2025 Minimum Elements for a Software Bill of Materials (SBOM)

Public Comment Draft
August 2025
Cybersecurity and Infrastructure Security Agency
Department of Homeland Security

About this draft: This is a pre-decisional draft for public comment. It does not represent the final position of the U.S. Government and is continuing to undergo updates as feedback is received.

Executive Summary

In 2021, the National Telecommunications and Information Administration (NTIA) published *The Minimum Elements for a Software Bill of Materials* (2021 NTIA SBOM Minimum Elements), recognizing the importance of software supply chain transparency for national cybersecurity. A software bill of materials (SBOM) is a formal record containing the details and supply chain relationships of the various components used in building the software. SBOMs provide those who produce, choose, and operate software with information that enhances their understanding of the software supply chain. That information can be transformed into insights to drive risk management decisions, including addressing known and newly discovered vulnerabilities and risks.

SBOM implementation practices have advanced significantly since the 2021 document was published. Increased adoption and implementation from stakeholders across the software ecosystem has uncovered new use cases and applications for SBOM data. SBOM tooling, in response to the increased adoption and implementation of SBOM, has matured far beyond the capabilities and functionalities of tools available in 2021. These advancements allow organizations requesting SBOMs to demand more information about their software components and supply chain than they could have in 2021.

As the U.S. lead cybersecurity agency, the Cybersecurity and Infrastructure Security Agency (CISA) is publishing this document to update and clarify the elements of the 2021 document; it introduces new elements to reflect current SBOM needs and capabilities, while preserving the core principles of the 2021 document. Automation, for example, remains critical for driving security at scale. This document aims to guide SBOM implementation for U.S. federal departments and agencies. It can also be used by many other organizations. The updates and additions included in this document will better position Federal Government agencies and other SBOM consumers to address a range of use cases, understand the generation process, and improve data quality.

Additions introduced in this document (*Component Hash*, *License*, *Tool Name*, and *Generation Context*) provide information for improved risk-informed decisions regarding software security. This document also updates details around some elements (*SBOM Author, Software Producer*, *Component Version, Software Identifiers, Coverage,* and *Accommodation of Updates to SBOM Data*) to clarify the data expected in SBOMs for more uniform implementations. Other elements (*Component Name*, *Timestamp, Dependency Relationship*, *Automation Support, Frequency, Known Unknowns,* and *Distribution and Delivery*) update the earlier version of the elements to improve information quality and align with technological developments. This version removes the *Access Controls* element and incorporates access control considerations in *Distribution and Delivery*.

Discussions across the software ecosystem have begun to explore possible additional elements for some types of software, namely software-as-a-service in cloud environments and artificial intelligence (AI) software systems. The minimum elements discussed in this document apply to all software. Additional data fields, practices, and processes may be beneficial for these different types of software. Any effort to improve software transparency, regardless of software type, should begin with the application of the SBOM minimum elements.

Table of Contents

Introduction

A software bill of materials (SBOM) has emerged as a key building block in software security and software supply chain risk management. An SBOM is a nested inventory, a list of ingredients that make up software applications and systems. The Minimum Elements for a Software Bill of Materials (SBOM) (2021 NTIA SBOM Minimum Elements) published by the National Telecommunications and Information Administration (NTIA) defined expectations for SBOM implementation.[1] This document reflected the technological maturity of SBOM tooling and the software community's awareness of SBOM in 2021. Since then, organizations, experts, and practitioners across the software ecosystem have become more familiar with SBOM and the applications of SBOM data in software and supply chain security. As the number of software producers generating SBOMs and the number of software choosers[2] and operators[3] requesting SBOMs increased, SBOM tooling matured as well. The SBOM state of art has advanced significantly since 2021, driven by collaboration among experts and innovations from practitioners. In accordance with the Office of Management and Budget (OMB) memorandum Software Attestation & Supply Chain Security (M-22-18), this document updates the 2021 NTIA SBOM Minimum Elements to reflect the current SBOM maturity of SBOM implementation practices.[4] This document updates the baseline data fields, practices, and processes for SBOMs generated or requested by U.S. agencies.

Why SBOM: Enhancing Transparency and Security

SBOMs are a critical step toward achieving software component transparency, illuminating the software supply chain and better positioning organizations to make risk-informed decisions regarding their software. An SBOM serves as the "ingredients list" for software, arming organizations with data about the make-up of their software. Organizations can then transform that data into insights that can drive actions to mitigate risks that could compromise the security of their software systems.

An effective mechanism for sharing and using software data must be machine-processable and scalable. The SBOM model achieves both by capturing software component data in a machine-processable format and supporting operations that analyze, share, and manage it. SBOM data can be mapped to other data sources such as security advisories or organization-level "approved/not approved" software databases to improve other priority practices (e.g., secure software development, vulnerability management). SBOM will not resolve all software security and supply chain concerns, but it is a necessary step that enables and empowers risk-informed security decision making.

---

[1] National Telecommunications and Information Administration. The Minimum Elements for a Software Bill of Materials (SBOM). July 12, 2021.

[2] A software chooser is the person (or organization) responsible for deciding what software to use. See CISA Open Working Group on SBOM Tooling & Implementation. Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM) Third Edition. September 3, 2024; NTIA Multistakeholder Process on Software Component Transparency. Roles and Benefits for SBOM Across the Supply Chain. November 2019. See CISA Open Working Group on SBOM Tooling & Implementation. Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM) Third Edition. September 3, 2024; NTIA Multistakeholder Process on Software Component Transparency. Roles and Benefits for SBOM Across the Supply Chain. November 2019.

[3] A software operator is the person or organization responsible for operating the software.

[4] Office of Management and Budget. M-22-18. September 14, 2022.

Why Minimum Elements: Driving Security at Scale

The minimum elements specify the baseline technology and practices an SBOM should meet. The volume of software used by agencies is too high for manual processes to effectively assess and mitigate risks. Given the role of software in enabling essential functions and critical infrastructure, software that is not tracked and managed poses a threat to national security. These two conditions emphasize the need for a universal baseline for SBOM data across U.S. agencies. Shared expectations for SBOMs will help agencies apply SBOM data to software and supply chain security practices and will facilitate interagency information sharing.

Beyond the U.S. Government, organizations across the software ecosystem and practitioners around the world have recognized the valuable role of SBOM in increasing software and supply chain transparency and have contributed to the growth of SBOM adoption and advancement of SBOM technological implementation. As SBOM adoption continues to spread across industries and sectors and internationally, the importance of harmonizing expectations for SBOM will only increase.

Scope

The minimum elements outlined in this document can apply to software acquired or developed by agencies and the components of that software, including open-source software, artificial intelligence (AI) software, and software-as-a-service (SaaS). While additional elements may be necessary to gain transparency into the entirety of the software and its components for more complex software systems such as AI or SaaS, an SBOM should still include the minimum elements. The additional elements that may be required for specific types of software are out of scope for this document.

The minimum elements, as described in this document, do not create new federal requirements but rather refine how the Federal Government should generate and request SBOMs. Data management and storage practices are out of scope for this report, as are precise encoding details.

Minimum Elements

The SBOM minimum elements support an evolving approach to software transparency by capturing both the technology and the functional operation. Over time, the capabilities for transparency in the software supply chain will improve, with future efforts incorporating more detail and technical advances.

The three categories of minimum elements are:

- Data Fields
- Automation Support
- Practices and Processes

An SBOM is roughly hierarchical in structure; software consists of components, each of which may have subcomponents. Each subcomponent may also be comprised of subcomponents, and so on. Components (and subcomponents) may be "third party" or "first party." Third party components are from an external source. First party components are from the same producer but identifiable as independent, trackable units of software. Each component should have an SBOM that captures its subcomponents and the hierarchy of its dependencies. The data fields apply to each component and are machine-processable, relying on tools and defined formats.

Data Fields

The core of an SBOM is a consistent, uniform structure that captures and presents information used to understand the components that make up software. Data fields contain baseline information about each component. The goal of these fields is to enable sufficient identification of these components to track them across the software supply chain and map them to other beneficial sources of data, such as vulnerability databases or license databases. Some data fields may present similar data in different contexts across the many components and subcomponents, but their inclusion enables clarity and identification. Appendix B contains the change log for all updates to the minimum elements.

SBOM Author (Major Update)
Definition: The name of the entity that creates the SBOM data for this component.

The *SBOM Author* element contains a string that identifies the entity that generated the SBOM for this component. This field is distinct from the *Software Producer* field, which identifies the producer of the software component. The *SBOM Author* and the *Software Producer* fields may be identical if the same entity produced the software component and generated the SBOM. In cases where an entity that did not produce the software component generates the SBOM for the component in question, the fields will be different.

Software Producer (Major Update)
Definition: The name of an entity that creates, defines, and identifies components.

The *Software Producer* field contains a human-readable string that identifies the entity that produced the software component. "Software producer" refers to the originator or manufacturer of the software component. Organizations can use the *Software Producer* element to learn more about a software component's producer. It can also enable the identification of a point of contact for software security concerns. The *Software Producer* element should allow for multiple entries.

For open source software projects, the software producer remains important. Naming practices may vary depending on how the project develops, distributes, and maintains packages and files. Certain ecosystems provide conventions for naming suppliers through package managers. Otherwise, the *SBOM Author* should use the original project or maintaining organization, when available. If there is no clear indication of a software producer, the *SBOM Author* should explicitly indicate that the component is of unknown provenance to acknowledge the lack of traceability.

This replaces the element of *Supplier Name* in the 2021 NTIA SBOM Minimum Elements. *Supplier Name* has proven ambiguous in practice, particularly around distributors of software. Although reduced by the change to *Producer Name*, some ambiguity will remain until there is a global consensus methodology for identifying entities that create software.

Component Name (Minor Update)
Definition: The name assigned by the *Software Producer* to a software component.

The *Component Name* element identifies the software component in a human-readable way. The software producer determines the name of a software component. This field is distinct from the *Software Identifiers* field. Data formats implementing the *Component Name* element should allow for multiple entries to capture alternate names. The *SBOM Author* should take care to avoid introducing confusion.

Component Version (Major Update)
Definition: Identifier used by the *Software Producer* to specify a change in software from a previously identified version.

The *Component Version* element captures the software version and allows an organization to identify a specific code delivery. If the *Software Producer* does not provide a version, then the *SBOM Author* may substitute the creation date of the file.

Software Identifiers (Major Update)
Definition: Identifier(s) used to identify a component or serve as a look-up key for relevant databases.

The *Software Identifiers* element contains at least one software identifier associated with the software component.[5] Machine-processable, unique software identifiers support automated analysis. Common software identifiers such as Common Platform Enumeration (CPE)[6] and Package Uniform Resource Locators (purl)[7] are preferred. This field may also include universally unique identifiers (UUID), organization-specific identifiers, commit hashes, and intrinsic identifiers such as OmniBOR[8] and SWHID.[9] If there are multiple software identifiers, either in the same software identifier format or different one(s), the *SBOM Author* should include all of them.

Component Hash (New)
Definition: The cryptographic value generated from taking the hash of the software component.

---

[5] Cybersecurity and Infrastructure Security Agency. Software Identification Ecosystem Option Analysis. October 26, 2023.

[6] National Institute of Standards and Technology. Official Common Platform Enumeration (CPE) Dictionary.

[7] GitHub. package-url.

[8] OmniBOR.

[9] SWHID. The SWHID Specification Version 1.1.

The *Component Hash* element contains a hash value and identifies the hash algorithm used or indicates that the *SBOM Author* does not have access to the original component artifact. If the *SBOM Author* does not have access to the original component artifact, then the *SBOM Author* should not include a *Component Hash*.

License (New)
Definition: The license(s) under which the software component is made available.

The *License* element captures the license information under which the software component is available to be used. When possible, the SBOM author should convey this information in a machine-processable fashion. This field should also include the existence of proprietary license conditions. If the SBOM author is not aware of the license information, then the SBOM author should indicate that license information is unknown.

Dependency Relationship (Major Update)
Definition: The relationship between two software components, specifically noting that Software X includes Component Y or that Component A is largely derived from Component B.

The *Dependency Relationship* element reflects how a given software component was included in the target software. The inclusion relationship ("includes" or "included in") supports the capability to build a dependency graph. In addition to inclusion, the dependency relationship should reflect how a given component may be largely derived from another piece of software, or that it is a descendant of another piece of software. This allows an SBOM to explicitly document backported or forked software.

Tool Name (New)
Definition: The name of the tool used by the *SBOM Author* to generate the SBOM.

The *Tool Name* element contains a string that identifies the software tool(s) that the Author of SBOM Data used to generate the SBOM and the tool's data sources (for example, hybrid or enriched). If the *SBOM Author* used more than one tool to generate, enrich, or augment the SBOM data for the specific component, the *SBOM Author* should list all tools used.

Timestamp (Minor Update)
Definition: Record of the date and time of the most recent update to the SBOM data.

The *Timestamp* field identifies when the *SBOM Author* last changed the SBOM data, either manually or using a software tool. If the *SBOM Author* does not make any changes to the SBOM after generating it, then the *SBOM Author* should input the time and date of generation. The content of this field should adhere to ISO 8601.

Generation Context (New)
Definition: The relative software lifecycle phase and data available at the time the Software Producer generated the SBOM (before build, during build, after build).

The *Generation Context* element offers insight into the circumstances, data, and visibility of the SBOM generation process. The three contexts, before build, during build, and after build, are relevant for organizations receiving SBOMs. Pre-build or source code SBOMs are based on data from repositories and other component sets. An SBOM Producer may generate a build or build-time SBOM as part of the creation of a releasable artifact (for example, executable or package), recording the exact components that contributed to the artifact. Binary analysis tools can generate an analyzed or post-build SBOM. This element provides context for the type of data from which the *SBOM Author* generated the SBOM.

Automation Support

Automation support is critical for managing software component data at scale, particularly across organizational boundaries. SBOM implementations must be compatible with each other to support automation due to the volume of data, diverse use cases, and variety of tools. The two data formats currently widely used by software ecosystem stakeholders to generate and consume SBOMs are:

- Software Package Data eXchange (SPDX)[10]
- CycloneDX[11]

These data formats are a product of open, international processes and are both machine-processable and human-readable.

Software producers or SBOM authors may choose their preferred SBOM generation format based on factors that may be specific to organizations, industries, or sectors. Organizations should accept any widely used, interoperable, and machine-processable SBOM format. However, agencies should avoid accepting SBOMs for new software generated in deprecated versions of any format to maintain compatibility with SBOM consumption and management tools.

Minimum support for automation means supporting all data formats that are widely used, open source, and compatible with existing data formats. Supported data formats should be reassessed regularly. If a format is broadly found to be incompatible, no longer maintained, or ineffective for SBOM use cases, it should be removed from automation support. This approach accomplishes the goal of building on existing tools for ease of adoption while supporting future evolution and extensibility.

Practices and Processes

An SBOM is more than a structured set of data. An organization's practices and processes for SBOM use should integrate SBOMs into the software development life cycle. An organization should explicitly address these elements in any policy, contract, or arrangement to ask for or provide SBOMs. Some of these elements, such as frequency, are straightforward. Others, such as access, involve multiple practices, building on both existing approaches and novel concepts.

---

[10] Linux Foundation. System Package Data Exchange (SPDX).

[11] OWASP Foundation. CycloneDX.

Frequency (Minor Update)
Each software version or update should have an associated SBOM. When a software producer issues a new build or release, they (or the SBOM author) should also generate a new SBOM to reflect the changes. This includes software builds that integrate updated components or dependencies. When a software producer (or SBOM author) discovers new details about the underlying components or needs to correct an error in the existing SBOM data, they (or the SBOM author) should issue a revised SBOM.

Coverage (Major Update)
An SBOM should include information for all components that make up the target software, including transitive dependencies. There is no minimum depth. If there are multiple instances of a software component with differences in metadata, each instance must be listed separately with the appropriate dependency relationship. An SBOM need not include non-code files; however, it may include security-relevant files such as configuration files. SBOMs should provide a comprehensive listing of the components to facilitate recipients' risk-based decisions. For instance, from a vulnerability management perspective, the recipient of an SBOM should be able to conclude that a newly reported vulnerability does not affect them if the component associated with the vulnerability is not listed in the SBOM.

Known Unknowns (Major Update)
If an SBOM does not list all dependencies, the SBOM author must explicitly identify Known Unknowns. This practice draws a clear distinction between components with no further dependencies and components for which the list of dependencies is incomplete. The default interpretation of the SBOM should be that the data is incomplete. The SBOM author should further distinguish between dependency information that is unknown to the SBOM author and that which is purposefully redacted. The SBOM author should have a process for recipients to ask about any redacted, security-related information. Organizations may consider an SBOM incomplete if essential dependencies are redacted. SBOM authors should indicate unknown or redacted components in a way that is distinguishable from unknown values for specific elements.

Distribution and Delivery (Minor Update)
SBOMs should be available promptly to those who need them. Access controls may limit the sharing of SBOM data with outside parties but should not prevent information sharing between agencies or restrict agencies from integrating SBOM data into trusted security tools. There are multiple ways of sharing SBOM data. For example, an SBOM can accompany installation. Alternatively, an SBOM can be accessible through a version-specific URL, an application programming interface (API) to a database, or a public repository.

Accommodation of Updates to SBOM Data (Major Update)
Organizations should accommodate updates, including those to correct errors to SBOM data. SBOM authors should correct errors promptly. Agencies may consider errors, whether stemming from SBOM author practices or selection of inadequate tools, in agency risk management decisions.

Discussion

As SBOM and SBOM tooling continue to mature, four key areas warrant further consideration and potential guidance: cloud and SaaS, AI software, validation, and correlation with security advisories. Cloud and artificial intelligence (AI) software, two types of software, may warrant additional SBOM elements. Validation and correlation with security advisories present both valuable opportunities and challenges for all software. These topics are areas of ongoing exploration and future work.

SBOM for SaaS in Cloud Environments

The shared responsibility between SaaS producers and SaaS operators for risk mitigation and the frequent changes to SaaS complicate the translation of the SBOM model from on-premises software to SaaS. Both SaaS producers and SaaS operators play a role in administering and securing the SaaS. This shared responsibility may limit some SBOM use cases. The frequency of SBOM delivery for SaaS may also need further specification. The nature of cloud software and modern development practices results in high frequency changes to the software. Delivering and receiving SBOMs at such high frequency may overwhelm both the SaaS producer and the SaaS operator. Distribution mechanisms such as APIs can accommodate rapid code changes in continuous integration and continuous delivery (CI/CD) processes by capturing snapshots.

Acknowledging these challenges in applying SBOM to cloud and SaaS use cases does not negate the benefits of SBOM for cloud and SaaS software or suggest that achieving software component transparency for is not feasible. Further specifications, including potential additional elements that should be included in an SBOM for SaaS/Cloud software, may warrant additional exploration.

SBOM for AI Software Systems

AI software systems, like cloud and SaaS environments, have additional components that may not be captured by the minimum elements. As AI is becoming increasingly embedded in software systems and potentially playing a role in critical infrastructure, organizations should retain and improve visibility into the components of the system to appropriately manage associated risks. Since AI is software, the minimum elements will illuminate some components that agencies should be aware of, but there may be data that is not captured by the minimum elements that would provide more effective visibility. There are ongoing discussions about SBOM for AI use cases and possible data fields that AI users can request. However, this document does not introduce additional elements for SBOMs for AI systems.

Validation

SBOMs, like other security data, may warrant different levels of trust. An SBOM consumer may be concerned about verifying the source of the SBOM data and confirming that it was not altered. In the software world, integrity and authenticity are most often supported through signatures and public key infrastructure. SBOM formats have begun to implement some signature techniques. Organizations managing SBOM information should use existing software signing infrastructure, tools, and key management.[12] Agencies may seek to confirm the accuracy, coverage, and completeness of SBOM data by accessing sources such as open-source software repositories or using binary analysis tools to detect components not listed in the SBOM.

Correlating SBOM with Security Advisories

Security advisories, such as Vulnerability Exploitability eXchange (VEX) and the Common Security Advisory Framework (CSAF) [13] documents provide additional security information about relevant software components. Security advisories communicate information about the vulnerability status of a software component. A software producer can use security advisories to notify its customers that one of their software components is vulnerable and that they are working on a fix (or whichever vulnerability status is most appropriate). The vulnerability status updates not only highlight security issues but also help avoid unnecessary effort if a trusted source deems that potential risk does not translate into actual risk.

This security information is particularly impactful for vulnerability management. When new vulnerability information is shared, an organization with SBOMs can verify if vulnerable components are listed in their SBOMs, significantly improving response time compared to organizations that did not have SBOMs for their software. Security advisories like VEX and CSAF can further improve the efficiency of vulnerability management processes by narrowing the scope of at-risk software components.

Conclusion

As new use cases emerge and technology evolves, SBOM minimum elements should evolve to continue to provide transparency into software components. An SBOM alone is data about software components. Analysis of SBOMs transforms data into insights about associated risks. Agencies can then use those insights to drive security decisions about their software systems. Vulnerability management tools that can ingest SBOMs, analyze the data, and map it to other data sources will enable agencies to leverage data, intelligence, and actions driven by SBOMs.

---

[12] Elaine Barker, National Institute of Standards and Technology. Special Publication 800-57. Recommendation for Key Management: Part 1 –General, Revision 5. May 2020.

[13] OASIS. Common Security Advisory Framework.

The Cybersecurity and Infrastructure Security Agency (CISA) plays a key role in advancing and refining SBOM adoption and implementation, but much of the progress in maturing SBOM has been driven by the SBOM community and stakeholders across the software ecosystem. The experiences and expertise of each of the members of the SBOM community have been invaluable in shedding light on SBOM adoption and implementation challenges, sharing lessons learned across industries, and refining technological aspects of SBOMs. CISA will continue to leverage its position as the lead U.S. cybersecurity agency to facilitate information sharing and bring together experts across the community to identify and promote solutions to software supply chain challenges.

Appendix A: Table of Minimum Elements Data Fields

Data Field	Description
SBOM Author	The name of the entity that creates the SBOM data for this component.
Software Producer	The name of an entity that creates, defines, and identifies components.
Component Name	The name assigned by the *Software Producer* to a software component.
Component Version	Identifier used by the *Software Producer* to specify a change in software from a previously identified version.
Software Identifiers	Identifier(s) used to identify a component or serve as a look-up key for relevant databases.
Component Hash	The cryptographic value generated from taking the hash of the software component.
License	The license(s) under which the software component is made available.
Dependency Relationship	The relationship between two software components, specifically noting that Software X includes Component Y or that Component A is largely derived from Component B.
Tool Name	The name of the tool used by the *SBOM Author* to generate the SBOM.
Timestamp	Record of the date and time of the most recent update to the SBOM data
Generation Context	The relative software lifecycle phase and data available at the time the *Software Producer* generated the SBOM (before build, during build, after build).

Appendix B: Summary of Minimum Elements Changes

This section summarizes the changes between this document and the 2021 NTIA SBOM Minimum Elements. A change is reported if it significantly impacts implementation. An example of a reported change is if a field name (such as *Supplier Name*) has been updated (such as to *Software Producer*). An example of an unreported change is if another field's description has been updated to cross-reference the new field name.

Supplier Name (Major Update)

Change(s): Replace *Supplier Name* element with *Software Producer* element.

Explanation: The term "software producer" better aligns with terms commonly used in SBOM discussions. The term further clarifies that the field refers to the entity that originated the software. Establishing a fallback designation for an unknown software producer (indicating unknown provenance) helps ensure transparency and highlights potential risks in the software supply chain.

Component Name (Minor Update)

Change(s): Allow for multiple entries.

Explanation: Being able to add multiple entries better enables organizations to map the data to the appropriate software component.

Version of the Component (Major Update)

Change(s): Replace *Version of the Component* element with *Component Version* element. Specify that if the software producer does not provide a version, then the SBOM author may substitute the creation date of the file.

Explanation: Being able to associate software component data with a specific version is important for processes such as vulnerability management. The clarification to use the file creation date provides some version tracking even if the software producer does not provide a version.

Other Identifiers (Major Update)

Change(s): Replace *Other Unique Identifiers* element with *Software Identifiers* element. Specify that at least one software identifier must be included and add reference to OmniBOR.

Explanation: While there is not yet a universal solution to the problem of software identification, there has been progress. The changes reflect both the maturity of the discussion around software identification and the maturity of the software identifier formats themselves.

Component Hash (New)

Change(s): Add *Component Hash* element.

Explanation: Hashes play a significant role in existing software and supply chain security processes.

License (New)

Change(s): Add *License* element.

Explanation: A software component's license may affect an agency's software system security. Improper use of licensed software components could trigger enforcement actions that could affect a software producers' ability to deliver on security support commitments. License and re-license information about a software component should inform an agency's risk prioritization for software, as changes in license or inadequate license information may impact the agency's ability to rely on the software to perform its functions, or the producer to support the software.

Dependency Relationship (Major Update)

Change(s): Require pedigree for all backported and forked software.

Explanation: The addition reflects developments in the understanding of necessary information for dependency relationships.

Author of SBOM Data (Major Update)

Change(s): Replace *Author of SBOM Data* element with *SBOM Author* element.

Explanation: The term "SBOM author" is common in SBOM-related discussions and more accurately captures the role of the entity generating the SBOM.

Tool Name (New)

Change(s): Add *Tool Name* element.

Explanation: The addition of this element provides information regarding the manner in which the SBOM author generates the SBOM.

Timestamp (Minor Update)

Change(s): Clarify that entries should adhere to ISO 8601.

Explanation: Following ISO 8601 for entries better supports automation and interoperability.

Generation Context (New)

Change(s): Add *Generation Context* element.

Explanation: Provide more information regarding the source of the data represented in the SBOM. SBOM data may vary depending on the lifecycle phase of the software during which the SBOM author generates the SBOM. This additional context better informs organizations for assessing software risk.

Automation Support (Minor Update)

Change(s): Remove SWID from list of data formats.

Explanation: SWID tags are not a widely used SBOM data format for which multiple tools exist.

Frequency (Minor Update)

Change(s): Rewrite the element to use updated terminology.

Explanation: The update clarifies the *Frequency* element and reflects current terminology but leaves the intent of the element unchanged.

Depth (Major Update)

Change(s): Replace *Depth* element with *Coverage* element. Define *Coverage* as including horizontal breadth (in addition to vertical breadth) of software component information.

Explanation: The *Depth* element was previously defined as capturing software component information for top-level dependencies only. This definition reflected the capabilities of SBOM tooling at the time rather than the depth of information needed to make informed security decisions. Rather than define the degree of depth for dependency information required, the *Coverage* element includes horizontal as well as vertical breadth of software component information. The maturation of SBOM tooling since 2021 makes specifying this breadth of software component information feasible.

Known Unknowns (Major Update)

Change(s): Separate information that is known but purposefully withheld from that which is unknown to the *SBOM Author*.

Explanation: As SBOM implementation has increased, organizations have raised concerns regarding the ambiguity of Known Unknowns. Some software producers wanted to be able to clarify to their customers that information identified as Known Unknown was known to them to signal that they are aware of what is in their software. Some organizations that received SBOMs from upstream suppliers wanted a way to clarify what information their suppliers were unaware of and what information they were unwilling to share. The discussion of known, redacted information should reduce the ambiguity of Known Unknowns and share more information with downstream consumers regarding why some information may not be provided.

Distribution and Delivery (Minor Update)

Change(s): Simplify and clarify *Distribution and Delivery* element.

Explanation: Given the maturity of SBOM sharing practices, a more concise definition is appropriate.

Access Control (Removed)

Change(s): Remove *Access Control* element from the SBOM minimum elements.

Explanation: The development in SBOM practices since 2021 has made the *Access Control* element no longer necessary as a stand-alone requirement. Access controls are instead incorporated into the *Distribution and Delivery* element.

Accommodation of Mistakes (Major Update)

Change(s): Replace *Accommodation of Mistakes* element with *Accommodation of Updates to SBOM Data* element. Refine the definition for what types of changes to SBOM data should be accommodated.

Explanation: The relative immaturity of SBOM adoption and implementation in 2021 made explicitly accommodating mistakes necessary. Technological developments since 2021 have significantly improved SBOM data quality, such that recipients can expect SBOM data to be accurate. Changes to SBOM data should largely be to bring information up-to-date rather than correct inaccurate information.