz sicherheitsrelevanter Gründe veröffentlicht werden:

- Konfigurationen (z. B. Art der verwendeten Verschlüsselung) von der Software trennen.
- Fachlich sensible, nicht öffentliche Informationen (z. B. Abläufe, Berechnungen) in Konfigurationen oder Einstellungen auslagern.
- Geeignete organisatorische, personelle und technische Massnahmen planen, umsetzen und regelmässig üben.
- Geeignete Qualitätsaktivitäten bei der Veröffentlichung planen und durchführen. Siehe folgendes Kapitel «Analyse und Vorbereitung».

Aufgaben:

- Die Software auf sicherheitsrelevante Gründe prüfen, die einer Veröffentlichung entgegenstehen.
- Den Aufwand für die Behebung vorübergehender Gründe abschätzen.
- Massnahmen planen, um sicherheitsrelevante Informationen vom Quellcode der Software zu trennen.
- Sicherstellen, dass die Anwendung keine Verfahren enthält, die nicht öffentlich werden dürfen.

Entscheid:

Bestehen zwingende sicherheitsrelevante Gründe, die eine Veröffentlichung verunmöglichen?

3.4. Klärung der Veröffentlichung

In diesem Schritt wird geklärt, wer die Software veröffentlicht. Wurde die Software von einem externen Anbieter erstellt, so kann die Veröffentlichung auch diesem Anbieter übertragen werden. In diesem Schritt ist zu bestimmen, wo der Quellcode veröffentlicht wird, und eine entsprechende Governance festzulegen.

Wurde die Software bundesintern entwickelt und kann oder will der Bund sie nicht selbst veröffentlichen, so kann ein Veröffentlichungsauftrag an einen Lieferanten erteilt oder können die Rechte abgetreten werden (Übertragung der Pflichten an Dritte).

Minimale Freigabe und Support

Das EMBAG macht keine Vorgaben zur Veröffentlichung. Eine minimale Freigabe des Quellcodes erfüllt die gesetzliche Vorgabe. Weitere Aktivitäten sind nicht erforderlich. Die Frage, ob und in welchem Umfang Support angeboten wird und wie aktiv das Projekt gepflegt wird, lässt sich anhand des Em002-4 Leitfadens OSS-Community beantworten.

Open-Source-Softwareentwicklung (OSSD)

Bei der Open-Source-Softwareentwicklung (OSSD) wird zusätzlich zur Veröffentlichung des Quellcodes der gesamte Entwicklungsprozess öffentlich durchgeführt. Alles von den Anforderungen (Issues) bis zum Quellcode ist transparent. OSSD wird unterstützt durch öffentliche Kommunikationswerkzeuge (Mailinglisten, Foren usw.), ein Versionsverwaltungssystem (Git), Fehler-

und Funktionslisten, eine Roadmap und Entwicklerwerkzeuge. Durch diese transparente Arbeitsweise profitiert die gesamte Community und die Zusammenarbeit über Organisationsgrenzen hinweg wird ermöglicht. Bei diesem Ansatz ist bei Projektabschluss keine weitere Veröffentlichungsaktivität nötig, da dies bereits zu Projektbeginn berücksichtigt wurde.

Aufgaben

- Eine grobe Aufwandschätzung für die Analysephase erstellen. Je nach Grösse und Komplexität der Anwendung liegt der Aufwand zwischen drei Tagen und zwei Wochen.
- Ressourcen für weitere Schritte, insbesondere für die Analyse, freigeben.
- Grundsätzlich zustimmen, die Anwendung als Open Source zu veröffentlichen.

Entscheid:

Grundsätzliches Einverständnis und Machbarkeit für die Veröffentlichung der Anwendung als Open Source sind gegeben.

3.5. Aufwand minimieren

Die Checklisten und die Anleitung sind so aufgebaut, dass der Aufwand für Freigaben möglichst gering bleibt. Eine konsequente vorausschauende Planung der Freigabe zu Projektbeginn und eine auf die Freigabe ausgerichtete Entwicklung minimieren den Aufwand. Ein gewisser administrativer Aufwand und gewisse Kosten fallen zwangsläufig an [Le2023]. Diese lassen sich unter Umständen über Communities mit geteilten Entwicklungskosten zurückgewinnen (siehe Em002-4 Leitfaden OSS-Community).

3.6. Wahl der Veröffentlichungssprache

Bei der Sprachwahl sind sowohl das potenzielle Zielpublikum als auch die Arbeitsweise des Teams zu berücksichtigen. Erfreulicherweise sind Übersetzungen mit neuen Werkzeugen deutlich einfacher geworden. Als Optionen kommen die Amtssprachen des Bundes[9] und Englisch in Frage.

Ist das Projekt international relevant, so sollte Englisch die Zielsprache sein. Für eine nachträgliche Veröffentlichung ist ein Sprachwechsel naturgemäss wenig sinnvoll.

Mindestens die README.md sollte in mehreren Sprachen vorliegen, damit Interessierte sofort erkennen, ob das Projekt für sie relevant ist.

Die öffentliche Kommunikation sollte in erster Linie das Zielpublikum berücksichtigen. Zu beachten ist zudem, dass es sich um öffentliche Kommunikation des Bundes handelt.

Hinweis: Mit der Veröffentlichung betritt die Bundesverwaltung einen neuen Kanal der öffentlichen Kommunikation. Geschieht dies nicht sorgfältig und professionell, so kann das öffentliche Ansehen des Bundes rasch leiden.

4. Analyse und Vorbereitung

Ziel: Die für eine Open-Source-Veröffentlichung erforderlichen Arbeiten sind abgeschlossen. Die Entscheide zu Lizenz und Art der Community sind gefällt. Dies erfolgt anhand der Em002-2.2 Checkliste OSS-Analyse und -Vorbereitung.

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

4.1. Quellcodeanalyse

Der zu veröffentlichende Quellcode und weitere zugehörige Dokumente werden vor der Veröffentlichung analysiert. Diese Analyse stellt sicher, dass keine vertraulichen Informationen veröffentlicht werden.

Ist umfangreicher Quellcode zur Veröffentlichung vorgesehen, so empfiehlt sich der Einsatz geeigneter automatisierter Werkzeuge für die folgenden Tätigkeiten.

Die Einzelheiten sind in Abschnitt E festgelegt.

Aufgaben

- Prüfen, wie die bestehende Codequalität und die Richtlinien umgesetzt wurden (z. B. ISO/IEC 25010:2023[10]).
- Sicherstellen, dass keine sensiblen (Test-)Daten enthalten sind, z. B. Daten, die auf realen Personen oder Fällen beruhen.
- Die Softwaredokumentation aktualisieren (siehe Anhang).
- Unnötige Dateien oder veraltete Dokumente und Daten löschen.
- Den Quellcode auf Secrets/Zugangsdaten prüfen.[11]
- Gezielte Sicherheitstests durchführen und anschliessend ein (öffentliches) Bug-Bounty-Programm einrichten.[12]
- Alle verwendeten Bibliotheken bezüglich Lizenzen prüfen und die entsprechenden Listen (Attributionen) erstellen. Ist eine Bibliothek unter mehreren Lizenzen verfügbar, so ist dies in den Listen entsprechend anzugeben.
- Das Deployment beschreiben (CI/CD-Pipeline) und allenfalls eine Demo-Instanz einrichten.

Diese Tätigkeiten sind unabhängig von der Veröffentlichung, bilden jedoch Voraussetzungen für eine gute Softwarequalität und die Sicherstellung der Compliance. Die Beachtung dieser Massnahmen vereinfacht die Zusammenarbeit, erhöht die Qualität externer Beiträge und verbessert das Image der Software bzw. der Behörde.

4.2. Lizenzwahl

Siehe Em002-3 Leitfaden OSS-Lizenzierung.

International etablierte Lizenztexte sollten verwendet werden, wo dies möglich und sinnvoll ist. Haftungsansprüche von Lizenznehmern sind so weit rechtlich möglich auszuschliessen.

Aufgaben

- Allfällige Abhängigkeiten von bestehenden Lizenzen prüfen.

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/de/home/aktuell/im-fokus/2022/git.html
12. https://www.ncsc.admin.ch/ncsc/de/home/infos-fuer/infos-it-spezialisten/themen/bug-bounty-programme.html

- Bei Drittcode die Nutzungsbedingungen auf Kompatibilität mit der vorgesehenen Lizenz prüfen und den Code nötigenfalls ersetzen.[13]
- Die geeignete Lizenz auswählen.

Die Lizenzkompatibilität lässt sich mit Werkzeugen wie ORT Toolkit,[14] Black Duck,[15] FOSSA[16] oder FOSSology[17] prüfen. Die ToDo Group [18] bietet eine aktuelle Übersicht über Werkzeuge.

Entscheid:

- Festlegen und dokumentieren, unter welcher Lizenz die Software veröffentlicht wird.

4.3. Open-Source-Dokumentation

Mit der Veröffentlichung wird ein Mindestmass an Dokumentation erwartet. Im Open-Source-Ökosystem haben sich mehrere Dokumente etabliert. Diese geben Besuchenden rasch Einblick in die Software einschliesslich der Lizenzierung.

Folgende Dokumente werden erwartet:

Dokument	Beschreibung / Inhalt:
README(.md)	Dient als Einstiegsdokument und gibt einen raschen Überblick über Zweck, Umfang, Status, Lizenz und Zielgruppen des Projekts.
LICENSE	Beschreibung der im Projekt gewählten Lizenz
CONTRIBUTING(.md)	Beschreibt den Prozess, wie man zum Projekt beitragen kann.
CODE_OF_CONDUCT(.md)	Der Code of Conduct legt die erwarteten sozialen Normen innerhalb des Projekts fest.
CHANGELOG(.md)	Führt eine Liste der Änderungen in den Softwareversionen.
THIRD-PARTY-LICENSES(.md)	Beschreibung der Drittlizenzen bzw. der verwendeten Komponenten.

13. Siehe Em002-3 zur Auswahl von Drittcode bzw. Bibliotheken für einen Technologie-Stack und zur Kompatibilität der Lizenz.
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/

Getting started	Kurzanleitung zur Installation und Nutzung der Software.
Create Gitignore	Beschreibt Dokumente, die nicht veröffentlicht werden, z. B. Konfigurationen, Testdaten oder generierte Inhalte. Geschützte und sensible Informationen sollten ausserhalb des Projekts in geeigneten, dafür vorgesehenen Werkzeugen und Speichern verwaltet werden.
publiccode.yml	Strukturierte Metadaten zur Beschreibung der Software auf Basis des Standards https://yml.publiccode.tools/. Das festgelegte Format macht die Veröffentlichung besser auffindbar und wiederverwendbar. Weitere Informationen in Anhang F.

Tabelle 1 – Open-Source-Dokumente

Weitere technische Einzelheiten zu diesen Dokumenten sind im Anhang beschrieben.

Aufgaben

- Standarddokumentation erstellen.
- Copyright, Lizenzhinweise und Haftungsausschluss in allen Dateien aufführen.
- Je nach Veröffentlichungsplattform können zusätzliche Beschreibungen erstellt werden.

Entscheid:

- Definitiver Entscheid, die Anwendung als Open Source zu veröffentlichen.

5. Veröffentlichung und Bekanntmachung

Ziel: Die Anwendung ist als Open Source veröffentlicht und für Dritte leicht auffindbar. Alle Komponenten sind für die Veröffentlichung bereit und die Community ist aufgebaut oder etabliert. Dies erfolgt anhand der Em002-2.3 Checkliste OSS-Freigabe und -Veröffentlichung. In gewissem Umfang werden dabei bereits gefällte Entscheide rekapituliert.

5.1. Wahl der Plattform

Es gibt mehrere Möglichkeiten, Quellcode zu veröffentlichen. Die folgende Tabelle gibt einen Überblick über mögliche Source-Code-Management-Systeme (SCM) und Plattformen.

Die geeignete Plattform ist vor der eigentlichen Veröffentlichung auszuwählen. Einige Bundesbehörden sind bereits auf GitHub vertreten.[19] In diesem Fall ist es sinnvoll, bereits etablierte Plattformen zu nutzen.

Empfehlung:

Wir empfehlen derzeit, für die Veröffentlichung von Quellcode pro Bundesbehörde eine Organisation auf GitHub einzurichten. Innerhalb dieser Organisation können entsprechende Projekte (Repositories) angelegt und Nutzende mit passenden Berechtigungen verwaltet werden. Künftig sollte eine zentrale Bundesplattform aufgebaut werden. Um eine optimale Benutzerverwaltung zu gewährleisten und das

19. Die Website https://ossbenchmark.com gibt einen Überblick über Organisationen und Behörden, die auf GitHub aktiv sind.

Ansehen der Schweizerischen Eidgenossenschaft zu wahren, ist es wichtig, weniger, dafür besser verwaltete Repositories zu nutzen.

Plattform	Eigenschaften
GitHub[20]	Online-Plattform von Microsoft mit grosser Verbreitung. Eine grosse Menge an Open-Source-Software wird auf GitHub gehostet. Bietet neben SCM viele zusätzliche Werkzeuge. Verfügbar in einer Free- und einer Enterprise-Version (Subskription).
GitLab[21]	Online-Plattform des gleichnamigen Unternehmens GitLab, die selbst unter einer Open-Source-Lizenz veröffentlicht ist. Bietet neben SCM viele zusätzliche Werkzeuge. Kann auch lokal betrieben werden und bietet damit ein gewisses Mass an Souveränität. Verfügbar in einer Free- und einer Enterprise-Version (Subskription).
Bitbucket[22]	Online-Plattform von Atlassian, speziell in dessen Ökosystem integriert. Verfügbar in einer Free- und einer Enterprise-Version (Subskription).
Internes SCM-System	Wird eine eigene SCM-Plattform bereitgestellt, so kann diese ebenfalls veröffentlichen und bestimmte Repos/Projekte verfügbar machen. Der Betrieb muss sichergestellt sein.
Bundesplattform	Derzeit gibt es keine zentrale Bundesplattform für die Veröffentlichung von Software. Eine entsprechende Massnahme wird in Em002 Strategische Leitlinien für Open-Source-Software in der Bundesverwaltung vorgeschlagen.

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

Website, öffentlicher FTP-Server usw.	Das EMBAG legt nicht fest, wie Software zu veröffentlichen ist. Eine minimale Freigabe[23] kann auch auf einer Website oder anderen öffentlich zugänglichen Ressourcen erfolgen. Für eine nachhaltige Zusammenarbeit im Sinne von Open Source ist dies jedoch nicht geeignet.

Tabelle 2 – Plattformen für die Veröffentlichung von Open-Source-Software

Wird die Software gemeinsam mit einem externen Partner oder einer externen Organisation veröffentlicht, so ist darauf zu achten, dass entsprechende Berechtigungen und Zugriffe auf den Quellcode vorhanden sind.

Aufgaben

- Geeignete Plattform einschliesslich entsprechender Organisation/Projekt evaluieren.
- Sofern noch nicht geklärt, die Benennung von Projekt/Repository festlegen.
- Berechtigungen am Repo festlegen und prüfen.
- Angemessene Governance und eine allfällige Replikation des Quellcodes sicherstellen.

Entscheid:

- Die Plattform für die Veröffentlichung wurde sachgerecht gewählt.

5.2. Veröffentlichung der Software

Sind die Voraussetzungen geklärt, so kann die Software veröffentlicht werden.

Aufgaben

- Die Voraussetzungen anhand der Em002-2.3 Checkliste OSS-Freigabe und -Veröffentlichung prüfen.
- Der Zugang zur Plattform besteht und die Governance ist geklärt.
- Veröffentlichung von Quellcode und Dokumentation durch die Entwicklung.

5.3. Kommunikation

Nach der eigentlichen Veröffentlichung können durch und innerhalb der betroffenen Bundesbehörde weitere Kommunikationsaktivitäten angestossen werden. Diese unterstützen die Verbreitung und bringen den gewünschten Mehrwert aus der Veröffentlichung.

Aufgaben

- Beschreibung im Repo-Hosting, Sicherheitseinstellungen usw.
- Interne und gegebenenfalls externe Kommunikation
- Eine Mitteilung an die Community verfassen.

23. Eine minimale Veröffentlichung umfasst nur die Veröffentlichung des Quellcodes und weiterer minimaler Artefakte. Diese Art der Veröffentlichung erfüllt die Vorgaben des EMBAG. Ein Mehrwert wird durch die Veröffentlichung jedoch nicht erzielt.

- Die Anwendung innerhalb der Organisation und in Fachgremien bekannt machen, deren Nutzung und Beiträge fördern.
- Eine publiccode.yml-Datei[24] im Repository für den OSS-Katalog der Bundesverwaltung erstellen.[25]

5.4. Aufbau und Pflege der Community

Bei der Veröffentlichung von Software kann eine Community entstehen, die durch Beiträge (Pull Requests), Fragen, Dokumentationsvorschläge usw. zur Software beitragen kann. Für eine aktive Community ist erheblicher Aufwand sowohl für den Aufbau als auch für die laufende Pflege erforderlich. Eine gut funktionierende Community minimiert unerwünschte Forks der Software. Ziel ist es, den Mehrwert der Community durch gezielte Aktivitäten zu aktivieren. Auch bei einer minimalen Veröffentlichung sollte der Umgang mit Rückmeldungen und Fragen zur Software festgelegt werden.

Aufgaben

- Klären, wer potenzielle Nutzende oder Interessierte sind.
- Die geeignete Form für die konkrete Anwendung anhand des Em002-4 Leitfadens OSS-Community bestimmen.
- Die Community aufbauen.
- Aktivitäten zum Community-Aufbau.

6. Anhang

6.1. Änderungen gegenüber der Vorversion

- Kapitel 4.3 und 5.3: Veröffentlichung mit publiccode.yml ergänzt
- Anhang F «Public Code» wurde ergänzt
- Verschiedene kleinere redaktionelle Änderungen und Verbesserungen

6.2. Referenzen

Siehe *Em002 Strategische Leitlinien für Open-Source-Software in der Bundesverwaltung*.

6.3. Abkürzungen

Siehe Em002 Strategische Leitlinien für Open-Source-Software in der Bundesverwaltung und Em002-6 FAQ zu OSS.

6.4. Begleitdokumentation Em002-2.2 Checkliste OSS-Analyse und -Vorbereitung – Dokumentation

6.5. Quellcodedokumentation

Die Dokumentation sollte Teil des Quellcodes des Projekts sein, damit Änderungen an der Dokumentation revisionssicher abgelegt werden und die Dokumentation parallel zum Projekt versioniert wird. Damit ist automatisch sichergestellt, dass die Dokumentation bei sachgerechter

24. Der Metadatenstandard https://yml.publiccode.tools/ beschreibt die Software. Das festgelegte Format macht die Veröffentlichung besser auffindbar und wiederverwendbar. Weitere Informationen in Anhang F.
25. OSS-Katalog: https://www.opensource.admin.ch

Pflege nicht vom Entwicklungsstand des Projekts abweicht und dass die Dokumentation zu älteren Versionen des Projekts jederzeit abgerufen werden kann.

Zudem erhalten Dritte damit die Möglichkeit, auf einfache Weise Änderungen an der Dokumentation beizutragen.

Als Auszeichnungssprache für die Textformatierung sollte Markdown verwendet werden, da es einfach zu schreiben, weit verbreitet und maschinenlesbar ist. Zudem werden Markdown-Dokumente von allen gängigen Plattformen in einem gut lesbaren Zielformat dargestellt. Markdown-Dokumente lassen sich unabhängig von der eingesetzten Plattform mit statischen Website-Generatoren wie beispielsweise Jekyll, MkDocs, Gatsby oder Sphinx in jedes Zielformat rendern, etwa mit den Corporate-Identity-Vorgaben einer Verwaltung.

6.6. Adressaten und Aufbau der Dokumentation

Die Projektdokumentation sollte sowohl Endnutzende als auch technische Fachpersonen ansprechen und sich somit nicht ausschliesslich auf die Technik konzentrieren. Zur Sprachwahl siehe Kapitel 3.6. Dies unterstützt die Verbreitung der Software.

Eine Datei README.md dient als Einstiegsdokument. Dateien mit diesem Namen werden von allen gängigen Plattformen erkannt und als Einstiegspunkt in die Dokumentation angezeigt.[26]

README.md sollte einen raschen Überblick über Zweck, Umfang, Status, Lizenz und Zielgruppen des Projekts geben. Konkret sollte README.md die folgenden Punkte enthalten:

- Name der Software
- Kurzbeschreibung von Zweck und Funktion der Software. Umfang und mögliche Einschränkungen
- Installationsanleitung
- Nutzung der Software
- Hinweis auf eine allfällige Demo-Instanz
- Support- und Kontaktangaben
- Wie man zur Open-Source-Software beiträgt (Contributing)
- Verwendete Lizenz (Licence)
- Projektstatus / Entwicklungsstand

6.7. Open Source hervorheben und Lizenz ablegen

Direkt nach dem Leitbild sollte klar und unmissverständlich festgehalten werden, dass es sich beim Projekt um Open-Source-Software handelt. Die vom Projekt verwendete Lizenz sollte mit dem entsprechenden SPDX-Identifikator[27] angegeben und mit dem konkreten Lizenztext in der Datei LICENSE verlinkt werden. Lizenzvorlagen enthalten typischerweise einen variablen Teil (z. B. Projektname, Copyright-Jahr, Name der Autorin oder des Autors), der in der Datei LICENSE anzupassen ist.

Die meisten Plattformen erkennen Lizenzen in einer Datei LICENSE und bieten übersichtliche Informationen zur Lizenz, z. B. eine kurze Zusammenfassung der Lizenzbedingungen.

26. Weitere Informationen zum Erstellen einer Readme-Datei: https://www.makeareadme.com/
27. SPDX-Identifikator – https://spdx.org/about

Es wird empfohlen, jeder neu erstellten Quelldatei einen minimalen Lizenz- und Copyright-Header hinzuzufügen. Bestehende Header in bestehenden Dateien sollten nicht verändert werden. Wird zu einem Projekt beigetragen, das klare Vorgaben für Header hat, so sind diese einzuhalten.

Weitere Einzelheiten siehe: Producing Open Source Software: State That the Project is Free.[28]

6.8. Aktueller Entwicklungsstand

Um den Zustand des Projekts rasch aufzuzeigen, sollte der aktuelle Entwicklungsstand in README.md dokumentiert werden. Für Lesende ist es wichtig zu wissen, ob sich das Projekt in einem reifen Zustand oder am Anfang der Entwicklung befindet, ob es aktiv gepflegt wird und wie häufig neue Versionen veröffentlicht werden.

In diesem Abschnitt kann beschrieben werden, welche Art von Unterstützung durch Dritte für das Projekt derzeit am wichtigsten ist. Dies könnte beispielsweise eine Entwicklerin oder ein Entwickler mit spezifischem Technologiewissen sein oder jemand, der die Projektdokumentation überarbeitet.

Weitere Einzelheiten siehe Producing Open Source Software: Development Status.[29]

6.9. Demo-Instanz

Bei Anwendungen empfiehlt es sich, eine Demo-Instanz bereitzustellen, damit die Anwendung ohne Installationsaufwand ausprobiert werden kann. Je nach Art der Anwendung kann die Demo-Instanz die produktive Installation oder eine Teststufe sein. Wichtig ist, dass Lesende möglichst eigenständig und direkt auf die entsprechende Instanz zugreifen können.

Als Alternative stellen viele Open-Source-Projekte Werkzeuge bereit, um die Anwendung einfach zu starten. Dies kann beispielsweise mit Containern, portablen Anwendungen oder Installationsskripten erfolgen.

6.10. Fachliche Ansprechperson und Projekteignerschaft

In der Datei README.md sollte die wichtigste fachliche Ansprechperson genannt werden, damit andere interessierte Verwaltungen und Organisationen möglichst direkt Kontakt aufnehmen können.

Persönliche E-Mail-Adressen sollten nicht verwendet werden.

In der Datei README.md sollte zudem kurz beschrieben werden, welche Organisation Projekteignerin ist, insbesondere wenn für das Projekt ein Verein gegründet wurde.

7. Installationsdokumentation

Bei Anwendungen sollte README.md auf eine Installationsdokumentation verweisen, die Hardware- und Softwareanforderungen beschreibt, welche Infrastrukturkomponenten (beispielsweise Datenbanken) verwendet werden und wie die Anwendung installiert und betrieben werden kann.

7.1. Developer Guidelines

Der Einstiegspunkt für die Developer Guidelines sollte in einer Markdown-Datei CONTRIBUTING.md abgelegt werden, die in README.md verlinkt ist. Die Developer Guidelines sind Teil der erweiterten Dokumentation für Entwickelnde, die zum Projekt beitragen möchten, und konzentrieren sich in erster Linie auf die Zusammenarbeit und Kommunikation im Projekt und nicht auf die Technik.

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

Die Developer Guidelines sollten die folgenden Punkte enthalten und gegebenenfalls auf die entsprechenden Dokumente verweisen:

- Einen Verweis auf den Code of Conduct des Projekts. Der Code of Conduct legt die erwarteten sozialen Normen innerhalb des Projekts fest.
- Es wird empfohlen, als Code of Conduct den unter CC-BY-4.0[30] lizenzierten Contributor Covenant Code of Conduct zu wählen oder darauf aufzubauen.[31]
- Einen Hinweis, dass das Projekt Code-Reviews in Form von GitHub-Pull-Requests durchführt, mit einem Verweis auf die GitHub-Pull-Request-Dokumentation.

Abschliessend sollte auf die Developer Documentation verwiesen werden.

7.2. Developer Documentation

Während die Developer Guidelines die Zusammenarbeit und die sozialen Normen des Projekts beschreiben, konzentriert sich die Developer Documentation auf die technischen Aspekte der Mitwirkung an der Entwicklung des Projekts. Sie sollte mindestens die folgenden Punkte beschreiben:

- Welche technischen Abhängigkeiten das Projekt hat und welche Werkzeuge für die Projektentwicklung nötig sind.
- Welche formalen Regeln für den Quellcode des Projekts gelten, beispielsweise zur Quellcodeformatierung.
- Wie der Quellcode des Projekts organisiert ist und wie Funktionen technisch getestet werden können.
- Eine Architekturskizze mit einer groben Beschreibung der Architektur, vorzugsweise einschliesslich Kontextabgrenzung und grober Komponentensicht.[32]
- Bei Anwendungen: Wie die Anwendung zu Entwicklungs- oder Debugging-Zwecken gestartet werden kann und welche Infrastrukturkomponenten hierfür nötig sind.

8. Begleitdokumentation zur Em002-2.2 Checkliste OSS-Analyse und -Vorbereitung – Quellcode

8.1. Quellcodehistorie

Der Quellcode eines Projekts wird üblicherweise in einem Source-Code-Management-System (SCM) wie GitHub abgelegt. Diese Systeme speichern nicht nur den aktuellen Stand des Quellcodes, sondern auch Änderungen, die im Lauf der Zeit am Quellcode vorgenommen wurden. Dazu gehören insbesondere Löschungen im Quellcode einschliesslich gelöschter Dateien.

Zudem enthält das SCM Metainformationen zu Änderungen am Quellcode, etwa Name und E-Mail der Autorin oder des Autors oder PGP-Signaturen für signierte Commits und Tags.

8.2. Sensible, geschützte oder vertrauliche Daten

Der Quellcode und die Quellcodehistorie des Projekts können öffentlich verfügbare Daten enthalten, etwa ein Postleitzahlenverzeichnis, zufällig erzeugte Daten oder synthetische Daten.

30. https://creativecommons.org/licenses/by/4.0/
31. https://www.contributor-covenant.org/version/1/4/code-of-conduct.html
32. Architekturrahmen MMB (Modellierungsmethode Bund) oder arc42. http://arc42.org/

Der Quellcode und die Quellcodehistorie dürfen keine sensiblen oder vertraulichen (Personen-)Daten oder Informationen im Sinne des Datenschutzgesetzes (DSG) und des Informationssicherheitsgesetzes (ISG) enthalten. Dazu gehören insbesondere:

- Benutzernamen und Passwörter oder andere Geheimnisse wie private Schlüssel, Zugangstoken oder Zertifikate.
- Interne DNS-Namen, URLs, Hostnamen, IP-Adressen, Netzstrukturen oder Laufwerksnamen.
- Namen, Adressen, Bilder oder ähnliche Personendaten ohne Einwilligung der betroffenen Person
- Anonymisierte oder pseudonymisierte Daten (weil Anonymisierung oder Pseudonymisierung rückgängig gemacht werden können)

Auch wenn der aktuelle Stand des Quellcodes frei von sensiblen, geschützten oder vertraulichen Daten oder Informationen ist, können solche Daten oder Informationen weiterhin aus der Historie des SCM-Systems abgerufen werden, wenn sie in der Vergangenheit je Teil des Quellcodes oder der SCM-Metainformationen waren. Eine Bereinigung des aktuellen Quellcodestands entfernt diese Informationen nicht vollständig.[33]

Im Zweifelsfall wird daher empfohlen, die Quellhistorie nach der Bereinigung des Quellcodes vor dessen Veröffentlichung vollständig zu entfernen.

Namen und E-Mail-Adressen der Autorinnen und Autoren des Quellcodes sind üblicherweise direkt im Quellcode oder in den SCM-Metadaten verfügbar, beispielsweise in GitHub-Commits oder annotierten GitHub-Tags. Rechtlich ist dies unproblematisch, da das entwickelnde Unternehmen die Veröffentlichung solcher Informationen zu regeln hat. Es wird jedoch empfohlen, das entwickelnde Unternehmen bzw. bundesinterne Mitarbeitende darauf hinzuweisen, dass Namen und E-Mail-Adressen der Quellcode-Autorinnen und -Autoren bei der Veröffentlichung der Software offengelegt werden können.

8.3. Entfernen von Autorenangaben

Die Historie der Software kann geändert werden, um Autorenangaben zu verbergen. Werkzeuge[34] können diesen Schritt weitgehend automatisieren.

Das Entfernen von Autorenangaben beseitigt Attribution, Nachvollziehbarkeit und Kontaktinformationen. Der damit verbundene Aufwand ist je nach Umfang des Quellcodes nicht zu unterschätzen. Aus diesen Gründen wird das Entfernen von Autorenangaben nicht empfohlen. Namen dürfen veröffentlicht werden, wenn a) die Einwilligung der betroffenen Personen vorliegt und b) keine sicherheitsrelevanten Gründe dagegensprechen.

8.4. Berücksichtigung von Bibliothekslizenzen

Praktisch jedes Projekt nutzt Drittbibliotheken (Softwarebibliotheken, Abhängigkeiten), um auf häufig verwendete Funktionen zuzugreifen, ohne sie selbst programmieren zu müssen. Diese Bibliotheken hängen üblicherweise von weiteren Bibliotheken ab, woraus mehrstufige Abhängigkeiten entstehen (transitive Abhängigkeiten). Je nach Programmiersprache können selbst kleine Projekte Hunderte von Abhängigkeiten aufweisen.

Die Ergebnisse dieser Abklärung sind bei der Lizenzwahl nach Em002-3 Leitfaden OSS-Lizenzierung und in der Em002-2.3 Checkliste OSS-Freigabe und -Veröffentlichung zu berücksichtigen.

33. Eine Möglichkeit zur Bereinigung eines Repos bietet das Werkzeug BFG Repo-Cleaner
34. https://www.adamdehaven.com/blog/update-commit-history-author-information-for-git-repository/

Für jede dieser Abhängigkeiten ist sicherzustellen, dass ihre Lizenz mit der Projektlizenz kompatibel ist. Besonders problematisch sind Abhängigkeiten, die nicht unter einer Open-Source-Lizenz stehen, die keine Lizenz deklariert haben oder die eine virale Lizenz verwenden, die nicht zur Projektlizenz passt. Manche Bibliotheken sind unter mehr als einer Lizenz lizenziert und erlauben es den Nutzenden, eine auszuwählen.

- In allen wichtigen Programmiersprachen lassen sich die Abhängigkeiten des Projekts (einschliesslich transitiver Abhängigkeiten) und deren Lizenzen automatisch auflisten. Dazu gehören:
- Das Project-Info-Reports-Plugin für den Paketmanager Apache Maven für Java-Projekte, das mit dem Dependency-Report eine HTML-Ausgabe aller verwendeten Bibliotheken einschliesslich der deklarierten Lizenz erzeugt.[35]
- Der NPM Licence Checker für den Paketmanager NPM für JavaScript-Projekte, der alle verwendeten Bibliotheken einschliesslich der deklarierten Lizenz in verschiedenen Formaten ausgeben kann, z. B. CSV.[36]
- Der Licence Finder von Pivotal, der verschiedene Paketmanager und Programmiersprachen unterstützt, darunter Ruby Gems, Python Eggs und Godeps.[37]

9. Verwendung von Bibliotheken mit proprietären Lizenzen

Bibliotheken unter einer proprietären Lizenz sind in Open-Source-Projekten grundsätzlich problematisch und können in der Regel nicht verwendet werden, insbesondere wenn der Quellcode unter einer viralen Lizenz wie der AGPL veröffentlicht wurde.

Bitte wenden Sie sich an den Anbieter und den Rechtsdienst des betreffenden Amtes, um zu prüfen, ob und unter welchen Bedingungen die proprietäre Bibliothek in Ihrem Projekt verwendet werden kann.

9.1. Verwendung von Paketmanagern

Praktisch alle verbreiteten Programmiersprachen stellen Paketmanager zur Verwaltung ihrer Abhängigkeiten bereit, etwa Apache Maven für Java-Projekte, NPM für JavaScript-Projekte oder NuGet für .NET-Projekte.

Bei Verwendung eines Paketmanagers sind die vom Projekt genutzten Abhängigkeiten nicht mehr direkt Teil des Quellcodes des Projekts, etwa durch Kopieren des Quellcodes der Abhängigkeit in das Projekt oder durch Kopieren der Abhängigkeiten in binärer Form in das Projekt. Stattdessen bezieht das Projekt alle Abhängigkeiten über die Deklarationen des Paketmanagers.

Dies erlaubt es, alle Abhängigkeiten aufzulisten und korrekt zu referenzieren. Zudem verhindert es die unbeabsichtigte Veränderung von Abhängigkeitscode. Sind Änderungen nötig, so müssen sie direkt an der Quelle der Abhängigkeit vorgenommen werden.

9.2. Deklaration der Projektlizenz im Paketmanager-Deskriptor

Das Projekt sollte seine Lizenz im entsprechenden Paketmanager-Deskriptor deklarieren, unter Verwendung des SPDX-Identifikators,[38] damit die Projektlizenz vom Paketmanager ausgegeben

35. Plugin-Info 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 von Pivotal https://github.com/pivotal/LicenseFinder
38. SPDX-Identifikator – https://spdx.org/

werden kann. Dies ist besonders bei Bibliotheken wichtig, damit Nutzende der Bibliothek deren Lizenz automatisch über den Paketmanager abfragen können.

9.3. Attribution und Copyright-Hinweise

Viele Bibliotheken unterstehen einer Lizenz, die von den Nutzenden der Bibliothek einen ursprünglichen Copyright-Hinweis oder eine Attribution verlangt.

Dazu gehören die folgenden Lizenzen (Auszug):

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

Für weitere siehe die Liste der von der OSI anerkannten Lizenzen.[39]

In der Praxis betrifft dies nahezu alle verwendeten Bibliotheken. Da die Zahl der in kleinen Projekten verwendeten Bibliotheken bereits sehr hoch ist, ist ein rechtlich korrekter Copyright-Hinweis bzw. die nötige Attribution nur mit grossem Aufwand möglich.

Es wird daher empfohlen, wie folgt vorzugehen:

- Eine Liste der verwendeten Bibliotheken erstellen, mit den Mechanismen des Paketmanagers oder mit Werkzeugen wie dem NPM Licence Checker oder dem Licence Finder von Pivotal.
- Die Liste manuell aufbereiten, sodass sie die folgenden Angaben enthält: offizieller Name des Bibliotheksprojekts, Website des Bibliotheksprojekts, für die Bibliothek verwendete Lizenz als SPDX-Identifikator mit Link auf den Lizenztext.
- Die Liste im Quellcode des Projekts ablegen, z. B. in einer Datei THIRD-PARTY-LICENSES.md.
- Bei Anwendungen: Einen Link auf THIRD-PARTY-LICENSES.md in einem Dialog «Über» oder «Über diese Anwendung» aufnehmen.
- Der Dialog «Über» ist für diesen Zweck weit verbreitet, beispielsweise in Google Chrome (im Dialog «Über Chrome»).
- Bei Bibliotheken: In README.md unter THIRD-PARTY-LICENSES.md verweisen.
- Im Rahmen des Code-Review-Prozesses sicherstellen, dass Änderungen an den Bibliotheken in THIRD-PARTY-LICENSES.md nachgeführt werden.

10. Beispiel Codeblock THIRD-PARTY-LICENSES.md

Dieses Projekt verwendet Open-Source-Software:

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

10.1. Begleitdokumentation zur Em002-2.3 Checkliste OSS-Freigabe und -Veröffentlichung

39. Open-Source-Lizenzen sind Lizenzen, die der Open Source Definition entsprechen – kurz gesagt erlauben sie, Software frei zu nutzen, zu verändern und zu teilen. https://opensource.org/licenses

10.1.1. Veröffentlichung mit publiccode.yml

Publiccode.yml ist ein Metadatenstandard für Software-Repositories, die von öffentlichen Verwaltungen entwickelte oder beschaffte Software enthalten. Ziel ist es, diese Software besser auffindbar und wiederverwendbar zu machen. Der Metadatenstandard ist international etabliert und öffentlich unter https://yml.publiccode.tools/ beschrieben. Eine Datei publiccode.yml enthält typischerweise Angaben wie Titel und Beschreibung (in mehreren Sprachen möglich), Entwicklungsstand, Kontaktangaben, Lizenzinformationen usw. Für die Schweiz sind länderspezifische Erweiterungen geplant.

Für jede Software ist im Hauptverzeichnis des Repositorys eine yml-Datei zu erstellen. Besteht die Lösung aus mehreren (Hilfs-)Repos, so genügt eine Datei, da diese den Anwendungsfall der Lösung beschreibt.

Anhand der strukturierten Daten in den publiccode.yml-Dateien wird für die gesamte Bundesverwaltung ein OSS-Katalog (opensource.admin.ch) erzeugt. Der Katalog bietet einen Überblick über veröffentlichte Software und erleichtert das Auffinden und die Wiederverwendung bestehender Lösungen.

Wird ein neues Repository oder eine neue Organisation erstellt, so ist dies für die Aufnahme in den OSS-Katalog per E-Mail an opensource@bk.admin.ch zu melden.

Neben dem OSS-Katalog steht auch ein grafischer Editor und Validator für publiccode.yml-Dateien zur Verfügung.

OSS-Katalog und Editor: https://www.opensource.admin.ch/

10.2. Weitere Quellen und Lizenz

Die Checkliste, die Begleitdokumentation und zugehörige Vorlagen beruhen teilweise auf den folgenden Quellen:

- Google Open Source Docs von Google LLC, lizenziert unter CC-BY-4.0[40]
- Producing Open Source Software, How to Run a Successful Free Software Project von 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/