Subject: Response to CISA SBOM Minimum Elements Draft
Date: October 1, 2025
Submitted by: The SPDX Project
Contacts:

- Kate Stewart (kstewart@linuxfoundation.org)
- Rose Judge (rose.harber@gmail.com)
- Gary O'Neall (gary@sourceauditor.com)
- Steve Winslow (swinslow@linuxfoundation.org)
- Arthit Suriyawongkul (suriyawa@tcd.ie)

Table of Contents

I. Introduction

System Package Data Exchange (SPDX) is a freely available international open standard (ISO/IEC 5962:2021) capable of representing systems with software components as SBOMs. The following feedback is presented on behalf of the SPDX Project, which engaged in multiple community discussions regarding the CISA SBOM Minimum Element Draft. We believe effective and interoperable standards for SBOMs are essential to improving software supply chain transparency and security. Below are our observations, suggestions, and questions intended to support further refinement and promote adoption of the CISA SBOM Minimum Elements guidance.

II. Comments on Data Fields update

Generally, we suggest adding concrete examples to this suggestion in order to have a bigger impact for those trying to understand the document.

A. SBOM Author

We suggest clarity around what constitutes an "entity" that generates the SBOM. SPDX has a `createdBy` field which must be a type of Agent to record the SBOM Author. SPDX allows an Agent to be a software agent (e.g. a tool), person or organization.

Fields/Relationships to use for SPDX 3.0 to capture this: CreationInfo.createdBy

B. Software Producer

It is unclear if this definition includes the entity responsible for the final distribution of the components, the entity that originally developed the components, or both. In the SPDX Specification, we separate out 2 different properties of a component - one to represent the originator (the entity that created the component) and one to represent the supplier (the entity that is responsible for the final distribution of the component). It is unclear if the Software Producer is the originator or the supplier. Additionally, we have concern around the fact that this field allows for multiple entities. It is our opinion that a software producer field should reference a single supplier of the component. However, if the field continues to allow for multiple entries, the name of this field should be changed to "Software Producers" to reflect the cardinality.

We suggest keeping the original language of "Software Supplier" instead of the updated "Software Producer" terminology which is ambiguous and may reference the software originator *or* the software supplier without providing clarity as to which is which. Generally, we do not consider providing the software originator metadata a common scenario in projects where the supplier takes responsibility for maintaining the component (not the upstream originating open source project). That being said, capturing the originator still has value for other use cases and can be added as a separate field for a component.

Fields/Relationships to use for SPDX 3.0 to capture this: Artifact.suppliedBy & Artifact.originatedBy

C. Component Name

We find the text "*in a human-readable way*" to be too broad. It may be of use to quantify what this means based on the intended audience for the name. For example, a "human-readable name targeting developer audience" may have fewer variants (and tend to look like an identifier) while a "human-readable name targeting end-user audience" may have more variants (as they may be more sensitive to local intellectual property protections / cultural nuances), including names in non-roman script (non-ASCII). The Java version is an example of this – "8" is a product version one can find in marketing materials but "1.8" is a version reported by "java -version".

As the document is written now, there can be multiple Software Producers. If the Component Name is to be assigned by the Software Producer but there are more than one, what happens when the producers don't agree? In this case, there would be multiple component names referring to the same binary. We suggest limiting Software Producer to a single entity and using SPDX external references to provide additional naming characteristics of the component.

What is the recommendation for handling non-English languages and English language synonyms, i.e. when the project name is in a language with a different character script; how would one specify an expected character set? What about left to right encoding?

Fields/Relationships to use for SPDX 3.0 to capture this: Element.name

D. Component Version

There is some concern with the implication of the text "*specify a change in software from a previously identified version*". We recommend the definition to be: "*Identifier that represents the software version associated with the component name, if known.*"

Fields/Relationships to use for SPDX 3.0 to capture this:
Software.Package.packageVersion

E. Software Identifiers

The reference in footnote 9 on page 7 of the CISA SBOM Minimum Elements document is to the SWHID Specification v1.1, while the current version of the specification is v1.2.

Fields/Relationships to use for SPDX 3.0 to capture this: ExternalIdentifierType

F. Component Hash

It is our opinion that this field is mostly useful for individual files, and will not have the intended effect of binary verification. We suggest making it an explicitly optional field (instead of only optional when the SBOM Author does not have access to the original component artifact).

From an open source perspective, it is also crucial that the hash algorithms used for this field be available publicly and free of charge. Otherwise we could see a scenario where an SBOM Producer uses a proprietary hash algorithm and a recipient may be required to pay fees in order to be able to validate their SBOM/SBOM Components.

Fields/Relationships to use for SPDX 3.0 to capture this: Element.verifiedUsing

G. License

We strongly recommend that a license field not be considered part of the "minimum elements" for an SBOM intended primarily for security use cases. While license information is important and useful for other concerns in the software supply chain, users who want to create an SBOM for security vulnerability management purposes should not be required to provide licensing information. Making this field mandatory would place a burden on those SBOM creators whose priority and focus is security, and potentially perpetuate poor licensing information being passed along in the supply chain.

As a project that originally focused on the exchange of license information to facilitate license compliance, the SPDX Project is well aware of the complexity and challenges of obtaining reliable and accurate license information. As a result, the SPDX Project includes a legal sub-team focused on ensuring the SPDX Specification accommodates a variety of potential licensing realities, as well as a robust process for identifying and matching common license texts.

The License field definition and explanation in the proposed 2025 Minimum Elements is brief and lacks clarity which may have unreliable results. For example, the recommendation text "*This field should also include the existence of proprietary license conditions*" is problematic. There is no accepted definition of a "proprietary" license or "license condition". Bilateral agreements for closed-source code are usually confidential, may be lengthy and contain a range of obligations that blur the lines between contractual clauses and license conditions; and may be highly negotiated, thus non-standard. While we are highly supportive of conveying license information in a machine-readable way, this is only possible for standardized and publicly available license texts.

Another example of text that may yield unhelpful results is: "*If the SBOM author is not aware of the license information, then the SBOM author should indicate that license information is unknown.*" Earlier versions of SPDX required explicit statements that license information is unknown, but the community found that such mandatory statements were unhelpful and added unnecessary redundant data to SBOMs. Instead, more modern versions of the specification indicate that an absent licensing field should be interpreted as NOASSERTION. For example, in SPDX 2.3 (section 7.13.1) it states: "*If the Concluded License field is not present in a package, it implies an equivalent meaning to NOASSERTION.*" For the CISA guidance, we suggest adding another line saying something similar to the following: "*Alternatively, the absence of licensing information in an SBOM may be presumed to indicate that the SBOM author is making no assertions about the licenses applicable to the software component.*"

Fields/Relationships to use for SPDX 3.0 to capture this: `hasDeclaredLicense`, `hasConcludedLicense` relationship types. More detail about the distinction between a declared and concluded license can be found in the Licensing Description of the spec.

H. Dependency Relationship

There was consensus among the group that the text "*largely derived from*" is extremely subjective and it is unclear how one would determine this qualification. Furthermore, if there are multiple producers allowed for a software component, how would one denote what percentage (and in what context) of the software component was derived from which producer? Asked another way, how do you correlate which producer is associated with which part of the derivation? Lastly, from a vulnerability and licensing point of view, a component is either derived or not, there is no spectrum of applicability (i.e. a vulnerability cannot only affect 40% of a software package). We recommend amending the text to speak in absolutes by removing the word "largely."

The text "*reflects how a given software component was included in the target*" is ambiguous in the sense that there is no guidance around what is the scope of the term "included"? Is this simply included/not included or do you expect characterization of how the code was incorporated (i.e. statically linked, dynamically linked, documentation, test case of, etc)

Fields/Relationships to use for SPDX 3.0 to capture this: `contains` relationship type

I. Tool Name

The guidance states that multiple tools may be listed in this field but it is unclear how the time sequence for these tools would be interpreted by the SBOM recipient. For example, if something was generated then enriched, how do you know what data comes from which source at which timestamp? We strongly believe that SBOMs are immutable and any changes via enrichment tools performing additional modifications should constitute a separate SBOM instead of denoting multiple tools in this field.

Reference to "*the tool's data sources*" is not always clear or available. For example, multiple SCA tools may feed into a company's software management system that eventually generates an SBOM. Many software distributors, however, may not be tracking which metadata comes from which source and would therefore not be able to communicate any meaningful information about the tool's data sources. Additionally, data sources may change over time – how far back would one go in the data generation process when filling out this field?

Fields/Relationships to use for SPDX 3.0 to capture this: createdUsing property for a Core.Classes.Tool

J. Timestamp

As mentioned in the section above, we strongly believe that SBOMs should be immutable. If changes are made to an SBOM, a new SBOM should be generated with a new timestamp instead of updating the timestamp as the guidance suggests.

If the guidance is recommending or allowing for multiple tools to be used to generate and enrich an SBOM, the timestamp of the change should correlate to the tool used to make the change. A timestamp alone gives no information or context around what changes were made. We fail to see how this would be useful without that context as a timestamp alone would, in meaning, only denote "from what point in history has this SBOM been immutable".

Fields/Relationships to use for SPDX 3.0 to capture this: DateTime

K. Generation Context

We suggest including guidance on how someone would fill out this field if an SBOM has data from multiple contexts (i.e. before/during or during/after). Furthermore, if there is a hash required for software components, this almost exclusively happens after buildtime.

In SPDX 3.0 there is no clear before/during/after field to denote the generation context. Instead, you would use the Build Profile and its properties. Is it CISA's intention that an

SBOM spec explicitly add this field or would the SBOM reader make this determination based on timestamps?

For software written in an interpreted language, what context would an interpreted language be considered as there is no real build process when the code runs – is the executable actually the interpreter?

Fields/Relationships to use for SPDX 3.0 to capture this: Build Profile

III. Comments on Automation Support update

The reference to footnote 10 on page 9 incorrectly references the SPDX project's prior name "Software Package Data Exchange" when its current name should be stated as "System Package Data Exchange". The footnote, however, correctly cites the "System Package Data Exchange" website.

The document says "*agencies should avoid accepting SBOMs for new software generated in deprecated versions of any format to maintain compatibility with SBOM consumption and management tools*". We suggest adding clarity on who exactly determines whether an SBOM standard version is deprecated. Many standards support backwards compatibility but certain versions may be deprecated within companies and organizations while still officially supported in the community. Also, it is common for some tools to only support newer versions of a standard despite older versions still being supported in the community.

IV. Comments on Practices and Processes update

A. Frequency

More clarity is needed around if an updated timestamp (as defined by the Timestamp field) constitutes a revised SBOM or is considered the same SBOM with a new timestamp? Is the guidance suggesting that a new SBOM should only be generated when the software components change and not when any metadata inside the SBOM changes? We are of the opinion that SBOMs are immutable and the latter is true/recommended.

B. Coverage

The term "relevant files" is ambiguous. What is a relevant file? Is it something that the program needs to run or something strictly optional? As an example of an optional file, an image loader could be a vulnerability point but may not necessarily be considered a relevant file. Do we need to include data files? Clarification around what a "non-code" file is would also be helpful.

Additionally, clarity around what exactly constitutes a transitive dependency would be useful (i.e. do you need to provide .o files as a transitive dependency? Is the intention runtime transitive dependencies or build time transitive dependencies or both?).

C. Known Unknowns

SPDX currently does not have a way to declare something as a known unknown but it’s possible to add this as a type of RelationshipCompleteness.

V. Appendix A: Table of Minimum Elements Data Fields

It would be helpful to add namespace qualifiers to the data field names in Appendix A. For example, instead of just “timestamp”, calling it “SBOM timestamp” so it is clear what the field applies to. Additionally, it would be helpful to summarize the minimum fields in a table with basic descriptions and information so readers don’t have to frequently scroll back through the document when trying to meet the minimum qualifications. In the table on page 8 of this document, we have added suggested namespaces to the Data Field Descriptions.

We suggest, at a minimum, the summary table includes the field name, description and cardinality of the field. Ideally, it would also include the SPDX and CycloneDX field mappings.

Data Field Description	**Description**	**Multiple entries?**	**SPDX Specification Mapping**	**CycloneDX Specification Mapping**
SBOM Author				
SBOM Generation Tool Names		Y		
SBOM Generation Context				
SBOM Timestamp				
Software Producers		Y		
Software Identifiers		Y		
Component Names		Y		
Component Version				
Component Hash				
Component Licenses		Y		
Component Dependency Relationship				

VI. General Comments

- The SBOM Minimum Elements document should include a concrete mapping to SPDX and CycloneDX fields for each of the required categories. This will make it much easier for readers to interpret if they are already familiar with one of the specifications, or help them translate their current SBOMs to one of the specifications if they are not already familiar.
- The Minimum Elements document should be framed into “optional” and “must” rather than the English language, which is ambiguous in some places.

VII. Conclusion

We appreciate CISA’s initiative and leadership in advancing software supply chain transparency through this draft SBOM Minimum Elements guidance. Establishing clear and practical guidance is essential to driving adoption across industries and ensuring that SBOMs deliver meaningful value. The SPDX community strongly supports efforts to harmonize standards, promote implementation consistency, and lower barriers for organizations of all sizes to adopt SBOM workflows.

We thank CISA for the opportunity to provide feedback and look forward to continued collaboration to strengthen the security and resilience of the software ecosystem.