

June 17, 2021

Ms. Evelyn L. Remaley  
Acting Administrator  
National Telecommunications and Information Administration (NTIA)  
U.S. Department of Commerce  
1401 Constitution Avenue, N.W., Room 4725  
Washington, District of Columbia 20230

Submitted via electronic mail to SBOM\_RFC@ntia.gov

**Re: Software Bill of Materials Elements and Considerations, Docket No. 210527-0117**

Dear Acting Administrator Remaley:

Microsoft appreciates the opportunity to provide comments on NTIA's proposed minimum elements of a Software Bill of Materials (SBOM) and related considerations. Along with other efforts prompted by the Executive Order on Improving the Nation's Cybersecurity (EO), NTIA's coordination with the Secretary of Commerce to publish minimum elements of an SBOM has the potential to significantly impact both near- and longer-term efforts to enhance software supply chain security and integrity. NTIA's approach to inviting participation from and consulting with the multistakeholder community on its cybersecurity initiatives on SBOM and other topics has fostered a unique dialogue, helping to lay the groundwork for a more broadly shared understanding of challenges and opportunities intrinsic to industry efforts.

Across our engineering groups and industry engagement functions, Microsoft invests substantial resources in software security and supply chain integrity efforts, including those related to SBOM. We've worked on internal formats to convey SBOM information and contributed to industry and multistakeholder groups that are developing and discussing SBOM and other software security and supply chain integrity standards, specifications, and tools.

Our internal efforts and ecosystem partnerships have helped demonstrate to us the immediate and potential value as well as the complexity of providing SBOM information and implementing other practices to enhance software supply chain integrity. We're encouraged by NTIA's approach to focusing on minimum SBOM data fields and operational factors while also inviting input on how various issues might impact future expectations related to minimum SBOM elements. Consistent with NTIA's approach on SBOM and other multistakeholder cybersecurity initiatives, we recognize the importance of and share an interest in driving near-term progress, including by orienting around what both provides the greatest security value and is feasible to deploy widely – without losing sight of our longer-term vision and practical next steps.

Below, in responses to questions posed by NTIA as well as supporting context, we share our perspective on considerations for defining SBOM elements today and in the future. In developing these responses, we brought together experts from our Cloud + AI and Experiences and Devices engineering groups, which include Azure, Office, and Windows among other products and services, and from GitHub, where over 65 million users develop software, to represent our diverse vantage points.

**1. Are the elements described above, including data fields, operational considerations, and support for automation, sufficient? What other elements should be considered and why?**

As further described below in sections I and II on [Elements for an SBOM](#) and [Operational Considerations](#), we recommend a flexible approach in which there are both required and optional minimum SBOM elements and an allowance for operational variability.

Required minimum data fields should include: supplier name, component name, version of the component, cryptographic hash of the component, and author of the SBOM data. Optional fields should include: any other unique identifiers for the component and dependency relationships.

For each of the above data fields, we include below considerations related to use (e.g., where data fields should primarily be for human consumption rather than for unique identification of the component) and suggestions where further clarity is needed (e.g., “supplier name”). We also provide context for distinguishing between required and optional data fields.

Among operational considerations, we also encourage NTIA to develop requirements around authentication of SBOMs through methods such as digital signatures.

**2. Are there additional use cases that can further inform the elements of SBOM?**

For the purposes of defining minimum SBOM elements for the EO, we recommend focusing on software integrity and vulnerability analysis use cases, which have a high return on investment in improving supply chain security. As further described in section III on [Microsoft’s Vision for Supply Chain Integrity](#), other use cases, which may require additional SBOM elements beyond the baseline, could be explored as optional.

**3. SBOM creation and use touches on a number of related areas in IT management, cybersecurity, and public policy. We seek comment on how these issues described below should be considered in defining SBOM elements today and in the future.**

We are encouraged by NTIA’s focus on driving implementation of what is feasible today while also engaging with stakeholders on broader issues and opportunities related to SBOMs. Our perspective on identified issues is included below in sections II and III, [Operational Considerations](#) (near term) and [Microsoft's Vision for Supply Chain Integrity](#) (longer term).

**4. Flexibility of implementation and potential requirements. If there are legitimate reasons why the above elements might be difficult to adopt or use for certain technologies, industries, or communities, how might the goals and use cases described above be fulfilled through alternate means? What accommodations and alternate approaches can deliver benefits while allowing for flexibility?**

We are supportive of outcomes-focused approaches that recognize the potential for variability across a complex and dynamic ecosystem. We underline the need for flexibility in sections II and III, [Operational Considerations](#) and [Microsoft's Vision for Supply Chain Integrity](#).

## I. Elements for an SBOM

Minimizing the number of mandatory elements, without compromising the primary use cases of software integrity and vulnerability analysis, is critical for broad and rapid adoption of SBOMs. The below-proposed required and optional elements support these primary use cases without imposing undue burden or risking significant delays in broad adoption.

We recommend the following adjustments or clarifications related to minimum required data fields as proposed by NTIA:

- Define *supplier name* as this may be ambiguous in use cases such as redistribution of a component. We recommend that this element refer to the originator or manufacturer of the software, rather than the end supplier, as the originator or manufacturer name is more useful when searching vulnerability databases. Specifications such as Decentralized Identifiers (DIDs)<sup>1</sup> provide a mechanism for describing supplier identities in a more structured and consistent way without imposing the management burden of naming authorities.
- Leverage the *component name* and *version* primarily for human consumption rather than for unique identification of the component. It is common for software to already have a definition of component name version (this is often required by package managers, installers, and other distribution mechanisms), and keeping these elements freeform will allow suppliers to use those values as-is. Whenever feasible, unique identification of the component should be accomplished using structured identifiers provided in the "any other unique identifier" element of the SBOM; in the absence of unique identifiers, the supplier name, component name, and version of the component may be used as a fallback – for example, when a software ecosystem doesn't have structured identifiers. When multiple unique identifiers are provided for a software component, they should be considered alternate ways to identify that software component with no priority inferred by their ordering. In scenarios where a component has been modified by a developer, it's important to represent this in the SBOM since the component materially changed in some way and therefore may not be affected by vulnerabilities in the original component. We recommend that this is described as a new component and that a new supplier is identified but that the new component is linked back to the original component from which it was derived.
- Ensure that the required *hash of the component* is flexible enough to (a) support multiple, concurrent hashes; (b) be resilient to hash algorithms being deprecated over time (agile); and (c) selected from the set of hash algorithms as specified by NIST.<sup>2</sup> Initial requirements should include a strong hash such as SHA-256 immediately; later updates can codify rules and guidance on how to handle deprecation and new algorithms over time. Allowing any of the recommended algorithms without requiring a specific algorithm will make it more likely that suppliers can use hashes that are already being generated in their processes; for example, many installer and packaging technologies already generate hashes that could be used for SBOM generation. If a component is split across multiple physical files, a hash should be provided for each file. This is discussed further in the "Depth" operational consideration below.

---

<sup>1</sup> <https://www.w3.org/TR/did-core/>

<sup>2</sup> <https://csrc.nist.gov/projects/hash-functions>

- Define SBOM *author information* to reduce ambiguity (as with “supplier name”). Additional constraints on the SBOM author information may be required if it needs to be correlated to the signature applied to the SBOM.

We recommend that the following data fields proposed by NTIA are included as optional:

- *Unique identifiers for the component.* While not always readily available (e.g., when a software ecosystem doesn’t have structured identifiers), unique identifiers are important to correlate components with other sources of information and should be provided when available. Unique identifiers should require that the unique identifier namespace/scheme as well as the unique identifier itself be provided – as the unique identifier alone may be insufficient to infer the namespace/scheme. Moreover, unique identifiers should be used whenever feasible in lieu of the supplier name, component name, and component version for software identification. Unique identifiers are preferable for software identification as they better cater to the differences in how software ecosystems uniquely identify software; for example, in some ecosystems, it is important to know the hardware architecture of a component to uniquely identify it. We also encourage the use of package-URLs<sup>3</sup> when available for a component as these are precise, machine-processable, and gaining popularity in SBOM communities and tooling.
- *Dependency relationship.* SBOMs generated later in the supply chain may not be able to infer this information, which provides minimal additional context to the software integrity and vulnerability analysis use cases that we recommend prioritizing. Where provided, we recommend that a defined list of dependency relationships includes, at a minimum, CONTAINS (to describe physical composition, for example, when describing the contents of an archive file) and DEPENDENCY (to describe additional software components the software depends on).

## II. Operational Considerations

In general, we support NTIA’s consideration of the below operational issues and encourage recognition of the need for flexibility – and further work to understand the impact and define an appropriate approach – across different software products and services and use cases.

As NTIA has acknowledged, SBOM efforts to date have largely focused on software that is delivered traditionally and runs on customer premises. Cloud services have unique requirements in how they are described, how integrity is verified, and how vulnerabilities are published and shared. As such, beyond the considerations outlined below (especially in the context of delivery), we recommend that a multistakeholder working group specifically focus on addressing these and any other identified issues for cloud services. The group could be convened by NTIA in partnership with other engaged government stakeholders, including NIST and Sector Specific Agencies working on SBOM pilots and other approaches to exploring effective implementation. We would welcome the opportunity to contribute.

- *Frequency:* For software products that follow semantic versioning, an SBOM should be created for each version of a component delivered. This minimum could be imposed by the inclusion of

---

<sup>3</sup> <https://github.com/package-url/purl-spec>

cryptographic hashes, which will change with each version. For larger pieces of software, version numbers don't always change, and an SBOM might be created for a product and for each patch.

- *Depth:* At a minimum an SBOM should describe each individual file in the software component delivered so integrity can be verified. When a file is known to come from a different software component (for example, a dependency) or is composed of multiple software components (for example, archives or statically linked dependencies), then these software components should be described in the SBOM using the elements described in section I. While SBOMs that describe all transitive dependencies are the goal, achieving this goal will take time as it will require changes to build and packaging processes and may not be practical for legacy software or where upstream suppliers are not providing their own full-depth SBOMs. For this reason, we recommend requiring minimum depth SBOMs while still encouraging those that include a fuller depth.
- *Delivery:* We encourage SBOMs to be provided alongside the software component when feasible, and when an SBOM is delivered with the software component, it should be signed as recommended in the “Signing” operational consideration below. Space-constrained environments may require SBOMs to be shared out-of-band from the actual software component. The complete SBOM for a software component may be split across multiple files, for example, and dependencies may be described in their own SBOMs referenced from the software component's SBOM. (When referencing an external SBOM, sufficient information should be provided to verify the integrity of the referenced SBOM – for example, cryptographic hashes of the referenced SBOM.) Distribution or access restrictions may also mean that SBOMs need to be shared in a different manner. We recommend that NTIA in conjunction with multistakeholder communities contemplate variability in and investigate potential standards for a range of delivery scenarios, including for software delivered as cloud services (versus run and maintained by customers in their own environment).
- *Automation support:* We believe that automation is critical for accurate, complete, and verifiable SBOMs, and SBOMs must be machine-readable to achieve these goals. We encourage the use of existing machine-readable SBOM formats such as SPDX<sup>4</sup> and CycloneDX.<sup>5</sup> We are actively participating in the development of the SPDX standard. SBOMs should identify the schema and schema version they are using so that automation can process them appropriately.

In addition, we encourage NTIA to consider further operational issues in the near term:

- *Unique identification:* SBOM instances should be uniquely identifiable to allow them to be referenced unambiguously. The mechanism for unique identification may be dependent on the SBOM standard used but may include content addressing (where the hash of the SBOM is used as a unique identifier) or a namespace and identifier (where the identifier is unique within the namespace).
- *Signing:* Recipients of SBOMs need to be able to verify that they have not been tampered with and that they can verify the SBOM was produced by the SBOM author. For this reason, we

---

<sup>4</sup> <https://spdx.dev>

<sup>5</sup> <https://cyclonedx.org>

encourage requiring SBOMs to be signed with a NIST-approved signing algorithm, allowing multiple signatures as well as attached and detached signatures. We also recommend that any signature also capture, using a trusted mechanism, when the SBOM was signed. This will provide sequencing to signatures and additional evidence in the case of key compromises. Additionally, we recommend that SBOMs be signed with keys dedicated to SBOM signing rather than reusing code signing keys. This prevents key compromises affecting both runtime code validation and SBOM validation and avoids OS-specific runtime code signing requirements being imposed on SBOM signatures. We recommend that the lifetime of the keys used for signing SBOMs be limited (1 to 3 years max), and that timestamping is used to extend signature validity beyond the key lifetime. Along the same lines, standards or guidance should require rotation of the signing keys as well as timestamping keys. As a future enhancement, consider requiring multiple (2+) signatures on the SBOMs (each different algorithms) so that validation of previously signed SBOMs is resilient to compromise of the one cryptographic algorithm used for those signatures.

*Optionally Installed Components:* During the install of some software, a user is sometimes presented with a selection of options to install, and these options may contain different dependencies. Since this can affect the vulnerabilities present in an installed software package, we recommend that NTIA continues to partner with SBOM stakeholders on how to assess vulnerabilities in use cases such as this.

### III. Microsoft's Vision for Supply Chain Integrity

The above comments on Elements for an SBOM and Operational Considerations address our recommendations for the immediate needs of the EO. In this section, we outline a longer-term vision for the exchange and validation of supply chain evidence broadly, where SBOM is a class of evidence.

Microsoft envisages SBOMs, or more generally BOMs, and attestations about artifacts in those BOMs, are evidence stored as signed, immutable, non-repudiable evidence in evidence stores. Participants in a supply chain can exchange subsets of this evidence with each other to meet their obligations and ensure integrity of the supply chain. These attestations could also be published by or delivered to trusted third parties such as auditors. Adopting or creating industry standards for describing, publishing, and retrieving this evidence streamlines its exchange across organizations, allowing for rapid detection of and response to supply chain threats.

In response to NIST's call for position papers related to the EO, Microsoft submitted a paper proposing a Supply Chain Integrity Model (SCIM),<sup>6</sup> a pragmatic model for exchange of arbitrary evidence in a supply chain and the evaluation of policies based on that evidence. We consider SCIM as an approach that is responsive to the sophisticated supply chain attacks and potential use of SBOM contemplated in the "Threat model" issue highlighted in NTIA's request for comments (question 3). We believe this is achieved by generating discrete pieces of evidence throughout the supply chain that are stored in a trusted evidence store and can be evaluated for end-to-end consistency, detecting potential compromises early. Ultimately, a key purpose of an SBOM and other supply chain evidence should be to enable software consumers to trust the build environment in which software was produced. For

---

<sup>6</sup> <https://github.com/microsoft/SCIM>; [Microsoft - Executive Order - NIST workshop position paper 5- Software integrity chains Microsoft Corporation.pdf](#)

example, we believe that an SBOM for an artifact will ultimately refer directly or indirectly to the source code files and specific build tools used to produce the artifact, enabling consumers to independently validate that the build is reproducible and therefore nothing illicit was introduced into the build.

We also envisage the minimum elements in an SBOM forming a core on which additional use cases can be layered. In our collaboration with the Linux Foundation to define SPDX 3.0, we refer to these additional layers as profiles; these allow SPDX to exchange both the BOM and additional point-in-time attestations about artifacts in the BOM, such as:

- Licensing
- Vulnerabilities and mitigations
- Pedigree (for example, source code and build environment)
- Usage (for example, lifecycle)

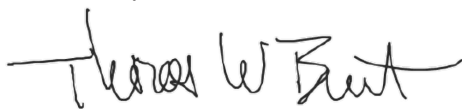
Being able to communicate both a list of vulnerabilities as well as mitigations that have been put in place would address the “Risk Management” issue highlighted in NTIA’s request for comments (question 3). Responding to this scenario has been a goal discussed in SPDX 3.0.

In addition to describing attestations, this core can be built upon to describe other types of artifacts, such as hardware, services, and data (including data used to create machine learning models). This would enable new profiles, such as hardware certification, privacy, and network policy.

In summary, it’s our belief that establishing a small set of supply chain primitives that can be extended and composed to address emerging use cases is the best way to future proof our supply chains.

Going forward, Microsoft is committed to working with NTIA and other U.S. agencies and partners to respond to a dynamic threat environment and accelerate implementation of software supply chain security and integrity best practices across the ecosystem. We welcome further opportunities for input and conversations with NTIA and others as we work collectively to foster a more secure cyberspace.

Sincerely,



Thomas W. Burt  
CVP, Customer Security and Trust  
Microsoft

DocuSigned by:  
  
F7551D62F9214CC...  
Michael Hanley  
CSO, GitHub  
Microsoft