Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)

Third Edition

Tooling and Implementation Working Group hosted by the Cybersecurity and Infrastructure Security Agency (CISA)

September 3, 2024

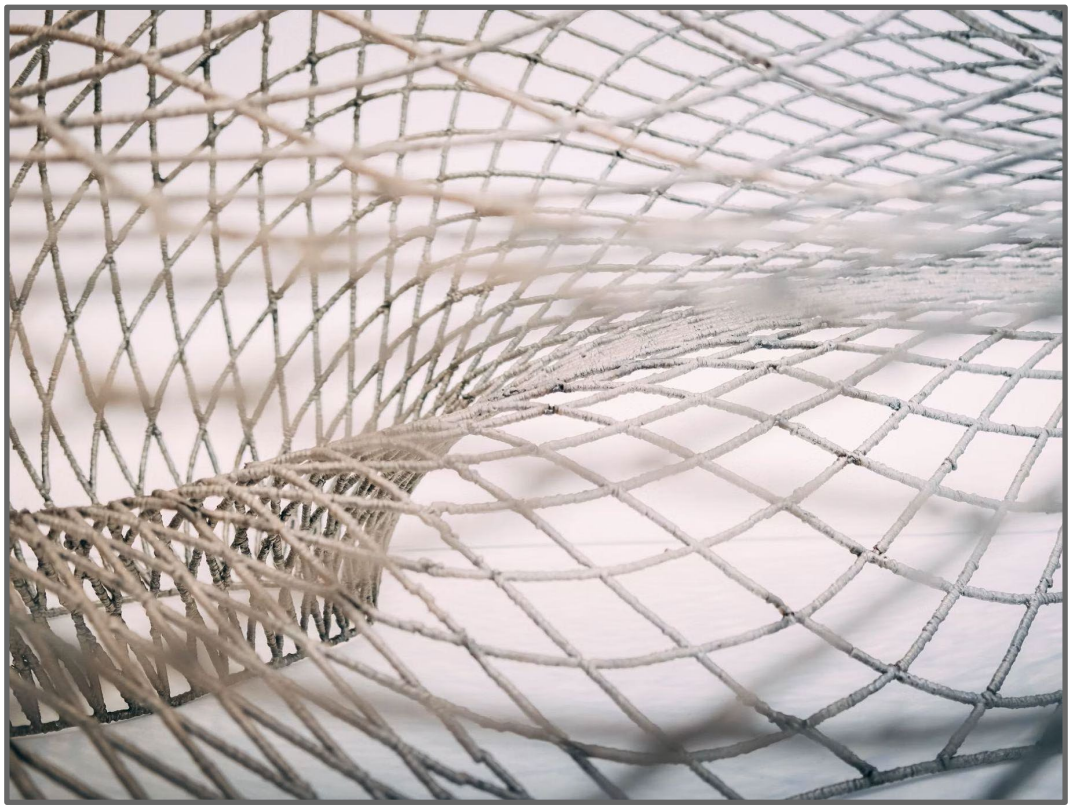

Photo by Luke van Zyl on Unsplash

Table of Contents

About This Document

The first edition of this document[1] was published in 2019 as part of the Phase I series of reports from the National Telecommunications and Information Administration (NTIA) Software Component Transparency multistakeholder process.[2] The concept and implementation of the Software Bill of Materials (SBOM), introduced in that edition, served as the foundation for subsequent work that further matured SBOM.

The second edition[3] updates published in 2021 focused on specific topics rather than a comprehensive revision of the entire document. The updates were based on insights from the Framing group, a workstream under the NTIA multistakeholder process, as well as feedback from other groups within the NTIA Software Component Transparency Multistakeholder Process and the broader SBOM community.

This document, the third edition, further defines and clarifies SBOM Attributes from the 2021 "Framing Software Component Transparency" document, offering descriptions of the minimum expected, recommended practices, and aspirational goal for each Attribute. The work reflected in this document is a product of extensive discussion in the SBOM Tooling and Implementation Working Group, a Cybersecurity and Infrastructure Security Agency (CISA) community-driven workstream, and feedback from across the software community.

This document, "Framing Software Component Transparency," is distinct from the "Minimum Elements for a Software Bill of Materials,"[4] also published by the NTIA in 2021 (SBOM Minimum Elements Document). The SBOM Minimum Elements Document was called for by Executive Order 14028 and was drafted by NTIA as an official government publication. The SBOM Minimum Elements Document establishes the U.S. Government's minimum requirements for an SBOM. CISA has the authority to update the SBOM Minimum Elements Document to further clarify U.S. Government expectations under the Office of Management and Budget (OMB) Memo 22-18.

The appendices listed below provide additional context and correspond to the content within.

[1] NTIA Open Working Group on SBOM Framing. Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM). November 12, 2019.
[2] This document was drafted by the SBOM Tooling and Implementation Working Group, a community-driven workstream. For more information see CISA's website.
[3] NTIA Open Working Group on SBOM Framing. Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM). October 21, 2021.
[4] NTIA. The Minimum Elements for a Software Bill of Materials (SBOM). July 12, 2021.

Appendix C - Acknowledgements to contributing members of the SBOM Community

1 Problem Statement

Modern software systems involve increasingly complex and dynamic software Supply Chains. Unlike many industries of physical goods, the software Supply Chain has historically not been required to provide transparency into the composition of software systems. This lack of visibility has contributed to cybersecurity and Supply Chain risks and increases the costs of software development, procurement, operations, and maintenance. In our increasingly interconnected world, risk and cost impact not only individuals and organizations, but also collective goods (e.g., public safety and national security).

Software Supply Chain Transparency can reduce risks and overall costs by:

- Identifying Components that may impact a Primary Component to enable an organization's analysis of risk
- Enhancing vulnerability management and incident response processes
- Reducing unplanned and unproductive work due to complex Supply Chains
- Reducing duplication of effort through data standardization across multiple sectors
- Facilitating the identification of suspicious or counterfeit software Components
- Improving resilience by encouraging stakeholder collaboration and enabling collective defense against common threats
- Improving accountability of secure software development practices via transparency

In an effort to improve software Supply Chain Transparency, NTIA initially convened a multistakeholder process, called the Framing Working Group[5]. CISA adopted the work as part of the SBOM Tooling and Implementation Working Group. This document is the graduated output from these community-driven working groups.

1.1 Goals

To achieve greater software Supply Chain Transparency, this document describes an SBOM meant for software Component information sharing that can be universally applied across the software ecosystem. This document addresses the creation and sharing of SBOMs, the roles of participants, and the integration of SBOMs with all Supply Chains.

To scale this model globally, it is necessary to address the difficult problem of universally identifying and defining certain aspects of software Components by:

(1) Declaring a required minimum set of Baseline Attributes necessary to identify Components with sufficient relative uniqueness; (2) identifying supplemental, optional Attributes and external elements beyond the baseline set to serve a variety of SBOM applications; and (3) enabling correlation of SBOMs with external sources for relevant analysis.

[5] NTIA Software Component Transparency Effort. April 28, 2021.

2 What is an SBOM?

An SBOM is a formal, machine-readable inventory of software Components and Dependencies, information about those Components, and their relationships. An SBOM's inventory should be as comprehensive as possible and should explicitly state where relationships cannot be articulated. SBOMs may include open source or commercially licensed software and can be widely available or access-restricted to protect proprietary or sensitive information.

Many modern software development processes support the automated generation of SBOMs throughout the software development lifecycle. However, the software Supply Chain still utilizes older systems that may require manual methods to generate an SBOM.

As documented in "Types of Software Bill of Materials (SBOM)."[6] different types of SBOMs provide specific information about the design, the source code, the built software, or the deployed software. These SBOMs would naturally be created at different points of the software lifecycle. In some cases, it is necessary to analyze finished software artifacts with heuristics to produce an SBOM.

An SBOM is populated with the Baseline Attributes for its listed Components. Gathering and declaring Component Baseline Attributes enables two goals: the unique identification of individual Components and the monitoring and managing risk of the software being distributed. The amount and type of information included in an SBOM may vary depending on the needs of the specified Consumer(s) within an individual industry or sector. For instance, the license and copyright holder may not be shared as widely as the Attributes needed for unique Component identification.

This document establishes a minimum expectation for creating a baseline SBOM that outlines the minimum amount of information required to support basic and essential features. In addition to the minimum expectation for each Baseline Attribute, this document also incorporates two more levels of data maturity, namely recommended practice and aspirational goal. These additions are to encourage those who have achieved the minimum required to evolve their SBOM content in maturity and quality.

Defining Baseline Attributes (Section 2.2) and processes (Section 3) allows for rapid adoption by a variety of stakeholders which can then be further evolved over time. This is one of the major drivers for establishing a basic set of information as a starting point rather than initially requiring a more robust set of Attributes that may be more resource-intensive to collect and maintain. Beyond the minimum baseline SBOM, additional information may be required as further development and practices mature in different sectors.

Structured data formats and exchange protocols are another key characteristic of a functional SBOM because they enable machine-readability and automation. Large SBOM Consumer organizations will need to collate and manage large amounts of data from different Suppliers.

[6] CISA Open Working Group on SBOM Tooling and Implementation. Types of Software Bill of Materials (SBOM) Documents. 2023.

Therefore, a machine-readable format is critical to support data Consumers to manage this for efficiency and expandability. Choosing a specific data format is an important part of this functionality. Another important aspect is the universal naming of Components. Without a specific Component naming identification scheme, it would be nearly impossible to identify, track, and manage Components that are named in an *ad hoc* fashion.

SBOMs do not provide significant value as independent entities, completely isolated from other data sources, but are a foundational element for the automation of other activities. For example, the use of SBOMs in vulnerability management requires a catalog of known vulnerabilities (e.g., Common Vulnerabilities and Exposures [CVE][7]), associations of vulnerabilities to Components (e.g., the use of Common Platform Enumeration [CPE][8]) in the U.S. National Vulnerability Database [NVD][9] or Common Security Advisory Framework [CSAF][10]) advisories and a means by which to convey the exploitability or exposure of a vulnerability at different points along Supply Chains.

SBOMs can also enhance the critical task of software inventory management in many ways. License management is a significant and difficult compliance task for many of the same reasons as cybersecurity and the increased transparency offered by SBOMs aids in that also. The use of SBOM for license management requires that licenses and their restrictions are mapped to Components.

2.1 SBOM Elements

Initially, participants in the NTIA Software Component Transparency Multistakeholder Process reviewed existing software identification formats, considered feedback from the various proof of concept exercises, and thoroughly debated and questioned which elements would be necessary to create a scalable and functional SBOM system. Many of the answers depended on the desired use cases that can be built on top of a sufficient quantity and quality of baseline SBOM data. Since the initial release of this document, implementation of the SBOM elements and evolution of tooling have grown and illuminated areas of this document where the quantity and quality of baseline data can be clarified. Systematically and consistently defining and identifying software Components and their relationships enables the desired use cases to function at scale. As a minimum, the Baseline Attributes are required. Supplemental elements and Attributes can be included to enable SBOM use cases identified later in this document.

2.2 Baseline Attributes

The primary purpose of an SBOM is to uniquely and unambiguously identify software Components and their relationships to one another. Therefore, one necessary element of an SBOM system is a set of Baseline Attributes that can be used to identify Components and their

[7] The CVE Program.
[8] NIST. Official Common Platform Enumeration (CPE) Dictionary.
[9] NIST. National Vulnerability Database.
[10] Oasis. Common Security Advisory Framework (CSAF).

relationships. An SBOM system that follows the guidance and framing proposed in this document *must* support these Baseline Attributes. An SBOM system or format *may* support supplemental Attributes. Attributes that are unavailable, not applicable, or do not materially contribute to Component identification are discussed in Section 2.3.

The Author Name, Timestamp, and Primary Component (or Root of Dependencies) Attributes provide meta-information about an SBOM; the remaining Attributes apply to Components that are direct or transitive Dependencies of the Primary Component (see the definition of the Component in Appendix B.)

Three Attribute maturity levels describe the evolving content provided in Attribute entries as well as possible approaches to achieve increased maturity. If there are no maturity levels for the Attribute, the instructions presented are the minimum expected. The data maturity levels and cybersecurity tooling automation support for each Attribute are intended to communicate the following guidance:

> **Minimum Expected** - This maturity level describes the minimum data elements for documenting a Primary Component and its Included Components for SBOMs globally.
>
> **Recommended Practice** - This maturity level describes the addition of Attribute data that supplements Component identification as well as practices for creating SBOMs.
>
> **Aspirational Goal** - This maturity level describes areas that creators of SBOMs can consider for documenting dynamic and/or remote Dependencies (see Appendix B for descriptions) that can be uniquely and unambiguously identified in an SBOM.

2.2.1 SBOM Meta-Information

2.2.1.1 Author Name

The Author Name is intended to be the name of the entity (e.g., person or organization but not the tool) that created the SBOM data. Including the Author Name allows the downstream Consumer to understand the context under which the SBOM was created, providing clarity on the origins and reliability of the data. The Author Name Attribute should name as many participants involved in authoring the SBOM data as possible. The tool(s) used to create the SBOM can also be declared to assist in SBOM data consumption. Multiple entries are permitted. In some cases, the Supplier Name (see Section 2.2.2.3) of the Primary Component may not be the Author of the SBOM data. This would indicate the SBOM was not created by the Supplier.

> **Minimum Expected** - An SBOM must list the entity that prompted the creation of the SBOM. These entities can be organizations, project teams, or individuals in the software Supply Chain responsible for the development, deployment, operation and/or support of software systems including individual software developers. The Author Name Attribute should include the name of the legal entity and some form of unique identification (e.g.,

an email address or website) if possible. If no legal entity name is available, attempt to uniquely identify the SBOM creator along with contact information.

Recommended Practice - In addition to listing the entity that prompted the creation of the SBOM, identify tool(s) and version(s) that assisted the Author in the SBOMs creation.

2.2.1.2 Timestamp

The Timestamp is the date and time that the SBOM was produced. As a **minimum expectation**, the Timestamp should be consistent across time zones and locales and use a common international format, such as ISO 8601[11] (e.g., 2024-05-23T13:51:37Z).

2.2.1.3 Type

The Type Attribute provides context for how and why the SBOM was created. As discussed in Section 2 (see footnote 8), different types of SBOMs can be created from different software artifacts. Documenting the SBOM Type may inform the utility and consumption of the SBOM that was created. This Attribute is **optional** and considered an **aspirational goal**.

2.2.1.4 Primary Component (or Root of Dependencies)

The Primary Component, or root of Dependencies, is the subject of the SBOM or the foundational Component being described in the SBOM. The Component Attributes detailed in Section 2.2.2 are also identified for this Component just as they are for the direct and transitive Components. Maturity levels are described in each Attribute section.

A product-level SBOM may identify a Primary Component that references a set of other products. See Guidance: “Assembling a Group of Products for SBOM” for more details.[12]

2.2.2 Component Attributes

After identifying an SBOM’s Primary Component and its Attributes in the SBOM meta-information, the next step in developing an SBOM is to uniquely enumerate top-level Components that a Supplier directly includes in the Primary Component. For each Component in the SBOM, identifying each Attribute should be attempted as indicated in the data maturity levels. For additional insight on uniquely identifying Components and Suppliers in these Attributes, see “Software Identification Challenges and Guidance” for a more detailed examination of Component and Supplier identification.[13]

Additionally, in order to scale effectively, an SBOM needs to capture transitive, or nested, Supply Chain relationships between Components to the extent that these dependency

[11] ISO. ISO 8601: Date and Time Format.
[12] CISA Open Working Group on SBOM Tooling and Implementation. Guidance on Assembling a Group of Products. January 26, 2024.
[13] NTIA Open Working Group on SBOM Framing. Software Identification Challenges and Guidance. March 30, 2021.

relationships are known. Bills of materials for physical Components often describe these relationships as a “Multi-level BOM.”[14]

The content of an SBOM can vary in depth and breadth based on the maturity of the entity creating the SBOM and the creation tools being used. The maturity levels for the depth and breadth of an SBOM are:

> **Minimum Expected** - SBOMs are expected to identify all static, direct Dependencies of the root or primary Component.
>
> **Recommended Practice** - In addition to the direct Dependencies, SBOMs should identify as many levels of subcomponents as possible. Section 2.3.3 provides more instruction for when a direct Component’s upstream or subcomponents are unknown.
>
> **Aspirational Goal** - In addition to the direct Dependencies and subcomponents identified in the SBOM, efforts are made to uniquely and unambiguously identify Dependencies that are dynamic and/or remote.

2.2.2.1 Component Name

The Component Name is defined as the public name for a Component defined by the Originating Supplier of the Component. Component names can convey Supplier names.

As an alternative, Component (and Originating Supplier) Names can also be conveyed using a generic namespace:name construct where the Originating Supplier Name is used as the namespace designator. Formats and tooling need to provide the capability to handle multiple names or aliases.

To illustrate and clarify the concept of Component and Supplier Names using a generic namespace:name construct, consider the following examples:

> Example 1: For a software Component provided by Acme, the Component Name could be Acme:SecurityModule where "Acme" acts as the namespace designator and "SecurityModule" is the name of the Component. This clearly conveys that the Component is supplied by Acme and the Component function.
>
> Example 2: In a document detailing purchasing information, the Component Name might be listed as Acme:Thermostat:2, indicating that the thermostat Component is a version 2 model supplied by Acme.
>
> Example 3: When referring to SBOM data created by a Supplier other than the Primary Component Supplier, the Author Name could be ThirdPartySecurityFirm:SBOM:1.2, where "ThirdPartySecurityFirm" is the namespace indicating the creator of the SBOM, and "SBOM:1.2" specifies the version of the SBOM.

[14] OpenBOM. OpenBoM Fundamentals: All About Multi-Level BOMs. May 16, 2017.

These examples demonstrate how using a namespace:name construct can provide clarity on the Supplier and component relationship, which is especially useful when the SBOM data is authored by entities other than the primary Supplier. It helps downstream Consumers to identify the source and trustworthiness of the Components within the software they are using.

As a **minimum expectation,** the Component name should declare the commonly used public name for the Component.

2.2.2.2 Version

The Version is a supplier-defined identifier that specifies an update change in the software from a previously identified version. This Attribute helps to further identify a Component and should be separate from the Component Name. As there is a wide range of versioning schemes in use, recording what is provided from the Supplier accurately is the primary goal. Semantic versioning is preferred.[15]

If the Component does not have a unique semantic version available to declare, make sure that a cryptographic hash is provided for the Component. Be aware that this will not indicate the relative release of the Component to its predecessors and successors.

As a **minimum expectation,** declare the version string as provided by the Supplier.

2.2.2.3 Supplier Name

Supplier Name is the entity that creates, defines, and identifies a Component. It should be identified carefully as it is a significant contributor to achieving Component identification at scale.[16] It is worthwhile to mention that mergers and acquisitions can impact the Supplier Name for the SBOM being created. The Supplier Name should be identified as the Supplier that is providing the software at the time of the SBOM creation.

As a **minimum expectation,** the Supplier Name should be declared for all Components. However, the Supplier Name for supplied software should be declared differently depending on whether it was incorporated in the Primary Component unmodified or modified from how it was supplied from the upstream Supplier.

- If the supplied software package or Component was *unmodified* when included, the Supplier Name entered is the upstream Supplier's name with the following guidance:
    - For Suppliers of commercially licensed software, enter the Legal Entity name. If the Legal Entity name is not globally unique, consider adding the Supplier's jurisdiction. In the unlikely case that the Legal Entity name is unknown, consider using the name of the software vendor from https://nvd.nist.gov/products/cpe.
    - For Suppliers of open-source software, list the project name. If known, add the host foundation before the project name (e.g., Apache Tomcat). Referencing the

[15] Semantic Versioning 2.0.0.
[16] See Section 5 of NTIA Open Working Group on SBOM Framing. Software Identification Challenges and Guidance. March 30, 2021.

open-source software (OSS) copyright statements can assist in identifying the supplier(s) or creator(s) of the software. For example, the copyright statement for facebook/react[17] identifies the supplier Name as “Meta Platforms, Inc and affiliates.”

    - Although not a recommended practice, if the component’s other attributes uniquely and unambiguously identify the component and the upstream supplier is difficult to identify, either:
        - Enter the domain URL of the software and/or the namespace of the Package URL (PURL).
        - Enter supplier name as unknown.
- If the supplied software package or Component was *modified* by the Primary Component’s Supplier before being incorporated, enter the Primary Component’s Supplier as the Supplier Name. Additionally, enter the Component’s upstream Supplier in an Attribute field communicating its heritage relationship (e.g., heritage or pedigree) are possible Attribute fields. See Section 2.2.2.6.3 for possible methods to declare this relationship. It is necessary to capture the heritage relationship to enable proper vulnerability monitoring of the Component. The Originating Supplier, if different from the upstream Supplier, may be a beneficial Attribute to capture for the Component as well.

2.2.2.4 Unique Identifier

Unique identifiers provide additional information to help uniquely define a Component. An identifier may be unique at a global level or locally unique within a globally unique namespace (e.g., organization). Both may be declared in the SBOM and provide value.

A unique identifier[18] can be generated relative to some globally unique hierarchy, namespace, reference an existing global coordinate system, or be generated from the content using a hashing scheme.

Additionally, some vendors may have proprietary unique identifiers within their globally unique organization namespace. The Component’s cryptographic hash (Section 2.2.2.5) may also effectively function as a unique identifier.

Examples of unique identifiers:

- CPE[19]
- PURL[20]
- Software Identification (SWID) Tags[21]
- Universal Unique Identifier (UUID) (also known as Globally Unique Identifier [GUID])[22]

---

[17] Facebook GitHub Repository. React Library License.
[18] CISA. Software Identification Ecosystem Analysis. October 2023; NTIA Open Working Group on SBOM Framing. Software Identification Challenges and Guidance. March 30, 2021.
[19] NIST. Official Common Platform Enumeration (CPE) Dictionary.
[20] Package-URL GitHub Repository. purl Spec.
[21] NIST. Software Identification (SWID) Tagging. April 24, 2024.
[22] Wikipedia. Universally Unique Identifier.

- Software Heritage ID (SWHID)[23]
- OmniBOR Artifact IDs (formerly known as gitoid namespace relative ids)[24]

The maturity levels for the unique identifier are:

> **Minimum Expected** - at least one unique identifier should be declared for each Component listed in the SBOM. A globally unique identifier is preferred.
>
> **Recommended Practice** - list as many globally unique identifiers as available for the Component.

2.2.2.5 Cryptographic Hash

A cryptographic hash is an intrinsic identifier for a software Component.[25] In addition to hash values, it must be clear how the hash was generated (i.e., the algorithm used and the object being hashed) so that it can be reproduced. It is worth considering SBOM format options that can create a hash of hashes for individual file Components.

It is possible and may be beneficial to provide multiple hashes for a Component or collections of Components. Suppliers and Authors choose how to define Components, which in turn defines the scope of the hash. For example, an SBOM could include a hash for a source Component, a hash for the compiled binary form of that Component, and a hash for a collection of Components.

The cryptographic hash data maturity levels are:

> **Minimum Expected** - Provide a hash for any Component listed in the SBOM for which the hash was provided or sufficient information is available to generate the hash. If sufficient information is not available, indicate as unknown.
>
> Along with the hash, provide the hash algorithm and the Component object being hashed to enable reproducibility. Hash algorithms accepted at this maturity level are MD5, SHA1, and SHA2 families, (including SHA256 and SHA512). Using a secure hash algorithm is recommended. Note that use of MD5 and SHA1 is no longer recommended and will be formally discontinued in 2030.[26]
>
> **Recommended Practice** - Provide at least one hash of the Primary Component at this maturity level. Along with the minimally expected hash, provide the hash algorithm and the Component object being hashed to enable reproducibility.

---

[23] Software Heritage. SoftWare Heritage Persistent Identifiers (SWHIDs). April 30, 2021.
[24] OmniBOR GitHub Repository. OmniBOR Specification.
[25] Roberto Di Cosmo, Morane Gruenpeter, and Stefano Zacchiroli. Identifiers for Digital Objects: The Case of Software Source Code Preservation. October 5, 2018.
[26] NIST. NIST Retires SHA-1 Cryptographic Algorithm. December 15, 2022.

Hash algorithms accepted at this maturity level are those that are cryptographically secure SHA2 family (SHA-256 and higher) for all Components and system Dependencies listed in an SBOM. If less cryptographically secure, hashes need to be included, adding an additional cryptographically secure hash is required.

2.2.2.6 Relationship

The Relationship Attribute describes the association of a Component listed within the SBOM to other Components. Relationships between Components can be quite varied. As discussed for "Dependency" in Appendix B, the Components listed in the SBOM may be static, remote, provided, or dynamic. Considering the Component and the other Components with which it interacts would inform the type of relationship declared for that Component. The data maturity levels for relationships declared in an SBOM are:

Minimum Expected - Relationships and relationship completeness declared for the Primary Component and direct Dependencies.

Recommended Practice - Relationships and relationship completeness declared for all Included Components listed in the SBOM.

Aspirational Goal - Relationships and relationship completeness to as many dynamic and remote Components as possible (e.g., loaded Components or services) are identified.

The following subsections give several types of common relationships to consider declaring in the SBOM.

2.2.2.6.1 Primary Relationship

A relationship type of *primary* is used when a Component is the subject of the SBOM. As discussed in Section 2.2.1.4, the Primary Component defines the subject of the SBOM (e.g., Acme Application in Table 2), including cases where the SBOM only includes one Component (e.g., Carol's Compression Engine in Table 3).

2.2.2.6.2 "Included In" Relationship

The dependency relationship between Components is inherent in the design of the SBOM model. The default relationship type is "includes." This represents the inclusion of or dependency on a separate upstream Component. To simplify the presentation, this document reverses the direction of the relationship to "included in." The choice of direction is not important to the model, as long as one direction is chosen and used consistently. Using the example from Section 2.6, the following statements are equivalent:

1. Acme Application v1.1 "includes" Bob's Browser 2.1.
2. Bob's Browser v2.1 is "included" in Acme Application v1.1.

It is possible to further refine the "included in" relationship, for example, conveying the difference between:

- Directly including, unchanged, an upstream binary Component.
- Including an upstream source code Component, unchanged, by linking or compiling.
- Selecting an upstream source code Component, modifying (forking) it, and then including it by linking or compiling.

2.2.2.6.3 Heritage or Pedigree Relationship

Modifying a Component effectively creates a new Component (e.g., a fork) and the modifier becomes the Supplier for that new Component. It is important in this example to maintain the heritage of the modified Component and convey that it has been modified. For example, SPDX supports GENERATED_FROM and DESCENDANT_OF relationship types, while CycloneDX supports “pedigree” relationships.

2.2.2.6.4 Relationship Completeness

Ideally, every Supplier will create and provide SBOMs for their Components and all Consumers will obtain complete chains of these authoritative SBOMs. For every Component, Author Name (2.2.1.1) will equal Supplier Name (2.2.2.3) and, in this ideal world, there will be complete knowledge of product Components. Until this state is achieved, SBOM authors may want to make non-authoritative claims or assertions about Components for which the authors are not the Suppliers. One expected case is that a Supplier wants to assert their belief about upstream Components for which an authoritative SBOM does not exist.

Relationship completeness can be recorded using a supplemental, optional Attribute. The following four categories cover the range of an author’s knowledge about another Supplier’s Components.

1. **Unknown**. This is the default. There is not yet any claim, knowledge, or assertion about upstream Components. Immediate upstream Components are not currently known and therefore not yet listed, or there may not be any upstream Components. This default value implies the open-world ontological assumption.[27]
2. **None**. There are no immediate upstream relationships. As defined by the Supplier, the Component has no upstream Components.
3. **Partial**. There is at least one immediate upstream relationship and may or may not be others. Known relationships are listed.
4. **Known.** The complete set of immediate upstream relationships is known and listed.

Relationship completeness assertions are intended to only assert the completeness of a Component’s immediate upstream relationships. Therefore, a Component with an assertion of “Known” completeness could have a transitive Component with an assertion of “Partial” or “Unknown” completeness. This is shown in the Section 2.6 example (see Figure 2 and Table 4) where Acme Application, the example’s Primary Component, is shown as “Known” while its

[27] Wikipedia. Closed-World Assumption.

upstream Components are given other assertions based on their immediate upstream Component completeness.

2.2.2.7 License

A Component's license identifies the legal terms for supplied software Components. Identifying the Component license enables transparency of the terms and conditions under which the software can be used, modified, and distributed.

License transparency is important to software security because unauthorized/unlicensed use of code may be subject to the inability to use or upgrade software Components (e.g., new features or patches; see also, "Delayed Open-Source Publication").[28]

Additionally, from a workflow perspective, there is significant value in a single artifact that satisfies both licensing reviews as well as vulnerability/trust reviews. Software vendors will have to satisfy two workflows, possibly with fractured assessment mechanisms; this will increase friction and potentially disparate information within an organization regarding software composition.

In most cases, license information provided for a Component would include a license identifier in a standard form, i.e., SPDX license identifier.[29] If the Component's license is not available in a standard form, refer to the specification for the SBOM format being utilized for alternate methods for declaring the license information.

The Component License data maturity levels are:

> **Minimum Expected** - Provide license information for the Primary Component.
>
> **Recommended Practice** - Provide license information for as many Components as possible.
>
> **Aspirational Goal** - Provide license information for all listed SBOM Components. Attestation of Concluded License information, i.e., license text and concluded terms and conditions, is included in the SBOM.

2.2.2.8 Copyright Notice

The Component Copyright Holder identifies the entity that holds exclusive and legal rights to the listed Component in the SBOM. Copyright information is very helpful for identifying the legal owner of the Component when triaging a security vulnerability.

Conveying the copyright notices for open-source Components is also a standard condition of many open source licenses. By including the copyright notices, the SBOM can more fully satisfy

[28] Seth Schoen, James Vasile, and Karl Fogel**.** Delayed Open Source Publication: A Survey of Historical and Current Practices. January 2, 2024.
[29] SPDX. SPDX License List (Version 3.25.0). August 19, 2024.

both security and legal workflows simultaneously and not require a second workflow for this specific license condition.

Additionally, not complying with a fundamental license obligation of many, if not most, of the open-source licensed software, introduces the security risk of being able to distribute that Component downstream.

Copyright information may be found in software file headers, LICENSE.txt, NOTICES.txt, THIRD PARTY.txt or other documents in the Component repository.

The Component Copyright Holder data maturity levels are:

> **Minimum Expected** – Provide copyright notice for the Primary Component.
>
> **Recommended Practice** – Provide copyright notice for as many Components as possible.
>
> **Aspirational Goal** – Provide copyright notice for all listed SBOM Components.

2.3 Undeclared SBOM Data

There are cases where certain Components or Component Attributes may not be available, may not make sense, may not be shareable due to contractual obligations, or may not materially contribute to Component identification at the time of an SBOM's creation. The following are recommendations for handling undeclared data in known use cases.

Be cautious if you proceed with these alternatives to providing Component Attributes, as the goal of SBOMs is to provide software Supply Chain Transparency. Providing options for SBOM data to be undeclared is not intended to promote the obfuscation of the Component data. These options are meant to enable SBOMs to be created in good faith even when missing data, conflicting data, or contractual obligations become a barrier. It is expected that, if an SBOM has undeclared data, the Author continues to work to remove the barriers and reissue the SBOM with the appropriate data.

2.3.1 Unknown Component Attributes

The lack of first-hand knowledge of the composition of Components significantly contributes to the need for a method to manage missing or inadequate data for Component identification. If the Author of the SBOM is not the Supplier of the software Component, the Author may lack the information or visibility necessary to generate some Attributes. Another factor is the point in time at which the SBOM (and the Component) is created, roughly: pre-build, at build or packaging time, and post-build. For example, binary software composition analysis performed post-build by a non-supplier. Author may detect a Component but not extract the binary Component to generate a hash.

SBOMs must handle cases of missing Attributes gracefully. A basic recommendation is to always provide all of the Baseline Attributes but explicitly define values that differentiate between “no assertion” (i.e., data is missing) and “no value” (i.e., the Attribute is not applicable for this specific SBOM). Alternatively, an SBOM format can permit missing Baseline Attributes and treat them as default values (i.e., “no assertion” or “no value”). Refer to the specific SBOM format specifications for implementation.

The Unknown Component Attributes data maturity levels are:

> **Minimum Expected** - Declaring a Baseline Attribute as “no assertion” or “no value” is only used when necessary to provide an SBOM in a reasonably timely manner.
>
> **Recommended Practice** - Declaring a Baseline Attribute as “no assertion” or “no value” is used rarely when sufficient effort matching the potential security risk of that Component is spent.
>
> **Aspirational Goal** - When a Baseline Attribute for a Component has been declared as “no assertion” or “no value”, it is proactively tracked as a compliance gap.

2.3.2 Redacted Components

At times, downstream Distributors of software will have contractual agreements with upstream Suppliers that require that the inclusion of software not be divulged publicly. Therefore, a redaction of that Component from the publicly distributed SBOM is needed. In this case, it is recommended to:

- Indicate the Component as redacted and provide a redaction rationale.
- Remove any Component identifying information but preserve the Component’s version or cryptographic hash.
- Maintain any of the dependency relationships to or from the redacted Component in a manner that honors the redaction.

The Redacted Components maturity levels are:

> **Minimum Expected** - A Component’s Attributes that could identify it are redacted only when a Supplier contract requires it.
>
> **Recommended Practice** - The contract with a Supplier of a Component that requires redaction is approached for re-negotiation of a contract that allows Supply Chain Transparency. Alternatively, a more transparent upstream Supplier is selected for future software development.

Disclaimer**:** Your regulatory authorities may require a complete and accurate disclosure of all manufacturer-developed and third-party software Components as part of an SBOM. Similarly, customers may also necessitate such disclosure.

2.3.3 Unknown Dependencies

When it is common knowledge that supplied software has Dependencies, but the included Dependencies are not known or only partially known, an SBOM system needs to be able to indicate that the list of Component Dependencies is incomplete. For example, if a proprietary operating system is included in your distributed software but many of the included Dependencies of the operating system are unknown, the SBOM lists the known Dependencies and an indication that there are additional unknown Dependencies not listed.

The Unknown Dependencies data maturity levels are:

> **Minimum Expected** - Every Direct Dependency to the Primary Component is identified in the SBOM with all Baseline Attributes identified (or indicated as shown in Section 2.3.1). Deeper Dependencies may be declared as unknown, when necessary. It is recommended to provide a rationale if Dependencies are declared as unknown.
>
> **Recommended Practice** - Contact the upstream Supplier for their SBOM to provide the Component data needed. Provide this information either nested within your Primary Component's SBOM or separately.
>
> **Aspirational Goal** - Use tooling to produce SBOMs for upstream supplied software where an SBOM cannot be procured. Also, use tooling to gather data from upstream Supplier SBOMs and your organizational development to assist in producing robust SBOMs.

More details about how to indicate unknown Dependencies in the assertions for relationship completeness can be found in Section 2.2.2.6.4.

2.4 Supplemental Information to Support Use Cases

In addition to Baseline Attributes, an SBOM can be supplemented to contain elements and Component Attributes to support different use cases. The specific information supplemented depends on the use case and not all supplemental elements or Attributes will support each use case. Potential SBOM use cases are described in Section 3.6. The following are just a few examples of supplemental elements and Attributes that could enhance the SBOM data.

Examples of supplemental Attributes:

- End-of-life date or level-of-support for Components.
- Indication of what technologies a Component implements or supports.

Examples of supplemental elements:

- Grouping of Components[30] (i.e., group by product lines or implemented technologies to be treated as a special type of upstream Component). For example, knowing that "component X and Component Y implement DNS" allows a user to identify all DNS-related Components and treat them as a collection.
- Authenticity and integrity capability (i.e., cryptographic authentication and verification of SBOM information). An SBOM ecosystem must support the ability to cryptographically authenticate and verify SBOM information. In general, this means that authors must be able to digitally sign SBOMs and Consumers must be able to verify signatures. Authentication and integrity protection requires appropriate digital signature and public key infrastructure.

[30] CISA Open Working Group on SBOM Tooling and Implementation. Guidance on Assembling a Group of Products. January 26, 2024.

2.5 Mapping to Existing Formats

Table 1 maps Baseline Attributes for both the SPDX and CycloneDX SBOM formats. In addition to the Baseline Attributes, authors should conform to the specifications of their chosen SBOM formats.

Table 1: Mapping baseline Component information to existing formats

Attribute	ISO/IEC 5962:2021	SPDX 3.0	CycloneDX v1.6 (ECMA-424)
SBOM Author Name	(6.8) Creator:	Core.CreationInfo.createdBy	metadata.authors
SBOM Timestamp	(6.9) Created:	Core.CreationInfo.created	metadata.timestamp
SBOM Type	(6.10) CreatorComment:	Software.Sbom.sbomType	metadata.lifecycles
SBOM Primary Component	(11.1) Relationship: DESCRIBES	Software.Sbom.rootElement	metadata.component
Component Name	(7.1) PackageName:	Software.Package.name	components[].name
Component Version String	(7.3) PackageVersion:	Software.Package.packageVersion	components[].version
Component Supplier Name	(7.5) PackageSupplier:	Software.Package.suppliedBy	metadata.supplier components[].supplier
Component Cryptographic Hash	(7.10) PackageChecksum: (7.9) PackageVerificationCode:	Software.Package.verifiedUsing	components[].hashes[]
Component Unique Identifier	(6.5) SPDX Document Namespace (7.2) SPDXID:	Core.Artifact.spdxId Software.SoftwareArtifact.contentIdentifier Software.SoftwareArtifact.externalIdentifier (cpe22, cpe23, cve, gitoid, packageUrl, swhid, swid, securityOther, other)	serialNumber + version components[].cpe components[].purl components[].swid components[].omniborId components[].swhid components[].evidence.identity
Component Relationships	(11.1) Relationship: CONTAINS	Core.Relationship contains dependsOn hasStaticLink hasDynamicLink hasProvidedDependency hasOptionalDependency	dependencies[] components[].components
Component License	(7.15) PackageLicenseDeclared: (7.13) PackageLicenseConcluded:	Core.Relationship hasConcludedLicense hasDeclaredLicense	components[].licenses[] components[].licenses[].acknowledgement[declared, concluded]

	(7.14) LicenseInfoFromFiles:		components[].licenses[].licensing (proprietary) components[].evidence.licenses[]
Component Copyright Holder	(7.17) PackageCopyrightText:	Software.SoftwareArtifact.copyrightText	components[].copyright components[].evidence.copyright

2.6 SBOM Examples

To further illustrate the relationships described in the previous sections, consider these SBOM examples. Figure 1 and Table 2 show two different approaches to viewing SBOM information and relationships. These are conceptual representations and not specific formats like SPDX and CycloneDX. SBOM examples supported by the SPDX[31] and CycloneDX[32] formats can be leveraged for additional insight.

In Figure 1 and Table 2, the SBOM authored by Acme includes four Components. One of these, the Primary Component, is the Acme Application, which defines the subject of the SBOM. Acme makes a Component named “Application” that uses two upstream Components, Bob’s Browser and Bingo Buffer. In this example, Acme was able to obtain SBOM information from Bob about Bob’s Browser, which, in turn, uses Carol’s Compression Engine and possibly other upstream Components. Acme was not able to obtain SBOMs from Carol or Bingo, so Acme authored SBOMs for those Components. Carol’s Compression Engine does not include upstream Components, while Bingo Buffer may or may not have any upstream Components.

[31] SPDX GitHub Repository. SPDX Examples.
[32] Cyclone DX GitHub Repository. BOM Examples.

Figure 1: Conceptual SBOM graph

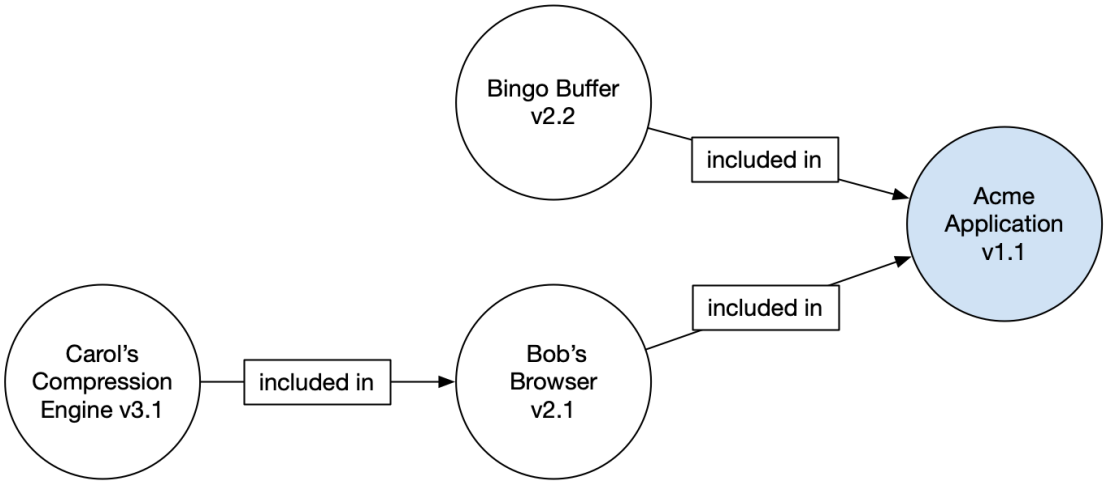

Table 2: Conceptual SBOM table[33]

Component Name	Supplier	Version	Author	Hash	UID	Relationship
Application	Acme	1.1	Acme	0x123	234	Primary
|--- Browser	Bob	2.1	Bob	0x223	334	Included in
|--- Compression Engine	Carol	3.1	Acme	0x323	434	Included in
|--- Buffer	Bingo	2.2	Acme	0x423	534	Included in

In the simplest case, a single Component is created entirely from scratch with no Dependencies: Carol's Compression Engine v3.1. The SBOM for this Component consists of only one entry that defines both the Component and the SBOM using the relationship type of "Primary." This example is shown in Table 3.

Table 3: Conceptual SBOM table for a single (and Primary) Component

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Completeness
Compression Engine	Carol	3.1	Carol	0x323	434	Primary	None

In Figure 2 and Table 4, the SBOM authored by Acme for the Component "Acme Application" also has four Components, and the relationship information is now enhanced with assertions regarding completeness.

[33] In all similar tables, the Timestamp Attribute is omitted and other Attribute names shortened for presentation purposes.

Figure 2: Conceptual SBOM graph with upstream relationship completeness

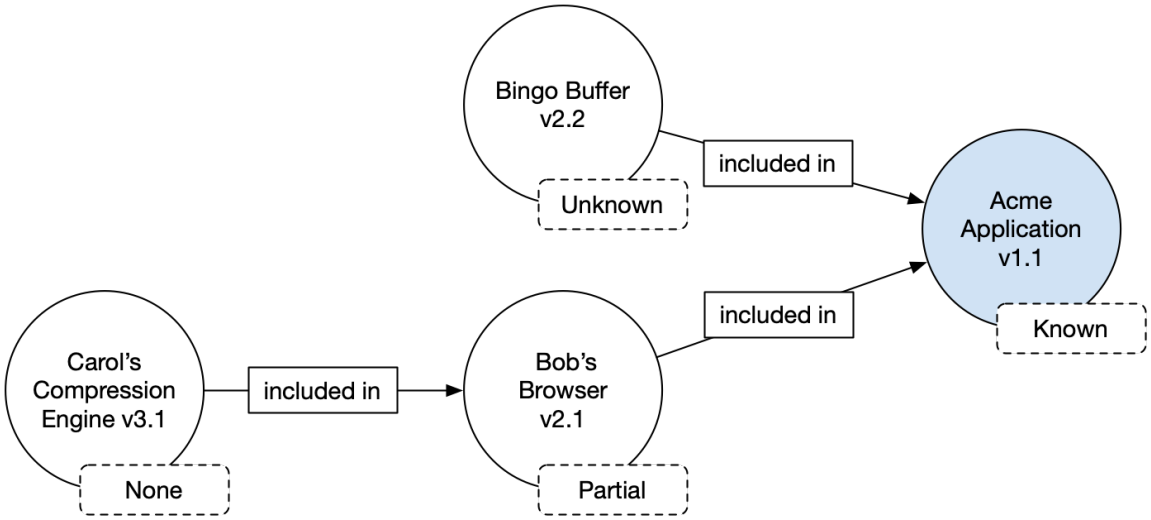

Table 4: Conceptual SBOM table with upstream relationship completeness

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Completeness
Application	Acme	1.1	Acme	0x123	234	Primary	Known
|--- Browser	Bob	2.1	Bob	0x223	334	Included in	Partial
|--- Compression Engine	Carol	3.1	Acme	0x323	434	Included in	None
|--- Buffer	Bingo	2.2	Acme	0x423	534	Included in	Unknown

Acme Application (the subject and Primary Component of this SBOM) asserts "Known" since all immediate upstream Dependencies are covered. Bob's Browser asserts "Partial" since at least Carol's Compression Engine is upstream of it. Carol's Compression Engine has no upstream Components and the assertion for relationship completeness is "None." Bingo Buffer is known to be an immediate upstream Dependency of Acme Application, but since nothing is known upstream of Bingo Buffer, the assertion for relationship completeness is "Unknown."

3 SBOM Processes

This section describes how to create and exchange SBOM information from three stakeholder perspectives: those who produce, choose, and operate software. These perspectives are described in detail in "Use Cases: Roles and Benefits for SBOM Across the Supply Chain."[34]

Three particular SBOM use cases — vulnerability management, intellectual property, and secure Supply Chain software assurance — illustrate an SBOM as an independent data source

[34] NTIA. Software Bill of Materials.

as well as how an SBOM can be integrated into typical business processes. These are discussed in Section 3.6.

3.1 SBOM Creation: How

To create an SBOM, the Supplier defines Components that the Supplier creates themselves, produces baseline and any supplemental Attributes for those Components, and enumerates all directly included Components. SBOM information will ideally be generated as an integral part of the Supplier's software build and packaging processes, which can be accomplished with modifications to existing development tools.

Any entity creating, modifying, packaging, and delivering software or software systems is considered a Supplier and is therefore responsible for defining Components and creating SBOMs. This includes system integrators, who are essentially considered Suppliers for SBOM purposes. An organization can also act as a Supplier for internally developed Components.

When SBOMs for Included Components are available from upstream Suppliers, those SBOMs are provided with or incorporated into the primary SBOM. Where such information is not available, a Supplier can provide "best effort" SBOMs, which will be indicated by the fact that the Author for an Included Component SBOM will not be the same as the Supplier of the Component. Section 2.2.2.6 describes a way for SBOM authors to make assertions about indirectly included upstream Components for which the Supplier has not provided an SBOM.

An SBOM includes Attributes used to identify Components and supplemental Attributes to capture characteristics of, or information about, Components. Identity Attributes are essential, and supplemental Attributes may or may not be required depending on the use case or application.

An SBOM from the Component's Supplier serves as a system of record or authoritative source of information about the Component. As noted elsewhere, some information may need to be validated with other external sources. For example, vulnerability information about a Component can sometimes be derived from the NVD using CPE.

3.2 SBOM Creation: When

An SBOM needs to be created when a Component is released. This loosely corresponds to build, packaging, or deployment activities. As discussed in Section 2 and Section 2.2.1.3, consider using the Type Attribute of SBOM to indicate the context from which the SBOM was created or assembled.

A new SBOM should be created when the Component is updated or versioned, including when new upstream Components are added. Changes to Components are often noted as updates, upgrades, releases, and patches. Ideally, changes to Components are indicated by a change in the Version String Attribute. The original SBOM may require updates when new SBOM information becomes available even if the Components themselves have not changed.

Maintaining a current SBOM with information and Baseline Attributes that are as complete as possible is essential.

When changing an existing Component (including patching or updating), there are two options for documenting the change: A) as a separate, new Component added to the existing SBOM or B) as the same Component with a new version string. These two options are illustrated in the example below.

Table 5: Options for documenting patches or updates in SBOMs

Before update	**SBOM Update Options**	
	Separate, new Component (A)	**Same Component with new version string (B)**
Bob's Browser v1.1	Bob's Browser v1.1 Bob's Browser update 37	Bob's Browser v1.1.1

3.3 SBOM Exchange

It is necessary to exchange SBOM information.[35] The primary exchange is directly from a Supplier to a Consumer through a single downstream Supply Chain link. As part of delivering the Component, the Supplier also delivers the SBOM, or a means by which the Consumer can easily obtain the SBOM, such as a URL or other reference. This direct delivery does not preclude aggregation or cataloging of SBOM information by Suppliers, Consumers, or others.

Due to the variety of different software and device ecosystems, it is unlikely that a single SBOM exchange mechanism will suffice. They can be provided as additional files as part of a Component's distribution or delivery. For devices with storage and power constraints, one option is to provide a URL to look up SBOM information on a Supplier's website. Dynamic access to an SBOM may be a good option for such devices as well. The Internet Engineering Task Force (IETF) has developed a protocol and format for end user discovery of SBOMs, whether they are shared on a local device or on a website. The specification is "format neutral," meaning it can support SPDX, CycloneDX, and future formats too.

3.4 Software Supply Chain Rules

Participants in a secure software Supply Chain include Suppliers and authors who create SBOM information, Consumers who receive SBOM information, and as providers of optional intermediary services such as composition analysis and Dependency analysis. In many cases, a participant acts as both a Supplier and Consumer, operating somewhere in the middle of a Supply Chain.

[35] CISA Open Working Group on SBOM Sharing and Exchanging. SBOM Sharing Primer. 2024.

Participants follow a set of Supply Chain rules so that SBOM systems function at scale. Suppliers create SBOMs for Components the Suppliers develop themselves, and Suppliers define these Components. For upstream Components, Suppliers obtain SBOMs from the appropriate upstream Suppliers. If upstream SBOMs are not available, the Supplier or other authors can create SBOMs, even when this involves researching the appropriate entry or omitting Baseline Attributes.

An SBOM must list a Primary Component, which defines the subject of the SBOM. An SBOM lists Components that are:

1. Originally created by a Supplier who is the authoritative source of the software
2. Integrated as a Component from an upstream Supplier who also provides an SBOM
3. Integrated as a Component from an upstream Supplier who does not provide an SBOM.

As part of delivering Components to users, Suppliers also deliver the associated SBOM(s) or provide a means for the Consumer to easily obtain SBOMs.

A set of many interconnected Supply Chains is likely a directed acyclic graph, as shown in Figures 1, 2, and 3. Ultimate upstream Suppliers only create original Components and do not include Components (i.e., do not have dependencies) from any other Supplier. In Section 2.6, Carol is an example of such a Supplier. Components flow downstream along Supply Chains throughout the graph. At the far ends of the graph, ultimate Consumers only obtain Components and SBOMs and do not produce Components or SBOMs. Throughout the middle of the graph, most participants act as both Suppliers and Consumers. Even end-user organizations may act as Suppliers, producing SBOMs for in-house Components or external Components such as websites, mobile applications, or Internet of Things (IoT) devices.

Suppliers are responsible for the Components they create and include in an SBOM. Suppliers are also responsible for providing the collected set of Components to their downstream Consumers. In a macroeconomic sense, Suppliers are the least cost avoiders, since they have high-quality authoritative information about their Components and comparatively low costs to generate and share that information.[36] This model distributes the cost to produce SBOM information to Suppliers.

In this secure Supply Chain network, there are some scenarios where a Supplier may create SBOMs for upstream Components, where the Supplier is acting as the Author of the SBOM and not the Supplier of the upstream Component. When a Supplier creates such SBOMs, the Supplier is expected to clearly convey that they are only the Author of the SBOM and are not the Supplier of the Component. This informs Consumers of the lack of first-hand, authoritative SBOM information for the Component. In such a case, the Author Name and Supplier Name would be different.

The concepts around SBOM exchange and network rules are designed so that those who choose and operate software can obtain comprehensive lists of Components they use across

[36] Paul Rosenzweig. Cybersecurity and the Least Cost Avoider. November 5, 2013.

different Suppliers and Supply Chains. Figure 3 expands the example in Figure 2 to show a user of two software products (Primary Components) from two different Supply Chains. The user has two SBOMs: one shown in Table 4 and one in Table 6.

Figure 3: User graph with two Supply Chains

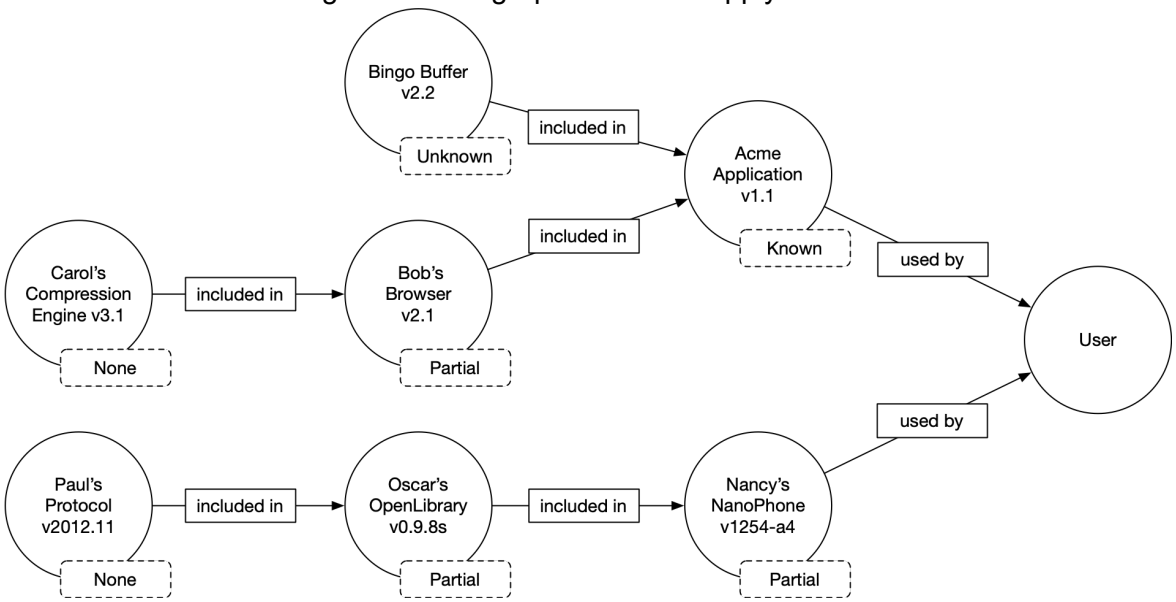

Table 6: Conceptual SBOM table representation for Nancy's NanoPhone

Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship	Relationship Completeness
NanoPhone	Nancy	v1254-a4	Nancy	0x523	237	Primary	Partial
|--- OpenLibrary	Oscar	0.9.8s	Nancy	0xA23	394	Included in	Partial
|--- Protocol	Paul	2012.11	Nancy	0xB53	934	Included in	None

3.5 Roles and Perspectives

3.5.1 Perspectives

Different stakeholders will use SBOMs in complementary, yet distinct, ways. "Roles and Benefits for SBOM Across the Supply Chain" presents three stakeholder perspectives: those who produce, choose, and operate software.[37] This document borrows the definitions introduced in "Roles and Benefits for SBOM Across the Supply Chain," with changes to update the terms to present use cases.

[37] NTIA Open Working Group on SBOM Use Cases and State of Practice. Roles and Benefits for SBOM Across the Supply Chain. November 2019.

3.5.1.1 Produce

This SBOM model is designed with the idea that all Suppliers create SBOMs for their own Components. When SBOMs for upstream Components are not available, a Supplier may need to Author an SBOM for a different Supplier's Component. In the case where a Supplier does not provide SBOM information, there is a higher likelihood that this lack of clarity will cause downstream users to assume the worst about the unknown parts of the product. An additional benefit to Suppliers is the ability to determine which organization to contact to get fixes for vulnerabilities in upstream Components.

3.5.1.2 Choose

SBOMs can be used by prospective Choosers (e.g., development, acquisition, or procurement) considering the use of a Component or product that has an associated SBOM. Choosers are likely to be interested in information directly attributable to the product, such as its baseline Component or license information. Supplemental SBOM information about vulnerabilities, current certification status, and level of support can factor into the selection process.

3.5.1.3 Operate

In any industry, Operators struggle with the lack of complete information on Components or products they are expected to support. An SBOM becomes a very relevant source of this information to provide visibility into the software and its Components. Some of this information may be static, such as licensing information. However, due to the dynamic nature of software, some of this information may change or be updated after a Component's initial distribution.

Most of the information of ongoing interest for Operators is expected to be found in SBOM updates. Operators can also use the current information to verify the state of the software before it is to be in production at their site or business.

3.6 SBOM Use Cases

The core focus of this SBOM model and the Baseline Attributes is to identify Components and their relationships. Supplemental elements and Attributes such as new Attributes, relationship types, or external data connections, can be added to SBOMs to enable different use cases. This section highlights several notable SBOM use cases.

3.6.1 Vulnerability Management and VEX

Vulnerability management is one of the more prominent SBOM use cases. Today, it is often an expensive and time-consuming effort to determine whether a vulnerable upstream Component is used, and if the vulnerability is present or exploitable in downstream Components. SBOM and Vulnerability Exploitability eXchange (VEX) data helps Suppliers, users, and other defenders more quickly and accurately assess the risk posed by vulnerable Components, which are often hidden behind opaque Supply Chain relationships.

While upstream Components are typically included to provide functionality, it is common for parts of the Component to be unused. A software program (component) might include a library (component) but only call some of the functions provided by the library. Additionally, certain features of a Component may be disabled during build or packaging. This becomes important in some SBOM use cases, particularly vulnerability management. For example, if a vulnerability affects an upstream Component, the vulnerability may or may not affect downstream Components.[38] VEX is designed to convey the status (not affected, affected, fixed, under investigation) of vulnerabilities in Components.[39]

Vulnerability management requires sources of vulnerability information (such as CVE, security advisories from Suppliers, [e.g. in CSAF, and the NVD]), mapping of vulnerabilities to Components (such as CPE as used in the NVD), and a way to convey vulnerability or exploitability status (such as VEX). While VEX was developed to address the vulnerability management use case, VEX is not limited to use with SBOMs nor expected to be included in the SBOM itself. One concern is the incorrect detection of vulnerabilities based on limited information such as version strings, protocol banners, or other heuristics. VEX can be used to indicate that software is not vulnerable or exploitable, even when the SBOM indicates the presence of vulnerabilities in upstream Components. This can save Suppliers and users the costs of managing, producing, and applying security updates for Components that are not affected.

Additionally, including the end-of-life date and level-of-support for the Components as supplemental to the SBOM provides the entity performing an impact assessment of a vulnerability with crucial information for mitigation options.

3.6.2 Intellectual Property

A number of Intellectual Property (IP) use cases can be improved with better inventory data. Managing software license and copyright notice information (including constraints on use or redistribution) for Included Components and tracking entitlement (permission to use copies or features of Components) are two common use cases. A notable market exists for software composition analysis tools to help determine the contents of Components, in part to identify license requirements. SBOM data would improve knowledge about composition without depending on binary analysis tools. Both SPDX and CycloneDX were initially designed to convey license and entitlement information.

This use case requires associations of different licenses and types of licenses to Components and a way to evaluate the net effect of different Components with different licenses combined into an assembled product.

---

[38] Veracode. Open Source Components: Vulnerability Information Sources and Vulnerability Likelihood. July 19, 2018.
[39]NTIA Multistakeholder Process. Vulnerability-Exploitability eXchange (VEX)—An Overview. September 27, 2021.

3.6.3 Secure Supply Chain Software Assurance

Secure Supply Chain of the source and integrity of Components requires information about the pedigree and provenance of Components, such as how they were built and packaged, who created and modified them, and their chain of custody through the Supply Chain. As with the other use cases, the secure Supply Chain documentation will require additional Attributes about Components, different relationship types, and likely different Supplier information.

3.7 Tool Support

The availability of SBOM generation and management tools will be critical for widespread adoption. Available tools and lists of tools include the CycloneDX Tool Center[40] and SPDX Tools[41]. SBOM functionality will need to be integrated into software development, packaging, and asset management systems.

[40] Cyclone DX. Tool Center.
[41] SPDX. Tools.

4 Conclusion

Organizations around the world are facing operational and secure Supply Chain software assurance questions about the software actively deployed in their environments. Much of this software handles critical parts of their business activities while providing little or no visibility into the software's Components. Questions about known vulnerabilities continue to go unanswered because of this lack of visibility. One way to increase cybersecurity automation and software transparency, enable enterprises to better manage the security of their networks, and enable vendors to monitor their Components is to establish a harmonized model for creating and sharing SBOMs.

To be useful to end-user organizations, an SBOM needs to include baseline identity and relationship information that allows software Components to be correlated and linked as they move through the secure software Supply Chain. In the interest of rapid adoption, a set of minimum Baseline Attributes has been defined. These Attributes generally align with existing formats such as SPDX and CycloneDX. As noted in this document, however, limiting an SBOM to only this baseline information is not sufficient to enable a number of identified use cases and applications.

As the use of SBOMs matures and becomes more common, the ready availability of baseline SBOM information will lead to further work to establish more coordinated and standardized methods for sharing and managing SBOMs. One of the reasons to standardize the structure and content of the SBOM is to enable these next steps. Tooling will also be an important factor in the adoption and further evolution of SBOMs.

Overall, the goal is to ensure that the necessary information, captured and exchanged through SBOMs, is available to those who need it, thereby leading to better asset management, IP management, vulnerability management, implementation of mitigations, and risk management.

Appendix A Edition Changes

Significant changes between the Second Edition (2021) and Third Edition (2024) include:

- Updated language throughout Section 2 to
    - Clarify SBOM expectations for each Baseline Attribute
    - Add two Baseline Attributes - license and copyright holder
    - Introduce maturity levels for multiple Attributes
    - Update mapping of Attributes to SPDX and CycloneDX formats
    - Remove SWID as an existing format
    - Add options for undeclared SBOM information in Section 2.3
    - Add the concept of risk management as part of the SBOM consumption process
- Terminology section:
    - Moved content to Appendix B
    - Added necessary terms due to updates
- Made various editorial improvements and clarifications

Significant changes between the First Edition (2019) and Second Edition (2021) include:

- Added Timestamp to Baseline Attributes
- Clarified requirements aspects of Baseline Attributes
- Added CycloneDX as an additional format
- Removed Existing Formats (previously Section 3), renumbered accordingly
- Updated language in Baseline Attributes and Terminology
- Updated and harmonized language across working groups
- Updated figures and tables
- Made various editorial improvements and clarifications

Appendix B Terminology

The following terms have specific meaning within the scope of this document and within the overall multistakeholder process. Each definition is written to be a direct grammatical replacement for the term.

Term	Definition	External Source
Attribute	An Attribute is a characteristic of, or information about, a Component. Baseline Attributes are defined in Section 2.2. Other Attributes can be defined as needed to meet specific use cases and applications.	This document
Author	The Author reflects the source of the metadata, which could come from the creator of the software being described in the SBOM, the upstream Component Supplier, or some third-party analysis tool. Note that this is not the Author of the software itself, just the source of the descriptive data.	SBOM Minimum Elements Document
Chooser	The Chooser is the person/organization that decides the software/products/Suppliers for use	Roles and Benefits for SBOM Across the Supply Chain
Component	A Component is a unit of software defined by a Supplier at the time it is built, packaged, or delivered. Many Components contain subcomponents, or upstream Components. Examples of Components include a software product, a library, or a single file. Depending on the perspective in the Supply Chain, a Component (often the Primary Component) can be considered to be a product, intermediate good, final good, or final assembled good.	This document
Concluded License	Frequently multiple licenses may be found in a Component that have different constraints. After resolving the conditions an overall license for the Component can be declared by the SBOM Supplier.	This document

Term	Definition	External Source
Consumer	The Consumer receives the transferred SBOM. This could include roles such as third parties, authors, integrators, and end users.	SBOM Sharing Lifecycle Report/SBOM Sharing Roles and Considerations
Dependency	A Dependency is the relationship between two Components. Many Components are dependent on other Components to function well. A Dependency could be described in the following ways. Multiple descriptors could be used for a single Component. ○ Static: both Components are included in the software. ○ Dynamic: at least one of the Components in the relationship is loaded upon request. ○ Remote: at least one of the Components in the relationship is called and runs outside of the Primary Component software being described in the SBOM. ○ Provided: at least one of the Components in the relationship is expected to be provided by the software environment on which the Component is run. ○ Direct: at least one of the Components in the relationship is the Primary Component. ○ Transitive: neither Component in the relationship is the Primary Component, meaning that the Component is nested at least two levels.	This document
Distributor	A Distributor receives SBOMs for the purpose of sharing them with SBOM Consumers or other Distributors.	SBOM Sharing Lifecycle Report/SBOM Sharing Roles and Considerations
Included Component	An Included Component is any Component that is in the distributed software (e.g., masked layers within a container of an image).	This document

Term	Definition	External Source
Operator	An Operator is a person/organization that operates the software Component.	Roles and Benefits for SBOM Across the Supply Chain
Originating Supplier	If the Component identified in the SBOM originated from a different person or organization than identified as Component Supplier, the Originating Supplier defines where or whom the Component originally came from. In some cases, a Component may be created and originally distributed by a different third party than the Supplier of the Component. For example, the SBOM identifies the Component as glibc and the Component Supplier as Red Hat, but the Free Software Foundation is the Originating Supplier.	This document
Software Bill of Materials (SBOM)	An SBOM is a formal, machine-readable inventory of software Components and Dependencies, information about those Components, and their relationships.	This document
Supplier	The Supplier refers to the originator or manufacturer of the software Component.	SBOM Minimum Elements Document
Supply Chain	A Supply Chain is a linked set of resources and processes between multiple tiers of developers that begins with the sourcing of products and services and extends through the design, development, manufacturing, processing, handling, and delivery of products and services to the acquirer.	Glossary | CSRC (nist.gov)
Supply Chain Transparency	Supply Chain Transparency is the amount of information that can be gathered about a Supplier, product, or service and how far through the Supply Chain this information can be obtained.	Glossary | CSRC (nist.gov)

Appendix C Third Edition Acknowledgements

This Framing document was drafted and revised by a diverse set of experts from across the software ecosystem and is intended to capture a threshold of the current state of practice and expectations for software transparency in 2024.

The leaders of the Tooling and Implementation Working Group, Melissa Rhodes (Medtronic) and Kate Stewart (Linux Foundation), thank the members of the working group for their time, expertise, and dedication to the content presented in this document's third edition. We also thank our hosts from CISA, namely Allan Friedman, Victoria Ontiveros, and Jeremiah Stoddard, for their assistance in bringing this working group together and hosting us as we worked through and finalized the content.

Acknowledgements do not imply endorsement of this document and its content.

Alicia Bond, Vigilant Ops
Andrea Grover, University of Nebraska at Omaha
Bill Pelletier, ZOLL Medical
Bob Haack, Johnson & Johnson
Bob Martin, The MITRE Corporation
Brian Benavidez, BIOTRONIK
Brian Smithson, TrustCB
Cassie Crossley, Schneider Electric
Charles Long, Arthrex
Charlie Hart, Hitachi America Ltd.
Chris Gregoire, Boston Scientific
Daniel John Audette, Hewlett Packard Enterprise (HPE)
David Dillard, Veritas Technologies LLC
Deanna Medina, United Airlines
Derek Garcia, University of Hawaii at Manoa
Dick Wilkins, Phoenix Technologies
Dmitry Raidman, Cybeats
Duncan Sparrell, sFractal Consulting
Evgeny Martinov, Cybellum Technologies
Eyal Traitel, Cybellum Technologies
Gary O'Neall, Source Auditor and SPDX
Girish Jorapurkar, Splunk
Ian Dunbar-Hall, Lockheed Martin and OpenSSF
Isaac Hepworth, Google
Isabella Donders, Security Pattern
Ixchel Ruiz, Karakun
James Tramel, Tesco
Jesse Martinez, State of Iowa
John Cavanaugh, Internet Infrastructure Services Corp

John Nuckles, Office of Director of National Intelligence
Joyabrata Ghosh, CARIAD SE
Dr. LaTrea Shine, Lenovo
Lynn Westfall, The Modem Lisa
Manish Jadhav, Vigilant Ops
Melissa Chase, The MITRE Corporation
Michael Bandor, Carnegie Mellon University/Software Engineering Institute (CMU/SEI)
Penny Rose Boulet, Hewlett Packard Enterprise (HPE)
Pete Allor, Red Hat
Prosunjit Biswas, Adobe
Ria Schalnat, Hewlett Packard Enterprise (HPE)
Ricardo Reyes, Tidelift
Ryan Stewart, SAP
Saquib Saifee, IBM
Sangeetha Shankar, The MathWorks Inc.
Scott Van Eps, Danaher
Shweta Singh, MathWorks
Sridhar Balasubramanian, NetApp, Inc.
Steve Springett, ServiceNow and the OWASP Foundation
Sunny Ahn, SettleTop
Syed Zaeem (‘Z’) Hosain, Aeris Communications Inc.
Timothy Walsh, Mayo Clinic
Vijaya Ramamurthi, Accenture Federal Services
Viktor Petersson, sbomify