Changes in SPDX 3.0

Bob Martin
Senior Software and Supply Chain Principal Engineer
Cyber Solutions Innovation Center
MITRE Labs

23 January 2024
Market Transparency through “Software Bill of Materials”

- Third party components are a known systemic risk.
  - Transparency can drive tools and behavior to document risk, support mitigations, and drive better SW development practices.

- NTIA at Commerce launched an open, community-driven, cross-sector “multistakeholder process” to promote software component transparency in 2018.
  - Understand the problem and define basics of SBOM
  - Develop use cases across sectors on how such data can be used, today and in the future.
  - Guidance on how to use existing standards to implement SBOM
    - Software ID tags (SWID)
    - Software Package Data Exchange (SPDX)
    - CycloneDX

- Deliverables https://ntia.gov/sbom
- Moved effort to DHS CISA (2021)…
- Deliverables https://cisa.gov/sbom
Roles and Benefits for SBOM Across the Supply Chain
NTIA Multistakeholder Process on Software Component Transparency
Use Cases and State of Practice Working Group

Introduction
The Software Supply Chain
About this document: Goals and Methodology
Perspective: Produce Software
Reduce unplanned, unscheduled work
Reduce code bloat
Adequately understand dependencies within broader complex projects
Know and comply with the license obligations
Monitor components for vulnerabilities
End-of-life (EOL)
Make code easier to review
A blacklist of banned components
Provide an SBOM to a customer
Perspective: Choose Software
Identify potentially vulnerable components
A more targeted security analysis
Verify the sourcing
Compliance with policies
Aware of end-of-life components
Verify some claims
Understanding the software’s integration
Pre-purchase and pre-installation planning
Market signal
Perspective: Operate Software
Organization can quickly evaluate whether it is using the component
Drive independent mitigations
Make more informed risk-based decisions
Alerts about potential end-of-life
Better support compliance and reporting requirements
Reduce costs through a more streamlined and efficient administration
Ecosystem, Network Effects, and Public Health Benefits of SBOM
SBOMs as a market signal
Software Component Transparency: Healthcare Proof of Concept Report
Drafted as part of a process convened by the National Telecommunications and Information Administration

Survey of Existing SBOM Formats and Standards
NTIA Multistakeholder Process on Software Component Transparency
Standards and Formats Working Group
Final Version - 20191025

October 1, 2019
NTIA Software Transparency Effort’s Formats and Tooling Working Group “Current SBOM Options Available”

SPDX

File formats: .xls, .rdf, .spdx, .json, .yml, .xml

SWID

File formats: .xml

CycloneDX

File formats: .json, .xml
SBOM Definition
NTIA Minimal Elements (EO 14028)

<table>
<thead>
<tr>
<th>Data Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier Name</td>
<td>The name of an entity that creates, defines, and identifies components.</td>
</tr>
<tr>
<td>Component Name</td>
<td>Designation assigned to a unit of software defined by the original supplier.</td>
</tr>
<tr>
<td>Version of the Component</td>
<td>Identifier used by the supplier to specify a change in software from a previously identified version.</td>
</tr>
<tr>
<td>Other Unique Identifiers</td>
<td>Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases.</td>
</tr>
<tr>
<td>Dependency Relationship</td>
<td>Characterizing the relationship that an upstream component X is included in software Y.</td>
</tr>
<tr>
<td>Author of SBOM Data</td>
<td>The name of the entity that creates the SBOM data for this component.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Record of the date and time of the SBOM data assembly.</td>
</tr>
</tbody>
</table>

Minimum Elements

- **Data Fields**: Document baseline information about each component that should be tracked: Supplier, Component Name, Version of the Component, Other Unique Identifiers, Dependency Relationship, Author of SBOM Data, and Timestamp.
- **Automation Support**: Support automation, including via automatic generation and machine-readability to allow for scaling across the software ecosystem. Data formats used to generate and consume SBOMs include SPDX, CycloneDX, and SWID tags.
- **Practices and Processes**: Define the operations of SBOM requests, generation and use including: Frequency, Depth, Known Unknowns, Distribution and Delivery, Access Control, and Accommodation of Mistakes.


---

SPDX
CycloneDX
SWID


© 2024 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case No: 23-01614-54
### Table 1: SBOM Type Definition and Composition

<table>
<thead>
<tr>
<th>SBOM Type</th>
<th>Definition</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>SBOM of intended design of included components (some of which may not exist) for a new software artifact.</td>
<td>Typically derived from a design specification, RFP, or initial concept.</td>
</tr>
<tr>
<td>Source</td>
<td>SBOM created directly from the development environment, source files, and included dependencies used to build an product artifact.</td>
<td>Typically generated from software composition analysis (SCA) tooling, with manual clarifications.</td>
</tr>
<tr>
<td>Build</td>
<td>SBOM generated as part of the process of building the software to create a releasable artifact (e.g., executable or package) from data such as source files, dependencies, built components, build process ephemeral data, and other SBOMs.</td>
<td>Typically generated as part of a build process. May consist of integrated intermediate Build and Source SBOMs for a final release artifact SBOM.</td>
</tr>
<tr>
<td>Analyzed</td>
<td>SBOM generated through analysis of artifacts (e.g., executables, packages, containers, and virtual machine images) after its build. Such analysis generally requires a variety of heuristics. In some contexts, this may also be referred to as a “3rd party” SBOM.</td>
<td>Typically generated through analysis of artifacts by 3rd party tooling.</td>
</tr>
<tr>
<td>Deployed</td>
<td>SBOM provides an inventory of software that is present on a system. This may be an assembly of other SBOMs that combines analysis of configuration options, and examination of execution behavior in a (potentially simulated) deployment environment.</td>
<td>Typically generated by recording the SBOMs and configuration information of artifacts that have been installed on systems.</td>
</tr>
<tr>
<td>Runtime</td>
<td>SBOM generated through instrumenting the system running the software, to capture only what is loaded and executing in memory, as well as external call-outs or dynamically loaded components. In some contexts, this may also be referred to as an “Instrumented” or “Dynamic” SBOM.</td>
<td>Typically generated from tooling interacting with a system to record the artifacts present in a running environment and/or that have been executed.</td>
</tr>
</tbody>
</table>
Software Bill of Materials Types

- Open source components
- Developed components
- Purchased components
- Build Tools
- Libraries
- Other documents: Multimedia, text
- Makefiles
- Generated code
- Target Images
- Execution
- Micro Controller Firmware
- Disk Images
- Virtual Machine Images
- Container Images
- Package Feeds
- SDKs & Build Tools

Data to address some Supply-chain Levels for Software Artifacts (SLSA) Threats

Source SBOM

Deployed SBOM

Dynamic libraries
External executables
Enabling Adoption of SBOMs

SBOMs

- Agriculture and Food
- Energy
- Transportation
- Chemical Industry
- Postal and Shipping

Sectors

- Water
- Public Health
- Telecommunications
- Banking and Finance
- Key Assets

Product & Service Suppliers

INTEGRATED DEVELOPMENT ENVIRONMENTS (IDEs)

SOURCE CODE & PACKAGE REPOSITORIES

Software Composition Analysis

Tools & Capabilities for Software

FRAMEWORKS

BUILD CHOREOGRAPHY

Cloud Tools

MITRE

© 2024 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case No: 23-01614-54

Tool-to-Tool SBOM Exchange Standard effort targeted here
CISQ/OMG Tool-to-Tool Software Bill of Materials Effort

Launched 24 Sep 2019

https://www.it-cisq.org/software-bill-of-materials/
## 3T-SBOM Usage Scenarios Around SBOMs

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refer, Transfer or Purchase</td>
<td>(definition of what it is)</td>
</tr>
<tr>
<td>Pedigree</td>
<td>(history of how it was produced)</td>
</tr>
<tr>
<td>Provenance</td>
<td>(chain of custody of it)</td>
</tr>
<tr>
<td>Integrity</td>
<td>(cryptographic basis of unalteredness)</td>
</tr>
<tr>
<td>Proper and Legal</td>
<td>(conditions about its use)</td>
</tr>
<tr>
<td>Known Sw Vulns</td>
<td>(known fixes are applied to it)</td>
</tr>
<tr>
<td>Assurance</td>
<td>(safe-secure-resilient)</td>
</tr>
<tr>
<td>SBoM of a SW Service</td>
<td>(SBoM of sw delivering service)</td>
</tr>
<tr>
<td>Supply Chain Sequence Integrity</td>
<td></td>
</tr>
</tbody>
</table>


© 2024 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case No: 23-01614-54
3T-SBOM Standard Timeline

- **Mar 2019**: Introduce Idea of a CISQ / OMG SBOM standards
- **Jun 2019**: Draft CONOPs for Tool-to-Tool SBOM standards
- **Sep 2019**: Face to Face kick-off meeting in Nashville TN
- **Nov 2019**: SBOM paper published
- **Sep 2020**: Consensus model of 3T-SBOM drafted

CISQ/OMG Tool-to-Tool Software Bill of Materials Effort
SPDX SBOM Standard Timeline

- **2010**: SPDX effort starts
- **2011**: SPDX 1.0 released
- **2012**: SPDX 1.1 released
- **2013**: SPDX 1.2 released
- **2015**: SPDX 2.0 released
- **2016**: SPDX 2.1 released
- **2018**: SPDX 2.1 FAST Path to ISO
- **Sep 2020**: Consensus model of SPDX 3.0 drafted
Who is Using SPDX?

- Wide range of users - from sophisticated security software analysts to non-technical procurement and compliance executives
  - Requires a wide range of “serialization” formats ranging from RDF/XML, to JSON, to YAML, to Spreadsheets

- Software and Software as a Service providers - from the very large to to the very small including many software tooling vendors
  - Many providing or utilizing open source libraries which support overall adoption

- Open source package maintainers across a wide variety of ecosystems (e.g. Python Poetry, Maven, Gradle, NPM)
Jan 2021: Merged w/SPDX

Common goals
Already working together
Fewer meetings
SPDX Governance

release 3.0 will be reviewed by the OMG Architecture Board

- Open
  - Is open to contributions from anyone interested - all team and working group meetings are open
  - Team leads nominated by any participant and steering committee composed of team leads

- Transparent
  - Most of our work is maintained in the SPDX Github repo including minutes

- Inclusive
  - Leads and steering committee consists of individuals from diverse backgrounds and industries
  - Over 40 organizations have contributed directly to the SPDX spec representing a majority of industry segments (Technology, Critical Infrastructure, Financial, Gov’t) and geographies
Timeline of SPDX Evolution - Use Case by Use Case

**Standardized Single Package Information:**
- **2010:** SPDX begins
- **2011:** SPDX 1.0
- **2013:** SPDX 1.2
- **2015:** SPDX 2.0
- **2018:** 2.1 to ISO
- **2019:** SPDX 3.0 initial draft
- **2020:** 3T-SBOM initial draft
- **2022:** SPDX 3.0

**Compliance Use Cases:**
- **Machine and human readable formats**
- **2010:** Compliance
- **2011:** SPDX 1.0
- **2013:** SPDX 1.2
- **2015:** SPDX 2.0
- **2018:** 2.1 to ISO
- **2019:** SPDX 3.0 initial draft
- **2020:** 3T-SBOM initial draft
- **2022:** SPDX 3.0

**Package Relations:**
- **2010:** Package relations
- **2011:** SPDX 1.0
- **2013:** SPDX 1.2
- **2015:** SPDX 2.0
- **2018:** 2.1 to ISO
- **2019:** SPDX 3.0 initial draft
- **2020:** 3T-SBOM initial draft
- **2022:** SPDX 3.0

**Security use cases:**
- **2010:** Security
- **2011:** SPDX 1.0
- **2013:** SPDX 1.2
- **2015:** SPDX 2.0
- **2018:** 2.1 to ISO
- **2019:** SPDX 3.0 initial draft
- **2020:** 3T-SBOM initial draft
- **2022:** SPDX 3.0

**Build Data Security AI Lite**

**SPDX & 3T-SBOM efforts merge:**
- **2010:** SPDX
- **2011:** SPDX 1.0
- **2013:** SPDX 1.2
- **2015:** SPDX 2.0
- **2018:** 2.1 to ISO
- **2019:** SPDX 3.0 initial draft
- **2020:** 3T-SBOM initial draft
- **2022:** SPDX 3.0

**Free ISO Standard:**
- **ISO/IEC 5962 SPDX available**

**Profiles:**
- **New areas of use cases:**
  - Build
  - Data
  - Security
  - AI
  - Lite

**Transition of SBOM work to DHS**

**Executive Order 14028**

**NTIA:**
- **Software Transparency begins**

**3T-SBOM:**
- **OMG/CISQ begins w/ CONOPs for Tool-to-Tool SBOM**

**EU Cyber Resilience Act**

**Legislation:**
- **Proposed software transparency, updatability & bill of material as reqts in safety critical sectors (automotive & healthcare)**

**MITRE**

© 2024 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case No: 23-01614-54
SPDX SBOM Ecosystem

SPDX Work Group

Standards
e.g. Open Chain,
ISO, etc.,

Tool Developers

Products
Services

Upstream / Producer

Open Source Projects
3rd Party Suppliers

Code, Binaries
*License IDs, SBOMs*

Intermediate Consumer / Producer

Assemble, Develop

Products
Services

Code, Binaries, Products, Services, *License IDs*, SBOMs

Downstream / Consumer

Consume

Products
Services

© 2024 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case No: 23-01614-54
SPDX 3 model

The Software Package Data Exchange (SPDX®) is a standard format for communicating information about components associated with software packages. It has wide industrial adoption as a standardized Software Bill

Branches and Formats

The editable files are written in a constrained subset of Markdown and are stored in the main branch.

These files are automatically processed into the following formats available at [https://spdx.github.io/spdx-3-model/](https://spdx.github.io/spdx-3-model/):

- HTML files generated by Ontopy: [https://spdx.github.io/spdx-3-model/auto-generated/](https://spdx.github.io/spdx-3-model/auto-generated/)
- JSON-LD context file: context.json
- Model SHACL and OWL files:
  - Turtle format: model.ttl
  - JSON-LD format: model.jsonld

Note that these files are also available in the gh-pages branch.

People who wish to read the current version of the information should be viewing the generated files, while anyone wanting to edit should be working on the former.

For information about how to contribute to a specific profile, please see [Contributing.md](https://github.com/spdx/spdx-3-model).

Contribute!

Feel free to join us and contribute! The discussions are happening on the spdx-tech mailing list and during our weekly meetings. All the details are in: [https://spdx.dev/participate/tech/](https://spdx.dev/participate/tech/)

https://github.com/spdx/spdx-3-model
Core Profile Overview

- Defines foundational concepts which are the basis for all SPDX 3.0 profiles
Software Profile Overview

- Defines concepts related to software artifacts
- Introduces and supports CISA SBOM Types
  - Design/Source/Build/Deployed/Runtime/Analyzed

SPDX 3.0 Simple Data Types, Complex Data Types and Enumerations

Legend

**Italics** - abstract, you must use a subclass

**Underline** - value type/struct semantics, equality determined by comparing values

Complex Data Types

These types have value type/struct semantics - equality is determined by comparing values and they MUST NOT be referenced by name across documents. Serialization formats MAY enable de-duplication within a single document.

CreationInfo

- `+ specVersion`: SemVer
- `+ profile`: ProfileIdentifier[1..1]
- `+ created`: DateTime
- `+ dataLicense`: “CC0”
- `+ createdBy`: Agent[1..1]
- `+ createdUsing`: Tool[0..1]
- `+ comment`: String[0..1]

NamespaceMap

- `+ prefix`: String
- `+ namespace`: IRI

ExternalMap

- `+ externalId`: IRI
- `+ locationHint`: URL[0..1]
- `+ verifiedUsing`: IntegrityMethod[0..1]
- `+ definingArtifact`: Artifact

ExternalIdentifier

- `+ externalIdentifierType`: ExternalIdentifierType
- `+ identifier`: String
- `+ comment`: String[0..1]

PositiveInteger/Range

- `+ start`: PositiveInteger
- `+ end`: PositiveInteger

ProfileIdentifier

- `+ name`: String

ExtensionClass

- `+ externalReferenceType`: ExternalReferenceType
- `+ baseArtifacts`: IRI
- `+ containerType`: MediaType[0..1]
- `+ comment`: String[0..1]

IntegrityMethod

- `+ comment`: String[0..1]

Hash

- `+ algorithm`: HashAlgorithm
- `+ hashValue`: byte[1..1]

RelationshipType

- `+ start`: Artifact
- `+ end`: Artifact

ExternalReferenceType

- `+ id`: URI
- `+ type`: URI

HashAlgorithm

- `+ hash`: String
- `+ format`: “SHA-256”

Provenance

- `+ ancestor`: Artifact
- `+ availableProvenance`: URI
- `+ variant`: Artifact

License

- `+ hasHeaderValue`: String
- `+ licenseVersion`: String

Obsoletes

- `+ buildTool`: String
- `+ testTool`: String
- `+ dependencyManifest`: String
- `+ distributionArtifact`: String
- `+ sample`: String
- `+ example`: String
- `+ documentation`: String
- `+ test`: String
- `+ requirementFor`: String

RelationshipCompleteness

- `+ complete`: [true|false]
- `+ incomplete`: [true|false]
- `+ requires`: URI

RelatedTo

- `+ relationship`: URI
- `+ relationshipType`: URI

SoftwarePurpose

- `+ purpose`: String
- `+ purposeType`: URI

DependencyScope

- `+ scope`: [“build”|“distribution”|“runtime”]

Dependency/Requirement

- `+ requirement`: [“obsoletes”|“requires”]
- `+ provided`: URI
- `+ providedFor`: URI

AnnotationType

- `+ annotationType`: URI
- `+ annotationValue`: String

Reserved

- `+ reserved`: [“true”|“false”]

Other

- `+ other`: String

AssessmentType

- `+ assessmentType`: URI

SimpleType

- `+ simpleType`: String
- `+ simpleTypeType`: URI

SimpleData Types

- `SemVer`: String
- `MediaTypes`: String
- `EmailAddress`: String

Example:

- `SemVer: 2.0.0`
- `MediaType: application/zip`
- `EmailAddress: user@example.com`

© 2024 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case No: 23-01614-54
SPDX Profiles Overview

Security
- vulnerability details related to software

Build
- provenance and reproducible builds

AI
- Information about AI models - ethical, security, and model data

Data
- Information about datasets - AI and other data use cases

Licensing
- Information about copyrights and licenses - supports compliance

Software
- Information specific to software

Core
- Information used across all profiles
SPDX 3.0 Increases Modularity

SPDX 3.0 (Core Model + Software Profile + ~Security Profile + Licensing Profile ) == SPDX 2.3
Security Profile Overview

Captures security-related metadata specific to a piece of software

- Security profile v3.0 properties and relationships support vulnerability management.
  - **discovery & disclosure**: Communicate the vulnerabilities found by person (auditor/researcher), tool, or organization in a particular piece of software, e.g. CVE
  - **severity**: Communicate severity and rank vulnerabilities in a specific piece of software using industry standardized methods, e.g. CVSS
  - **risk**: Communicate how a vulnerability affects a specific piece of software, e.g., EPSS, VEX
  - **remediation**: Communicate how a vulnerability may be addressed or has already been addressed for a particular piece of software

- Security assurances for software integrity and other secure SDL activities are also supported in SPDX 3.0
# Build Profile Overview

## Use Cases

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Build Provenance</th>
<th>Security</th>
<th>Audit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Providing and verifying build provenance of a software ensures that the provided software is what it claims to be.</td>
<td>Providing the necessary information to respond to supply chain compromises such as a bad build or build tool/configuration.</td>
<td>Checking if certain build tool (e.g., security compilation flag configuration) is enabled in the creation of the software.</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>Providing the necessary information to verify the reproducibility of a build to create more confidence in the built artifact and the build process.</td>
<td>Quality</td>
<td>Safety</td>
</tr>
<tr>
<td></td>
<td>Provide evidence to assert higher confidence that the specified software components in an SBOM is accurate and complete.</td>
<td>Providing information about all the inputs and processes that are used in the creation of the software so they can be audited and verified for safety standards in critical infrastructure and components.</td>
<td></td>
</tr>
</tbody>
</table>
AI Profile Overview

Automating Issue Identification Requires Machine Readable AI BOM

- Many issues exist even when comparing to one standard (EU AI Act) across most of the existing state-of-the-art AI models
- In order to be able to establish compliance to several regulations around the world and identify potential risks associated with the use and adoption of AI software an automated standard, that captures all the required data and is machine readable is pivotal.
Data Profile Overview

A profile that adds on top of the core-software profile to describe the dataset that is used to train or test an AI software. These datasets could also be used for other purposes. Similar, to AI profile, we take special care to ensure that process of forming a dataset is captured. In addition, we also make it a point to capture the provenance and the lineage associated with the dataset. We introduce new fields to do so as provenance and lineage of a dataset has more facets to it compared to traditional software.

Properties specific to Dataset profile
- datasetType
- datasetCollectionProcess
- intendedUse
- datasetSize
- datasetNoise
- dataPreprocessing
- sensor
- knownBias
- sensitivePersonalInformation
- anonymizationMethodUsed
- confidentialityLevel
- datasetUpdateMechanism
- datasetAvailability

Properties borrowed from other profiles
- originatedBy
- downloadLocation
- builtTime
- primaryPurpose
- releaseTime
Licensing Profile Overview

- Classes for License model components
  - includes those previously defined in SPDX 2.3 (and earlier) RDF, as well as SPDX License List fields
  - represents Licenses, License Exceptions, and the data model for license combinations (AND, OR, WITH, +)
  - now with License Additions (custom text for right side of WITH expressions)

- Relationships for Licensing-related metadata
  - conceptually similar to SPDX 2.3
  - better clarity and alignment across Software Artifact types
  - now structured as SPDX Relationships rather than properties
    - Enables one SPDX data creator to define the Software Artifacts, and others to later specify their conclusions as to licenses by reference to the same Software Artifact SPDX elements
SPDX Lite Overview

A subset spec. of SPDX focuses on the **minimum elements for license compliance**. The design policy provides clear information for easy practice, intended for use in the ISO/IEC 5230 and PRF 18974 processes.
Where is the SPDX 3.0 development?

- JSON-LD / RDF
- .xls, .xlsx (spreadsheet) (2.2 →)
- .xml (2.2 →)
- .json (2.2 →)
- .yaml (2.2 →)
- “Lite” .spdx (2.2→)

© 2024 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case No: 23-01614-54
Where is the SPDX 3.0 development?

- JSON-LD / RDF
- .xls, .xlsx (spreadsheet) (2.2 →)
- .xml (2.2 →)
- .json (2.2 →)
- .yaml (2.2 →)
- “Lite” .spdx (2.2 →)

RC1
Where is the SPDX 3.0 development?

- JSON-LD / RDF
- .xls, .xlsx (spreadsheet) (2.2 →)
- .xml (2.2 →)
- .json (2.2 →)
- .yaml (2.2 →)
- “Lite” .spdx (2.2→)
Where is the SPDX 3.0 development?
Beyond SPDX 3.0

- Software as a Service
  - Profile team has been active for about 3 months
  - Tracking the CISA workgroup with participation from Nisha

- Safety
  - We’re learning that we already have most of what we need in 3.0
  - For more info see https://www.linux.com/featured/sboms-supporting-safety-critical-software/

- Hardware
  - Safety Standards expect to know “system” that software is running on
  - Potential participants include RISC-V & ARM core adopters, Chips Alliance Members and Board Manufacturers
SPDX 3.0 Proposed Rollout

Jan/Feb - Profiles added to Model will be considered part of 3.0 ready to be reviewed
  ▪ Minimum will have Security, Build & Licensing (equivalence to 2.3)
  ▪ Serializations in JSON prototyped in Java & Python libraries
  ▪ Need to identify leads for JSON, YAML, RDF, Tag:Value, XML

March - Additional serializations added to libraries.
  ▪ Porting guide from 2.2/2.3 → 3.0 drafted, and available for review
  ▪ Provide examples of how to use new SPDX 3.0 features.
  ▪ Release candidate specification for SPDX 3.0 for other tools feedback

April/December - Release SPDX 3.0 release candidates
  ▪ Gather community feedback, and improve documentation

February - bring to OMG Architecture Board as an RFC once ready to start ISO submission update
  ▪ if no significant eratta after 2 months, start work on updating ISO submission
How to Get Involved - PRs & Issues

- How to Get Involved - PRs & Issues
  - https://github.com/spdx
- Quick start guides
  - https://github.com/spdx/outreach/tree/main/quickstart
- Specification
  - https://github.com/spdx/spdx-spec ← ISO submission format
  - https://github.com/spdx/spdx-3-model ← 3.0 development
  - https://github.com/spdx/spdx-examples
- Tooling
  - https://github.com/spdx/tools-python
  - https://github.com/spdx/tools-golang
  - https://github.com/spdx/tools-java
- License List
  - https://github.com/spdx/license-list-XML
Questions?