FOSSEPS Pilot Project

Identifying critical open source software used by European Public Services

Context and Survey Guidelines

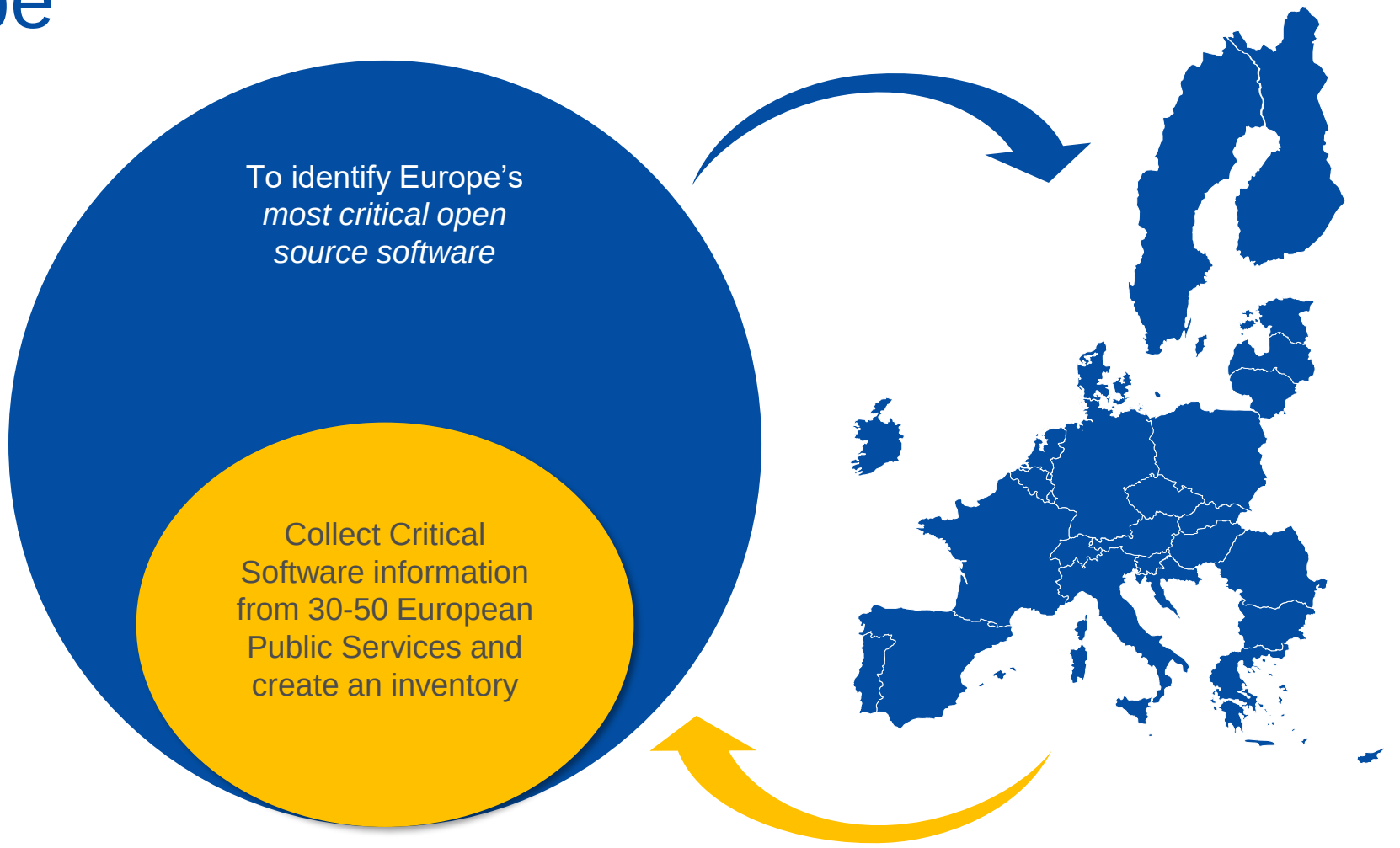
02 February 2022

The FOSSEPS Pilot Project
OSPO, DIGIT B3
EUROPEAN COMMISSION

DIGIT-OSPO@ec.europa.eu

FOSSEPS (Free Open Source Software for European Public Services) is the project name adopted for the Pilot project: Europe-wide solutions for free and open source software use by public services in the EU (budget line PP 01 21 04)

Scope



European Commission

What exactly is critical software?

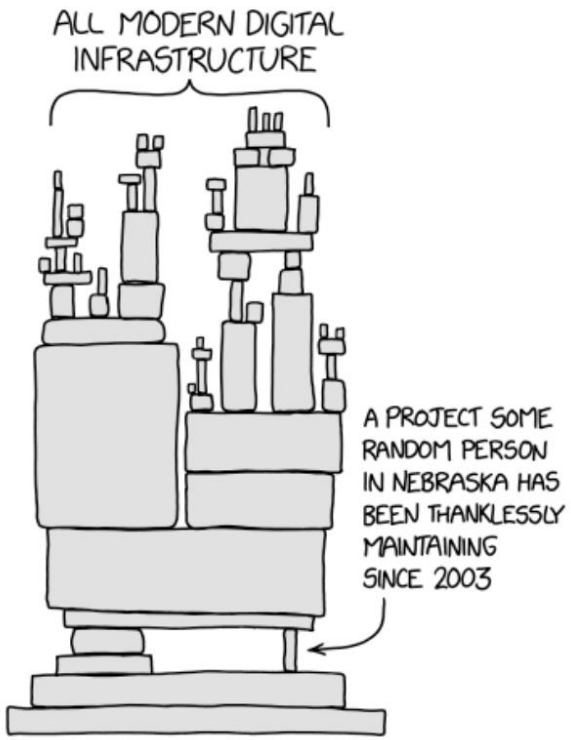
In this project "Critical software" is software that is significantly important for European Public services (EPS) and *whose continued usage and existence is at risk*.

The importance of a software item can be due to it being widely used inside an organisation or because it supports key processes for this organisation.

From a usage perspective: the software may not be well supported via internal support contracts or via inadequate/poor responses from the software community that maintains it.

From a sustainability perspective: the software's continued existence may be threatened due to the poor health of the software project community and their inability to maintain it.

Help us protect the open source software Europe runs on



Credit: Image courtesy of XKCD.com

All European Public Services use open source software to some extent. But do we know our most critical components?

Sadly, the answer is NO… many public administrations do not have an inventory of their open source, and so cannot easily identify their critical software!

This FOSSEPS Pilot project aims to fix this, and build an initial inventory of critical software.

That will help to establish which software is ***most critical*** to European Public Services → so that we can collectively secure it, sustain it and protect it.

We can't do this without you, and we encourage you to participate by completing a survey. The next few pages will explain how you can do that. If you need help, please **contact the following people**:

1. Inno^3: Camille MOULIN (cmoulin@inno3.fr)
2. FOSSEPS Project, DIGIT, EC: Saranjit ARORA (DIGIT-OSPO@ec.europa.eu)
3. OSPO, DIGIT, EC: Miguel DIEZ-BLANCO (DIGIT-OSPO@ec.europa.eu)

Note: Inno^3 and Deloitte are subcontractors on this project.

European Commission

Our proposed steps to find critical software

European Public Services

1. Answer FOSSEPS Project Survey

- Also contains questions about open source management

2a. EPS Analyses and Extracts Data

- Identify critical open source software
- Identify critical software dependencies

2b. EPS Uploads spreadsheets to survey

European Commission

3. Analyse received data files

- Consolidate data
- Identify recursive dependencies

4. Combine with already known critical software

- We already have a list of widely known critical software

5. Speak with critical software projects and to confirm problems and explore solutions

6. Publish an inventory of Europe's most critical software

7. Publish proposed solutions to fix issues that cause software criticality

How you can help

1. Answer Survey questions
2. Identify your critical software and software dependencies
3. Prepare and upload your two spreadsheets

Getting started

Survey questions	We have purposely designed a short survey, and added help text to make the questions self-explanatory.
Identifying my critical software	This is the primary objective. Right now, European Public Services do not know their critical software. So your answers will help us create a Europe-wide picture.
Simplest way	Get a group of people round a table, brainstorm, and write some potential candidates, and fill in the spreadsheets.
Small effort	Getting a list of key software your organisation uses, look for the most used, and compare support contracts. What's missing? Ask your developers what they think is really crucial for your organisations. Ask, if this software vanishes, can we survive?
Medium effort	Look at each data source – applications, infrastructure, cloud, virtual machines. Try and create extracts, and do some number crunching. Follow the guidelines below.
Large effort	Create a complete open source software inventory and follow the steps suggested below. Look at what the European Commission did in its own inventory process. Download the recently improved inventory methodology.
Filling in the spreadsheets	Fill in the spreadsheets as best you can. If in doubt, email us and we can set up a call.

Handling Survey Questions

1. Use the help text accompanying the survey questions
2. See if the definitions on the next page help
3. Email us for help

Key Definitions

Dependencies, libraries, frameworks	In the context of software development, “dependencies” is a generic term to describe third party code on which you rely to create your own application. They can be libraries (easily reusable pieces of code) or frameworks (consistent sets of libraries embodying a methodology).
Recursive, first level dependencies	The dependencies that you use can themselves have dependencies, thus creating a tree of recursive dependencies. Modern development tools allow to automatically download the transitive dependencies of the “first level dependencies” (i.e. the ones that you declare explicitly).
Infrastructure	Infrastructure is a generic term to designate applications that serve as a base for other (business) applications to work correctly: this includes application servers (like Apache Tomcat), databases server (like PostgreSQL), directories (like OpenLDAP), or operating systems (like Debian GNU/Linux).
Dual licensed, Open Core software	A software is “dual licensed” when it is made available under two different licenses, one open source and one proprietary, but they cover the same product (e.g. MySQL). Open Core software is available in two versions, one open source (often called “Community Edition”) and one proprietary with additional features (e.g. Gitlab).
Open source, proprietary software	When we mention “open source” we mean software licensed according to the Open Source Definition (https://opensource.org/osd) and to the Free Software Definition (https://www.gnu.org/philosophy/free-sw.html). “Proprietary” means non open source software.

Identifying Critical Software

1. Study the slides/pages below
2. Talk to people in your organisation
3. Email us for help and we can set up a call

2a. Identify your critical software

Many European public services and institutions such as the European Commission have, to varying degrees, created inventories of their open source software, tools, business applications and critical software.

Though helpful, having a complete open source software inventory is not a pre-requisite. You can start wherever you are right now.

It's the same conceptual process whether you are a small, medium, national or pan-European public body.

Follow the suggested steps described next to identify your organisation's most critical open source software

2a. EPS Analyses and Extracts Data

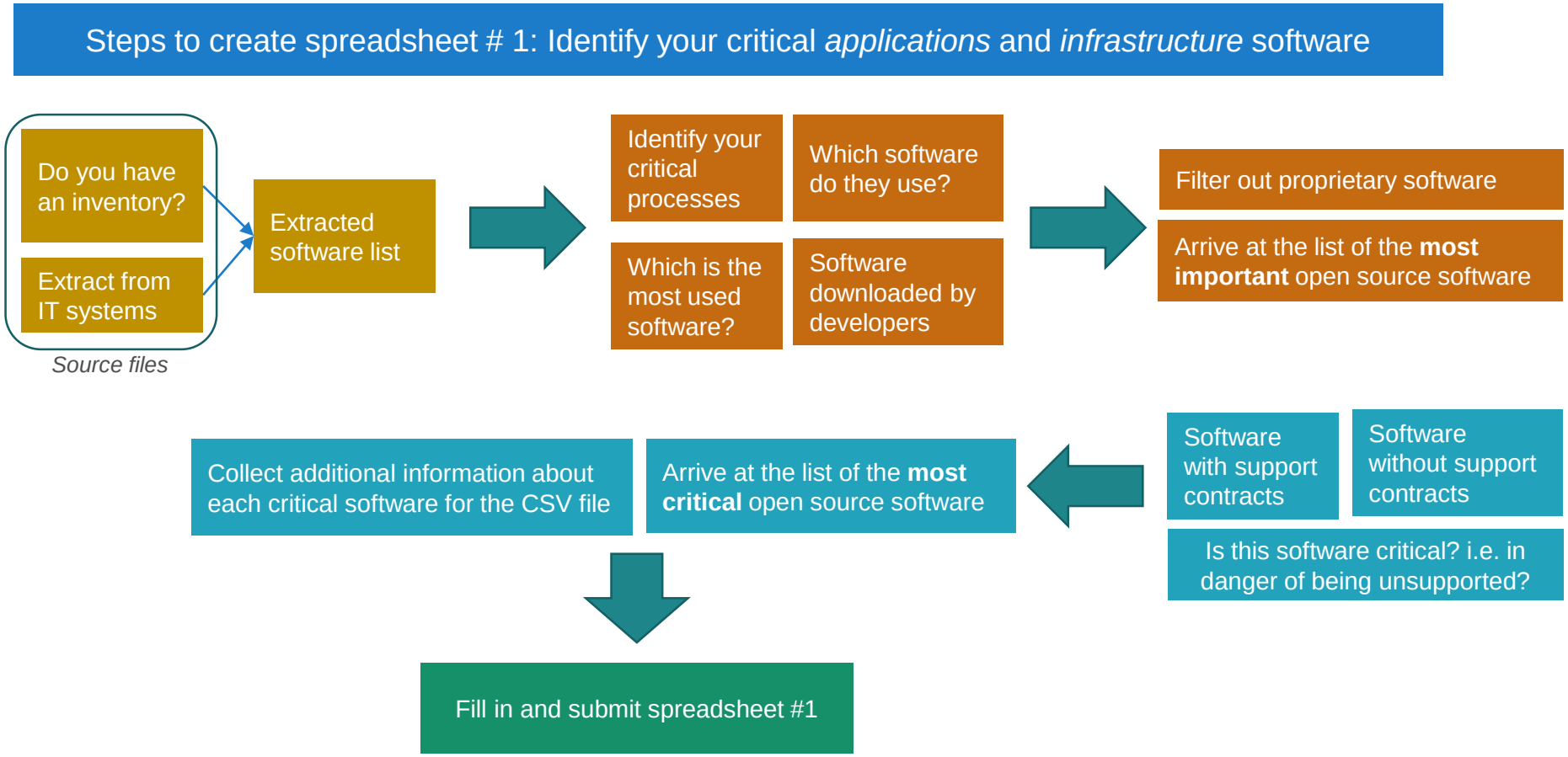
1. Identify critical software
2. Identify critical software dependencies

European Commission

2a.1 The process for identifying applications and infrastructure software

2a. EPS Analyses and Extracts Data

1. Identify critical open source software
2. Identify critical software dependencies

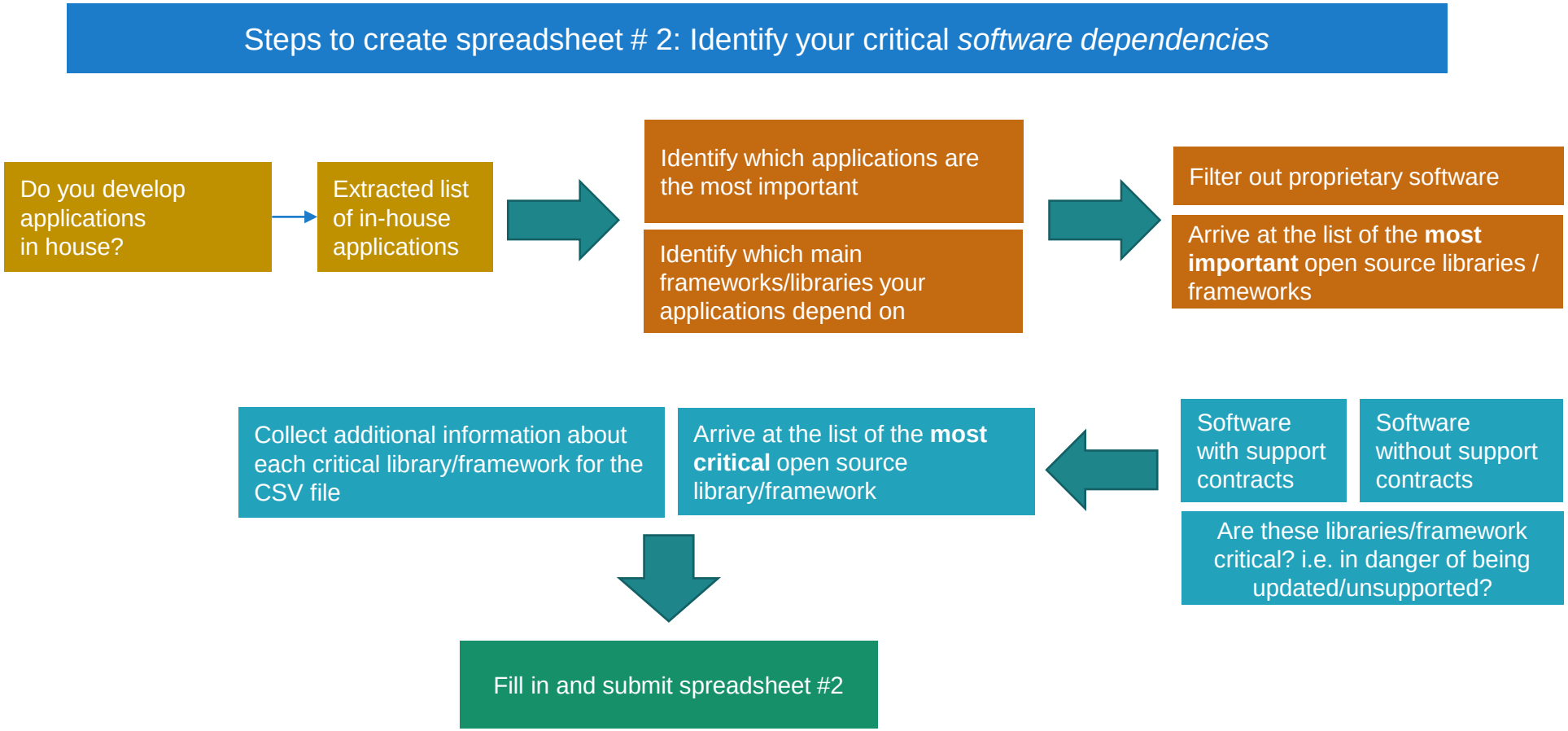


European Commission

2a.1 The process for identifying applications and infrastructure software…

	Medium-Large organisation	Small organisation
Step 1: Get a list of your software items	Contact your Software Asset Manager, they may have an inventory already.	Contact the head of IT, who can facilitate an extraction of the list of software used.
Step 1: Where to look for open source data	Open source software data can be found in the following places: data centres, departmental systems/servers, virtual machines, end user and developer pcs, cloud systems, mobile devices, network switches, routers etc. Data can be equally widespread to include: operating systems such as gnu/Linux (various distributions), applications running on servers for performance, messaging, email and connectivity, software development tools and frameworks and user desktop tools such as web browsers, utilities, office suites, password managers; etc.	
Step 2: Identify the most important ones	Select the most deployed ones and the ones which support the most important services provided by your organisation. The product owner/product manager of each critical service will know about the software used by their service or will be able to identify the person who has this information.	
Step 3: Filter out the proprietary ones	The person who has provided you with the list of elements in step 1 should be able to tell you which software is Free and Open Source or not. The goal is to only retain the 10 to 20 most important ones.	
Step 4: Evaluate the potential critical state	Identify if you have a support contract for each piece of software, from the software asset management, the IT department or the procurement department. Identify the critical state of the open source project by asking technical teams in contact with it.	
Step 5: Collect additional information about the software	Additional information will likely be available from the technical teams who use and operate the concerned open source software.	

2a.2 The process for identifying dependencies (libraries and frameworks)



Note: the EC can figure out recursive dependencies from the named open source libraries and frameworks you use.

2a.2 The process for identifying dependencies (libraries and frameworks)…

	Medium - Large organisation	Small organisation
Step 1: Get a list of in-house applications	Contact your Software Asset Manager, or the CIO/Head of IT to identify the groups with significant software development activity.	Contact the person of the IT team in charge of Software Development.
Step 2: Identify the most important ones and their dependencies	Select the software most deployed and ones which support the most important services provided by your organisation. The product owner/product manager of each application will know about the main libraries and frameworks they depend on. Please note that our focus is on first level dependencies and not the whole tree of recursive dependencies.	
Step 3: Filter out the proprietary ones	The person who has provided you with the list of elements in step 2 should be able to tell you which software is free and open source. The goal is to only retain the 10 to 20 of the most important dependencies ones across all your applications.	
Step 4: Evaluate the potential critical state	Identify if you have a support contract for each dependency, from the software asset management, the IT department or the procurement department. Then identify the critical state of the open source project by asking technical teams in contact with it.	
Step 5: Collect additional information about the software	Additional information will likely be available from the technical teams who operate the concerned open source software.	

Who should I contact in my organisation

Open Source expert/s	Will help you throughout the survey
CIO / CTO	Will help you with general information and to identify the right contact persons for specific topics
Software asset manager	Will help you identify the most used software, as well as other information like support contracts, etc.
Dedicated technical teams	Will help you get detailed informations on specific software
Product owners / product managers of services / processes	Will help you identify applications/infrastructure used by their respective services
Product owners / product managers of in-house software developments	Will help you identifying frameworks/libraries used by their respective applications
Procurement department	Will help you identifying support contracts

FAQs/Advice – 1

What is the difference between most used software and critical software?	Most used means that the software is widely used inside the organisation; critical means that it supports key processes. But in our case, we need to check the sustainability/health of the most used/critical software. If anything is not right, lets put it on the critical list.
We don't understand the needed output?	Call us, we can help you. Our contact information is at the end.
I am worried about security, sensitivity.	Your data is safe and we just want software names
What will we get out of this?	You will understand which software is critical for your organisation, how well it is supported, and also help protect European Public Services by helping us/them collectively identify and protect their critical open source software.

FAQs/Advice – 2

How do I figure out my most important processes?	Most important processes are the ones necessary for your organisation to perform its core duty.
This looks difficult and a lot of work!	We have listed key roles of who should be able to help you inside your organisation. If you can't complete all the survey, just send us the information you can get.
Can I just take a blank piece of paper and write out my most critical software?	Yes, you can. This is the simplest use case and it will still be useful to us. However, a more through process may flush out non-obvious critical software.
How much time will this exercise take?	The survey itself is quite brief, but gathering the information to fill the spreadsheet may take you several hours over several days of elapsed time.
How many critical software do you need?	We are looking for your top 10-20 software, but you may need to list more to get the most important/critical ones.
If a software has a valid support contract, can it still be critical?	Yes, if the open source project itself has maintenance issues from the software community that looks after its evolution and growth.
Should we include open core or dual-licensed software?	Yes.

Filling in the spreadsheets

1. Most fields are self-explanatory
2. Study pages below for descriptions of key fields
3. Feel free to leave out optional fields
4. Email us for help

Guidance on filling in fields in Spreadsheet #1

URLs – why should I fill this? It seems rather a clerical job!	If its www.postgresql.org then do not, if its an obscure one, the URL will help us to properly identify the software.
Number of instances	We don't need an exact number, just an order of magnitude.
Total number of instances of this type of software (both open source and proprietary). Why do you need this, especially the proprietary one?	This helps us understand the relative usage of open source in your organisation. For example, if you have 10 PostgreSQL servers running, alongside 10 MariaDB servers and 15 Oracle DB servers, the total servers would be 35. We are aware that this could be cumbersome, so if you have another way of counting (e.g. the size of the instances, etc.), please feel free to use it.
Maximum degree of importance of this software project for your organisation	In this context, "important" is measured on an arbitrary scale for "Critical" to "Low". "Critical" would mean that the dysfunction of the concerned instance of this open source item would have a severe impact on a key service you are delivering.
Open source project's governance type	The governance type of an open source project has a important impact on its sustainability model. Only the two major types are proposed as a possible response: • Vendor driven: Some projects are maintained and mainly developed by a single company, which also acts as a software vendor (e.g. https://about.gitlab.com/). • Community driven: This is the most common type of open source project. They can be governed as part of a foundation (tomcat.apache.org/) or have an informal governance (e.g. samba.org/samba/team/). • "I don't know": It is important for us to know your level of awareness of this, so don't hesitate to choose this option if it matches your situation.

Guidance on filling in fields in Spreadsheet #1…contd.

Your assessment of the health of this project	This is ranked on an 5 degree arbitary scale. It can be based on your impression of the responsiveness, facts you might have been made aware of, etc.
Type of support contract your organisation has for this open source project	This question is key for us to understand how the usage of this open source item is secured inside your organisation, and also how this securisation contributes to the sustainability of the open source project itself. • Vendor / Maintainer of the project: For community-driven projects, there is no single entity governing the project, but a group of contributors. The most active, regular and long term contributors, are generally called “maintainers”. They often work for companies who sell support contracts. • Company unrelated to the project: Any company can legally sell support for open source projects. Generic IT service companies sometimes sell contract support for open source even if they don’t, or rarely, contribute to the project. • Internal: your internal team can participate to a project’s community and ask for support on public discussions channels, for free but with no assurance of getting an answer. • None : if your are not aware of any specific support for this project in your organisation
Your contribution to this open source project	In case your team is involved with a open source project, we would be interested in knowing about the nature of your involvement. The possible types of contribution are: • **Code**: this can be submitting simple patches or full features (details can be given in the following field) • **Non-code**: this can be documentation, translation, QA (tests, bug reporting/triaging), support, communication, etc. • **Financial sponsorship for the project**: donating money directly to the project • **Financial sponsorship of the project foundation**: donating money to the umbrella organisation that hosts the project (like the Apache Foundation)

Guidance on filling in fields in Spreadsheet #2

Number of applications this dependency is used in [Mandatory]	This is the approximate number of applications that rely on this framework/library. We don't need an exact number, just an order of magnitude.
Maximum degree of importance of the applications using this open source library/framework [Mandatory]	In this context, "important" is measured on an arbitrary scale for "Critical" to "Low". "Critical" would mean that the dysfunction of the applications using this open source framework/library would have a severe impact on a key service you are delivering. Example: If you have 10 applications relying on the django framework, 9 used for low impact internal services and 1 used for a key service to the citizens, then the rating for the open source framework/library would be "Critical".

Thank you

Contacts

1. Inno^3: Camille MOULIN (cmoulin@inno3.fr) **Note**: Inno^3 and Deloitte are subcontractors on this project.
2. FOSSEPS Project, DIGIT, EC: Saranjit ARORA (DIGIT-OSPO@ec.europa.eu)
3. OSPO, DIGIT, EC: Miguel DIEZ-BLANCO (DIGIT-OSPO@ec.europa.eu)

European Commission