AboutCode Foundation

How PURL transforms OSPO operations

with Philippe Ombredanne, AboutCode

Agenda

OSPOs challenges: The world is changing fast

History of the SBOM world: Before and after mandated compliance

What is PURL? Introduction to Package-URL

Benefits for OSPOs: Unified license compliance, faster vulnerability responses, consistent workflows, and vendor-neutral tooling

How can we fix supply chain clarity? Community efforts

Latest developments: Current status and roadmap

Calls to action: Encouraging community participation

FOSS-first mission: Make it easier to reuse open source, safely and efficiently, with open source code and open data

Philippe Ombredanne

pombredanne@aboutcode.org
https://github.com/pombredanne
https://www.linkedin.com/in/philippeombredanne

- Lead maintainer of AboutCode
  - Open source code, data, and standards to automate and secure software supply chains with transparency and confidence
  - https://aboutcode.org
- Creator of PURL (Package-URL) and VERS, co-founder of SPDX and ClearlyDefined, and core contributor to CycloneDX
- CTO and co-founder of nexB
  - Providing SCA services and AboutCode support since 2007
  - https://nexb.com

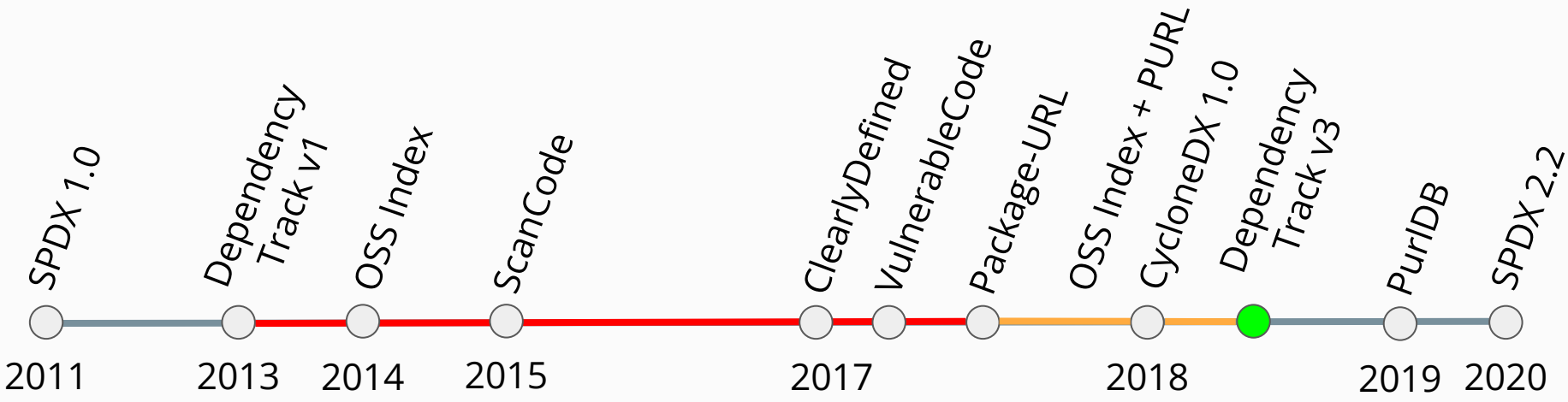
Managing enterprise-wide open source adoption

- Volume of FOSS is exploding
  - Reused and contributed
- Less budget and resources
  - Imperative to automate (almost) everything
- More compliance requirements
  - From license and origin to security, project health, and LLMs
  - CRA, NIS2, DORA and more
- Confusing tooling landscape
  - Both commercial and FOSS
- Emerging, conflicting standards for SBOMs and VEXs
- Difficult to navigate security and software development
- AI (especially GenAI) changes everything

How PURL transforms OSPO operations

1. Unified license compliance
2. Faster vulnerability responses
3. Consistent workflows
4. Vendor-neutral tooling

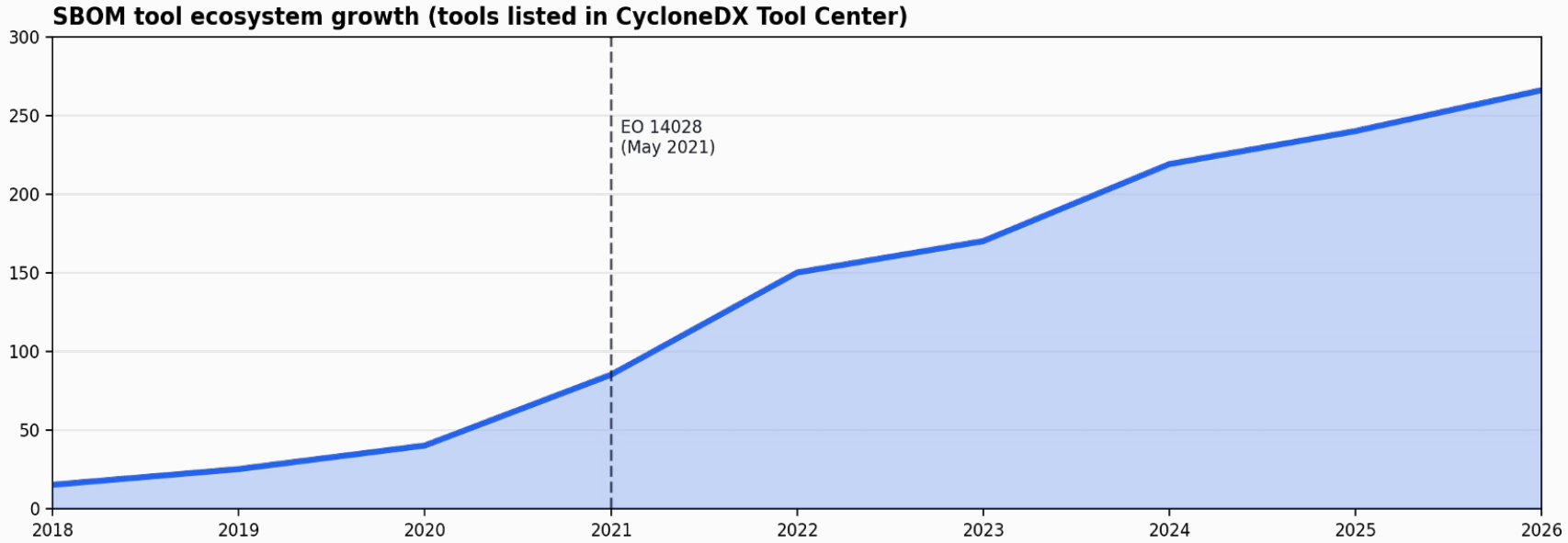
History of the SBOM World, Part 1



History of the SBOM World, Part 2

- Vulnerability management = primary use case
    - Human-readable AND machine-parseable software identifiers
        - Hash/checksum=GPS coordinates vs. PURL=street address
    - Portable inventory of components
    - Provenance of components

History of the SBOM World, Part 3



PURL made it all happen.

SBOMs describe software composition.

SBOMs describe software composition.

PURLs make those descriptions machine-readable and actionable.

PURL is the standard for package identification across ecosystems

- Package-URL (PURL) is the glue between all your software supply chain tools, data, and standards
  - Adopted in all SBOM and VEX specs
  - Most SCA tools
  - Many vulnerability databases
- Across Dev to Ops to Sec (to DevSecOps)

- Simple, obvious, expressive syntax:
  - `pkg:npm/file@1.9.1`
  - `pkg:deb/ubuntu/7zip@21.07+dfsg-4`
  - `pkg:pypi/django@1.11.1`

- Ecma standard at TC54
  - Planned ISO JTC1 standardization
  - Already included in CycloneDX, OASIS, ISO CSAF, ISO SPDX, and CVE schema

and VERS is a new standard for version ranges syntax

- Problem: Each ecosystem has its own convention to specify version ranges
- Solution: An expressive version range string that is minimal and PURL adjunct
    - Specify ranges reliably across tools and languages for dependencies and vulnerabilities
    - Path to universal dependency resolution and better vulnerability remediation automation
- A version range specifier (VERS) is a URI with this syntax:
    - `vers:npm/1.2.3|>=2.0.0|<5.0.0`
    - `vers:pypi/0.0.1|0.0.2|2.0pre1`
- Started with VulnerableCode in the "univers" library and now used in CycloneDX, CLE, TEA, CSAF and ScanCode
- Planned Ecma standardization with TC54-TG2 in Q2 2026

How PURL transforms OSPO operations

1. Unified license compliance
2. Faster vulnerability responses
3. Consistent workflows
4. Vendor-neutral tooling

Use PURL for origin and SPDX for license

PURL for accurate package origin

+ SPDX license expressions for accurate licensing documentation

= Unified policies, streamlined compliance

- Easy creation of accurate attribution notices
- Easy injections in SBOMs (Cyclone and SPDX)
- Easy policies based on accurate identifiers
  - At the license expression level
  - Or at the PURL + license level
  - And in the context of a product or usage
- This makes automation possible
  - SBOM validation, compliance reporting, vulnerability management, and more

How PURL transforms OSPO operations

1. Unified license compliance
2. Faster vulnerability responses
3. Consistent workflows
4. Vendor-neutral tooling

It’s Saturday, September 12th, 2026.

It’s Saturday, September 12th, 2026.

A key provision of the EU CRA went into effect yesterday.

You sell your software in Europe. The CRA categorizes it as a "*critical digital product*".

You sell your software in Europe. The CRA categorizes it as a "*critical digital product*".

And a vulnerability was just discovered in your software...

You have 24 hours to process and report!

Reporting vulnerabilities for CRA

1. Continuously track all the third-party packages used in all products
    - And how they are used
2. Continuously watch for known vulnerabilities
    - And listen to reported incidents
3. Determine exploitability
    - What are the process, teams and tools you need to achieve this?
4. Send reports to national CSIRTs and ENISA

- 24 hours: Report known-to-be-exploitable vulnerabilities
- 72 hours: Detailed reports
- 14 days: Final Reports

- Watch "Mission Possible: 24 Hours To Save The World With Security Compliance" for more information: https://sched.co/28XRP

1. Continuously track all the third-party packages used in all products

- Collect package inventories (SBOMs)
  - Minimum viable = subset of packages in the deployed product
  - Know what's in your software
  - Keep track of each shipped product details for 5 years

- Standardized SBOM format interoperability ensures consistent identification across formats and tools
  - Reference the same package using identical PURL identifiers

CycloneDX

https://cyclonedx.org/docs/1.6/json/#components_items_purl

SPDX®

https://spdx.github.io/spdx-spec/v3.0.1/model/Software/Properties/packageUrl/

2. Continuously track known vulnerabilities

- How to get a reliable vulnerabilities feed, open and keyed by PURL?
  - NVD CVE backlog is growing and CPEs do not help [1]
  - 38% growth rate in CVEs from 2023 to 2024 [2]

- Go upstream, straight from the source: Data directly from projects
  - GitHub advisory database and Google OSV are also doing a good job there
  - Too much noise with junk CVE, malware (no CVE for this!), unmaintained code…
- Need contextual severity to prioritize work and determine reachability
  - Use code reachability and static and dynamic analysis!

[1] Lyons, Jessica. “NIST’s security flaw database still backlogged with 17K+ unprocessed bugs. Not great.” The Register, 2 Oct. 2024, www.theregister.com/2024/10/02/cve_pileup_nvd_missed_deadline.
[2] "Metrics." CVE: Common Vulnerabilities and Exposures, 25 Aug. 2025, https://www.cve.org/about/Metrics.

2. Continuously track known vulnerabilities: 4 types

1. CI/CD pipeline scanning what's being deployed
    - At code build in CI/CD pipelines
2. Scanning artifactory stores
    - At artifact ingestion with Jfrog Artifactory, package registries
3. Monitoring cycle for the whole release
    - Global monitoring with DejaCode (or DependencyTrack)
4. Checking for current and past releases
    - Important that current releases include no known vulnerabilities
    - Critical to also ensure for previous releases!

2. Continuously track known vulnerabilities: Formats

- Standardized VEX and VDR format interoperability
  - Precise vulnerability matching and license compliance checking with accurate package identification
- Faster vulnerability correlation and incident response
  - Query a single PURL-indexed database rather than hunting across multiple

CycloneDX

https://cyclonedx.org/docs/1.6/json/#components_items_purl

CVE

https://github.com/CVEProject/cve-schema/releases/tag/v5.2.0

OpenVEX

https://github.com/openvex/spec/blob/main/OPENVEX-SPEC.md#appendix-b-software-identifier-types-table

CSAF

https://docs.oasis-open.org/csaf/csaf/v2.0/os/csaf-v2.0-os.html#31334-full-product-name-type---product-identification-helper---purl

OSV

https://google.github.io/osv.dev/post-v1-query/#parameters

3. Determine exploitability

- Historically delegated to US NVD for definition and severity, but not a sustainable process…
  - NVD backlog still growing
- SO MANY QUESTIONS!
  - What is a vulnerability? Who defines that? Is any bug a vulnerability? (See the kernel CVEs)
  - How to determine exploitability? In general? Or in a specific usage?

- Data sources for exploitability:
  - EPSS (Exploit Prediction Scoring System)
  - US CISA KEV (Known Exploited Vulnerabilities)
  - Exploit scripts in VulnerableCode

3. Determine exploitability, practically

- Traditional industry approach = ask dev teams to investigate 😓
  - Developers are not experts
  - Too many vulnerabilities and data feeds now
  - Developers drowning in security work

- Instead, reduce the number to investigate
  - Remove internal use, and rescore
  - And only a few developers see full end-to-end environment
  - Difficult to determine how and where packages are used
- Then, investigate exploitability
  - And the internal context
  - And link SBOMs to product code

4. Send reports to national CSIRTs and ENISA

- How? And to whom? Still unclear …
- The what is clearer:
  - A machine-readable SBOM = an inventory of third-party (and own) packages
  - A vulnerability disclosure report (VDR) or similar vulnerability exploitability report (VEX) = an assessment of the vulnerabilities, their impact, and possible remediation
- This will be publicly available as a disclosure
  - Need to ensure data is accurate
  - Reporting structure will be challenging for proprietary embedded in software supply chains
    - FOSS usage update cycle (usually) faster than proprietary
    - System and network admins already implement quickly, but developers will need help

/end rant

>>> back to main topic

How PURL transforms OSPO operations

1. Unified license compliance
2. Faster vulnerability responses
3. Consistent workflows
4. Vendor-neutral tooling

Enable efficient automation

- With PURL, automation is possible
  - Clear automated outcomes
  - Only handle policy exceptions
  - Automate all the rest!

- Consistent approval workflows for diverse development teams
  - Engineering teams using different stacks all feed into the same OSPO review pipeline
  - Teams don't need ecosystem-specific workflows for npm vs. Maven vs. PyPI
    - One identifier schema covers all of them

And actionable SBOMs and SCA

- You get an SBOM. With a correct PURL, you can lookup for:
  - Vulnerabilities
  - Licensing
  - Quality/Sustainability/Lifecycle
  - Internal policies

- You share a VEX. With a correct PURL, upstream suppliers and downstream users will not come back with inquiries about packages.

- You scan code. With correct PURLs, results can flow through your toolchain without any friction.

How PURL transforms OSPO operations

1. Unified license compliance
2. Faster vulnerability responses
3. Consistent workflows
4. Vendor-neutral tooling

Choose the best tools for your use case

- Security scanners and compliance tools can reliably match packages using PURL as a common identifier:
  - SCA
  - Security
  - Vulnerability databases
- Correct PURLs facilitate accurate tracking of components through the entire software supply chain, from development to deployment.

PURL is adopted industry- and community-wide

All open source SCA and SBOM tools use PURL, including:

- AboutCode's ScanCode and DejaCode
- Linux Foundation's ORT and Fossology
- OpenSSF OSV and GUAC
- OWASP Dependency-Track, Dependency-Check, and cdxgen
- All OWASP CycloneDX libraries
- GitHub's Dependency Graph
- Microsoft's OSS Gadget, SBOM tool
- Anchore's Syft and Grype
- Aquasec's Trivy
- LG's FOSSLight
- SCANOSS
- Snyk's Parlay

Most proprietary SCA, SBOM, and code host tools use PURL, including:

- GitHub
- GitLab
- Snyk
- Mend
- BlackDuck
- Sonatype

Vulnerability databases use PURL, including:

- Google's OSV
- Sonatype's OSS Index
- CVE specification v5.1
- VulnerableCode

PURL facilitates better compliance processes for end users, including:

- Most free and open source software foundations
- Five of the Big Tech companies, with three building their entire SCA compliance operations on PURL
- A leading database company
- A leading Linux company
- European and US government agencies
- All major European car manufacturers and most of their vendors
- Major US chip and microprocessor providers
- Four leading European industrial companies
- A leading European medical devices company

Community adoption is great.

But, we now need better and standardized PURLs.

How can we fix supply chain clarity?

SBOM clarity is … a work in progress

2024 SCA for Containers Report:

- Compared SCA tools for container scanning using SBOM accuracy as proxy
- The results were not great…
- Some invented Package IDs and invalid PURLs, lots of "creativity"
- Download report: https://nexb.com/sca-containers
- Watch talk from OSS Summit EU 2024: https://sched.co/1ej58

2025 CMU SEI's SBOM Plugfest Report:

- Compared SCA tools using SBOM accuracy as a proxy
- The results were not great … again.
- Inconsistencies on the format and content across many tools
- Download report: https://www.sei.cmu.edu/news/study-finds-key-causes-of-divergence-in-software-bills-of-materials/

Validate PURLs

- Extensive test suite as data (JSON)
  - Build compact dataset of all known PURLs to validate if a PURL exists
    - Using automatons, DAWG, FSTs, etc
  - Supported by distributed jobs to continuously catalog all the PURLs
  - Collect all Maven, Rust, npm and more (using existing PurlDB code)
  - Package as libraries with code and data (in Python, Go, and more)
- ... will be completed by validation service API (online) that could can also deploy on-premises
- Need to support that for all PURL libraries (JS, Ruby, C#, C, etc.)

Create a PURL benchmark

- Thoroughly validated set of open source app, container and package codebases
- Goal = Scanning tools can check they return the exact list of correct PURLs found in a codebase
  - Peer reviewed and updated!
  - Precursor to a future open SBOM benchmark
  - PURL-first
  - Support both SPDX and CycloneDX
- Call to Action = Let's join forces!
  - OpenChain and Bosch "Digital Dummies" project: https://github.com/Open-Source-Compliance/Sharing-creates-value/tree/openchain_and_friends_2026_prep/Tooling-Landscape/Dummy_Repositories
  - Nokia SBOM-QA.org project: https://nokia.github.io/SBOM-QA/
  - AboutCode PURL benchmarks

How to integrate PURLs into your tooling and workflows

- Improved test suite from Ecma standardization work on PURL
- Libraries are getting up to speed on support for ECMA-427, and we are helping as much as we can
    - We need YOUR help!
- No more excuses if your tools, FOSS or proprietary, produce junk PURLs! 🗑️
    - Some are really funky! 🎷
- Demand from all your supply chain partners to provide VALID PURLs
    - Or reject their SBOMs/VEXs

Ecma TC54 for software and system transparency

- Covers PURL and VERS (standard for version ranges)
  - https://github.com/Ecma-TC54/tg2
  - PURL and VERS are already part of EcmaISO standards, indirectly with CSAF 2.0 and SPDX 2
- Active participation by key community contributors, industry players and open source foundations

- TC54 started for CycloneDX
  - CycloneDX is now **ECMA-424**
- PURL and VERS in CycloneDX specs
  - PURL is now **ECMA-427**
- Includes TEA (Transparency Exchange API) and CLE (Common Lifecycle Events) task groups
  - CLE is now **ECMA-428**
- Other TG for Contributing.yaml

Better PURLs? Yeah!

- ✅ A standard with Ecma International
- ✅ Structured PURL types as data (JSON)
- ✅ Open source tools to validate PURL syntax
- ✅ Open data (and tool) to validate existence (offline) in Go, Python, Rust
- ✅ Validation service API (online)
- ✅ Extensive test suite as data (JSON)
- ✅ Adoption in language ecosystems
  - Perl, PEP 725 in Python, Raku
- ✅ PURL adopted in the CVE schema, Maven Central, and for all Rust crates in crates.io
  - Beach operations cleaning packages for Rust, Maven, and nixpkgs
- ✅ Work with PURL library maintainers
  - Adopt test suites (Java, Go, Rust, Perl, Ruby, JavaScript, Raku, etc...)
  - Upgrade to the latest standard
- ✅ Expose open metadata and scans, keyed by PURL

What's next for the PURL community

- More open source tools and open data to validate PURLs
- Validate package source vs. binaries?
- AboutCode + ClearlyDefined
  - Share all the scans, all the SBOMs
- AboutCode + Software Heritage
  - Scan billions of files
  - Get billions of PURLs

- ✅ PURL adopted in the CVE schema
- ✅ PURL for all Rust crates in crates.io
- ✅ PURL for all JARs at Maven Central
- Adoption in language ecosystems
  - Perl, PEP 725 in Python, Raku
- Commando beach operations cleaning packages for Rust, Maven, and nixpkgs
- New website 😍
  - https://packageurl.org/

New developments

ClearlyDefined

Ecosystem challenge:

- Generate SBOMs at scale for every build or release.
- And fix the same missing or wrongly identified metadata, again.

ClearlyDefined crowdsources a global database of licensing and metadata

- 55M+ packages soon keyed by PURL!
- Unlocking the data for distributed on-prem usage

ECOSYSTEM

OSS Review Toolkit · OPENCHAIN · AboutCode · SBOM · SPDX · CycloneDX · TODO · fossology · SW360 · GUAC · FOSSLight · OWASP

ORGANIZATIONS

Microsoft · GitHub · HPE · Bloomberg Engineering · kusari · ECLIPSE FOUNDATION · cisco · SAP · THE LINUX FOUNDATION · OPENRAIL ASSOCIATION

PURL (and VERS) need you!

- PURL types for your ecosystems!
- VERS standardization at Ecma, then ISO
  - Already included in CycloneDX, OASIS and ISO CSAF
- Help all CVEs contain (correct) PURLs
  - Open benchmark to validate SBOM accuracy (or rather PURL accuracy)
- PURL+VERS-based general purpose dependency resolution
- **C/C++ PURL open registry with C++ community**
  - And beyond C/C++
  - The missing registry for packages that do not have a registry
- PURL-keyed standard API for every ecosystem
  - We must stop rewriting API parsers and remapping package metadata schemas everywhere

Community support to help us do good!

- German Sovereign Tech Agency (STF) to support PURL and standardization
- Grants from NLnet Foundation with the EU NGI programs all about PURL:
  - CRAVEX, FederatedCode, T-Rust, back2source, Maven Heaven
- Grants from EU NGI Search:
  - AI-Generated Code Search, matching PURLs
- Completed GitHub Secure OSS Fund
- EU and ECCC (European Cybersecurity Competence Centre)-funded OCCTET project: occtet.eu
  - with Eclipse, DoubleOpen, Bitsea, European Digital SME Alliance, Expertware, Red Alert Labs
- ZEISS FOSS Award, Bloomberg FOSS Grant, Mercedes FOSS grant, Microsoft grant
- And grants from from two US big tech, two large US and several EU companies
- Help us do more!

Together, we can build better software compliance processes

Join the community and contribute!

AboutCode

GitHub: github.com/aboutcode-org

Slack: join.slack.com/t/aboutcode-org/shared_invite/zt-1paqwxccw-luafuiAvYJFkTqGaZsC1og

Community calls:
Mondays at 15:00 UTC at
meet.jit.si/AboutCode

TC 54 PURL + VERS

github.com/package-url/purl-spec

Join #purl channel on AboutCode slack for latest updates

Community calls:
Biweekly on Wednesdays at 16:00 UTC at
meet.google.com/ryq-aimn-ghd

Help with the 200+ issues:
github.com/orgs/aboutcode-org/projects/52/views/39 and
github.com/package-url/purl-spec/issues

ClearlyDefined

docs.clearlydefined.io/docs/get-involved/

CycloneDX

cyclonedx.org/participate/contribute/

SPDX

spdx.dev/engage/participate/

Let's connect!

Any questions?

pombredanne@aboutcode.org
https://github.com/pombredanne
https://www.linkedin.com/in/philippeombredanne