技術ガイドライン BSI TR-03183:
製造事業者および製品に対する
サイバーレジリエンス要件

第 2 部：ソフトウェア部品表（SBOM）

文書履歴

表 *1*：文書履歴

バージョン	日付	説明
1.0	2023-07-12	初版公開時の BSI TR-03183-2 バージョン
1.1	2023-11-28	英語への翻訳；作成者、バージョン、ライセンス情報の要件を更新
2.0.0	2024-09-20	「使用用語」「詳細レベル」「説明」に定義と更新を追加；「依存関係」「明示的仕様」「発効と移行制度」セクションを追加；必須データフィールド（実際のファイル名、実行可能プロパティ、アーカイブプロパティ、構造化プロパティ）を追加；追加フィールド（締結済みライセンス）を追加；追加フィールドからオプションフィールドへソースコードハッシュを移動；オプションフィールドを追加（宣言済みライセンス）；CycloneDX の最低バージョン要件を 1.4 から 1.5 に更新；SPDX の最低バージョン要件を 2.3 から 2.2.1 に変更；実行可能コンポーネントのハッシュ値をデプロイ可能コンポーネントのハッシュ値に変更；コンポーネントの実行形式の URI をデプロイ可能形式の URI に変更；コンポーネントの定義を拡張
2.1.0	2025-08-20	「用語」「ライセンス情報」「データフィールド」「説明」に定義と更新を追加；「データフィールド」を再構成；CycloneDX の最低バージョン要件を 1.5 から 1.6 に更新、SPDX の最低バージョン要件を 2.2.1 から 3.0.1 に更新；論理コンポーネントと識別済みコンポーネントを導入・定義、BOM 参照の使用方法を明確化；「個別データフィールドのマッピング」セクションを追加

連邦情報セキュリティ局

私書箱 20 03 63

53133 ボン

E-Mail: TR03183@bsi.bund.de

インターネット: https://bsi.bund.de/dok/TR-03183-en

© 連邦情報セキュリティ庁 2023 - 2025

目次

1 序論 5
2 要件の表現方法 6
3 基本事項 7
3.1 SBOM の定義 7
3.2 使用用語 7
3.2.1 コンポーネント 7
3.2.2 論理コンポーネント 8
3.2.3 外部コンポーネント 8
3.2.4 識別されたコンポーネント（他の BOM を参照しない場合） 9
3.2.5 他の BOM で参照されるコンポーネント（参照コンポーネント） 9
3.2.6 実行可能ファイル 9
3.2.7 依存関係 9
3.2.8 ライセンス情報 9
3.2.9 ベンダー/サプライヤーと作成者の区別 10
4 SBOM フォーマット 11
5 内容要件 12
5.1 詳細レベル 12
5.2 データフィールド 12
5.2.1 SBOM 自体に必要な必須データフィールド 12
5.2.2 各コンポーネントに必要なデータフィールド 13
5.2.3 SBOM 自体に関する追加データフィールド 14
5.2.4 各コンポーネントに対する追加データフィールド 15
5.2.5 各コンポーネントのオプションデータフィールド 15
6 表現仕様 16
6.1 ライセンス識別子と表現 16
7 移行システム 18
8 附属書 19
8.1 説明 19
8.1.1 機械処理可能 19
8.1.2 コンポーネント 19
8.1.3 論理コンポーネント 19
8.1.4 実行可能プロパティ 19
8.1.5 アーカイブプロパティ 19
8.1.6 構造化プロパティ 20
8.1.7 実行可能、アーカイブ、構造化プロパティの例 20
8.1.8 コンポーネント作成者 20
8.1.9 コンポーネントのバージョン 20
8.1.10 URI、IRI、Punycode の関係 21
8.1.11 納入範囲 21

8.1.12 コンポーネントへのパス 21
8.1.13 ライセンス情報 21
8.1.14 脆弱性情報 22
8.1.15 デジタル署名 22
8.1.16 他の BOM への参照 22
8.2 個々のデータフィールドのマッピング 22
8.3 SBOM の詳細レベル 37
8.3.1 トップレベル SBOM 38
8.3.2 n 階層 SBOM 38
8.3.3 推移的 SBOM 39
8.3.4 納入品目 SBOM 39
8.3.5 完全な SBOM 40
8.4 SBOM の分類 40
8.4.1 設計 SBOM 40
8.4.2 ソース SBOM 40
8.4.3 ビルド SBOM 40
8.4.4 解析済み SBOM 41
8.4.5 展開済み SBOM 41
8.4.6 ランタイム SBOM 41
8.5 詳細情報 41
8.5.1 NTIA からの情報 41
8.5.2 CISA からの情報 41
8.5.3 CycloneDX からの情報 41
8.5.4 SPDX からの情報 41

1 序論

本技術ガイドラインは、「ソフトウェア部品表（SBOM）」の要件を規定する。SBOM は機械処理可能な文書であり、電子部品表/部品リストに相当する。コードベースを網羅的に記録するため、ソフトウェアで使用される全コンポーネントに関する情報を含みます。この情報は、基本構造からソフトウェア製品とそのコンポーネントの詳細な内訳まで、様々な広さと深さで提示可能である。SBOM の表現と配布には複数のフォーマットが存在する。

SBOM は、ソフトウェアの複雑性を透明性をもって表現し、使用されているコンポーネント（例：ライブラリ）を把握するために、すべてのソフトウェア作成者およびプロバイダが使用すべきである。この知識は、ソフトウェア管理プロセス、特に継続的な IT セキュリティプロセスとソフトウェアのライフサイクル管理にとって不可欠であり、したがって安全なソフトウェアサプライチェーンのための「ベストプラクティス」と見なされている。SBOM は公開・非公開のいずれでもよく、様々な方法で配布可能である。通常、ソフトウェア開発者は 1 つ以上のサードパーティ製コンポーネントを使用する。彼らは自社ソフトウェアの SBOM を作成・管理すると同時に、組み込まれたコンポーネントの SBOM については消費者としての役割を担いる。SBOM 情報の膨大さと構造上の差異の可能性は、各開発者にとって多大な作業負荷を意味する。これを効果的に解決できるのは自動化のみである。

SBOM 情報は、製品のコンポーネントリストを脆弱性データベースのリストと比較することで、製品が脆弱性の影響を受ける可能性があるかどうかを確認するために使用できる。ただし、SBOM には脆弱性やその悪用可能性に関する記述は一切含まれません。特定のソフトウェアバージョンの SBOM データは静的と見なされる一方、脆弱性情報は動的特性を持っている。統合コンポーネントの脆弱性が SBOM 対象製品にリスクをもたらすか否か、またその程度については、SBOM で詳細に記述すべきではない。脆弱性情報と潜在的影響の交換は、セキュリティアドバイザリ（例：CSAF[1]（共通セキュリティアドバイザリフレームワーク）や VEX[2]（脆弱性悪用可能性交換））を通じて促進されるべきである。

製品が脆弱性の影響を受けるか否かを確認するには、製品の SBOM を、CVE（Common Vulnerabilities and Exposures）やコンポーネント製造事業者・プロバイダのセキュリティアドバイザリなどの脆弱性情報と照合する必要がある。さらに、ソフトウェア自体を分析し、潜在的に影響を受けるサブコンポーネントがどのように使用されているかを特定し、自社のソフトウェア製品が影響を受けるかどうか、またその影響の程度を判断する必要がある。これは製品の脆弱性対応の一環として実施されなければならない。この分析結果は、製品のセキュリティアドバイザリまたは VEX としてソフトウェア利用者に提供される。

当初、SBOM は主にライセンス管理に利用されていた。本技術ガイドラインでは、SBOM の一般的なユースケースに対する完全な要件セットを提供するため、このユースケースも相互運用可能な形で規定している。

サイバーレジリエンス法（CRA）[3] では、SBOM の作成が義務付けられている。CRA は、デジタル要素を含む製品に対する EU の市場アクセス規制であり、そのプロバイダに脆弱性対応プロセスの継続的な運用と、製品に関する情報を透明かつ理解しやすい形で提供することを義務付けている。米国では、2021 年 5 月の大統領令 14028 号（[4]）により連邦政府が調達するソフトウェアに SBOM が既に要求されており、FDA（食品医薬品局）は 2023 年 3 月以降、新規医療機器の承認プロセスの一環として SBOM の提出を義務付けている（[5]）。

[1] https://csaf.io/
[2] https://www.ntia.gov/files/ntia/publications/vex_one-page_summary.pdf
[3] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R2847
[4] https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/大統領令-on-improvingthe-nations-サイバーセキュリティ/ Section 4
[5] https://www.congress.gov/117/bills/hr2617/BILLS-117hr2617enr.pdf Section 3305

2 要件の表現方法

本文書における「MUST」「MUST NOT」「REQUIRED」「SHALL」「SHALL NOT」「SHOULD」「SHOULD NOT」「RECOMMENDED」「NOT RECOMMENDED」「MAY」「OPTIONAL」というキーワードは、BCP 14[6]（RFC 2119[7]、RFC 8174[8]）で定義された通りに解釈される。ただし、これらがすべて大文字で表記されている場合に限る。

[6] https://www.rfc-editor.org/info/bcp14
[7] https://www.rfc-editor.org/rfc/rfc2119
[8] https://www.rfc-editor.org/rfc/rfc8174

3 基本事項

3.1 SBOM の定義

「ソフトウェア部品表」（SBOM）とは、ソフトウェア製品に使用されたコンポーネントのサプライチェーン関係および詳細情報を含む、機械処理可能なファイルである。これにより、これらのコンポーネントに関する情報の自動処理が可能となる。これには、いわゆる「主要コンポーネント」と使用された（例：外部／サードパーティ）コンポーネントの両方が含まれる。

SBOM は、特定の最低限の情報を含まなければならない（第 5 節参照）。第 5 節で詳述される対象コンポーネント間のあらゆる関係を記述する。コンポーネントは、SBOM 内のそのレコードが第 5.2 節で規定される適用可能なデータフィールドを全て含む場合に「完全に記述されている」とする。本技術ガイドラインに準拠する SBOM は、矛盾を生じさせない限り、必要に応じて拡張・詳細化することが許容される。

注記：この要件は、拡張情報をプロバイダしながらも、本技術ガイドラインへの適合性チェックを確定的に実施できるようにするものである。セクション 5.2 で規定されたデータフィールドを含まない拡張情報は、本技術ガイドラインへの適合性ステータスを変更することなく、常に SBOM から削除できる。したがって、ツールは準拠していない情報をフラグ付けでき、SBOM 作成者は拡張情報に関する報告されたエラーを無視するか修正するかを選択できる。コンテキストが欠落している場合など、どの情報が拡張情報であるかを自動的な方法で判断することはしばしば困難である。したがって、そのような判断は SBOM 作成者に委ねられる。

各ソフトウェアバージョンごとに個別の SBOM を生成しなければならない。特定のソフトウェアバージョンに対する SBOM の更新版は、含まれるコンポーネントに関する追加情報が提供された場合、または SBOM データの誤りが修正された場合に限り生成しなければならない。したがって、いずれかのコンポーネントが変更された場合、このコンポーネントに新しいソフトウェアバージョンを割り当てる必要がある。また、デリバリーアイテムの SBOM の場合、依存するすべての親コンポーネントにも同様に適用される。

SBOM は脆弱性情報を含んではならない。SBOM データは変更されないソフトウェアに対して静的であるためである。ただし、SBOM 記述フォーマットは脆弱性情報を含める機能を提供する場合がある。それでも、SBOM と脆弱性情報の両方を含む文書は本技術ガイドラインに準拠しない。附属書 8.1.14 も参照のこと。

3.2 使用用語

使用用語の説明と例については、附属書の 8.1 節を参照のこと。

3.2.1 コンポーネント

本技術ガイドラインで使用される「コンポーネント」とは、論理コンポーネント（3.2.2 節参照）、単一の実行可能ファイル、またはアーカイブファイルを指す。したがって、これらは SBOM においてコンポーネントとして表現されなければならない。さらに、その他のファイルもコンポーネントとして表現することが可能である。アーカイブも実行可能ファイルとなり得ることに留意すること。

プライマリコンポーネント、すなわち製品自体および本 SBOM の観点における依存関係ツリーのルートも、本技術ガイドラインで使用される「コンポーネント」である。

複数のコンポーネントを構成要素として含むコンポーネントで、元のコンポーネントを特定できない場合、セクション 5.2.2、5.2.4、5.2.5 に従い、適切な情報と共に元のコンポーネントをリストすべきである。コンポーネントの組み立て方法により利用できない情報は、必須データフィールドとして指定されていても省略しなければならない。例：コンポーネントのハッシュ値やファイル名。

コンポーネントに関する情報の収集には、以下の原則が適用されなければならない：

コンパイル済みコードの場合

リンカーの実行中に使用されたコンポーネントは、代替コンポーネントや実装が使用可能であった場合でも、必ず記載しなければならない。

技術的な理由によりリンカ実行が行われない場合、解釈コードセクションが適用される。

両方の記述は、静的リンクおよび動的リンクされたコンポーネント/ライブラリに適用される。

インタプリタコードの場合

コンポーネントが納品物の一部である場合、そのバージョン番号は正確に必要なバージョンを反映しなければならない。

コンポーネントが納入品の一部でない場合、コンポーネントの作成者が定義した最小必要バージョンを反映すべきである。この際、事実上の最小技術要件以外の要素も考慮すべきである。これは、コンポーネント作成者がサポート終了版や既知のセキュリティ脆弱性があるバージョンをスキップすべきであることを意味する。最低必要バージョンは、テストに使用された最低バージョン（例：ソフトウェア開発中に使用されたバージョン）によって決定されてもよい。

単一の依存関係に複数実装が存在する場合、「最小公倍数」または標準実装を使用すべきである。例：

- スクリプトがすべての Bourne Shell 互換シェルで実行可能な場合、SBOM で指定されるコンポーネントは Bourne Shell であるべきである。
- スクリプトが任意の Python 実装で実行可能な場合、代替実装（Pyston）ではなく標準実装（Python）を使用しなければならない。
- Java プログラムが任意の Java ランタイム環境（JRE）実装で実行可能な場合、プロプライエタリ実装ではなくオープンソースソフトウェア実装（OpenJRE）を使用しなければならない。

3.2.2 論理コンポーネント

論理コンポーネントとは、複数のコンポーネントを統合する抽象化レベルである。例えば、製品名を識別するため（複数のファイルをアプリケーションに論理的に関連付ける手段として。附属書の 8.1.3 節も参照）。

論理コンポーネントを完全に記述するには、セクション 5.1 で規定されている以下のデータフィールド（セクション 5.2 参照）のみを必ず指定しなければならない：

- コンポーネント作成者
- コンポーネント名
- コンポーネントバージョン
- 他のコンポーネントへの依存関係
- 配布ライセンス
- その他の一意の識別子
- 元のライセンス
- 有効なライセンス
- **security.txt** の URL

3.2.3 外部コンポーネント

「外部コンポーネント」とは、コンポーネント作成者がプライマリコンポーネントの作成者と異なるコンポーネントを指す。

コンパイル済みコードおよびインタプリタ型コードの場合に記載すべき情報の割り当てについては、セクション 3.2.1 の一部として説明されている。

3.2.4 識別されたコンポーネント（他の BOM を参照しない場合）

SBOM の詳細レベルに応じて、すべての部品を完全に記述する必要はなく、単に識別するだけでよい場合もある。
例：附属書 8.3.4 節を参照。

識別が必須であり、完全な記述が要求されないコンポーネントについては、以下のデータフィールド（5.2 節参照）を 5.1 節で規定された通りに対処しなければならない：

- コンポーネント作成者
- コンポーネント名
- コンポーネントバージョン
- その他の一意の識別子

3.2.5 他の BOM で参照されるコンポーネント（参照コンポーネント）

別の BOM に記録され、SBOM によって参照されるコンポーネント（完全に記述されているか単に識別されているかを問わない）については、参照されるコンポーネントのコンポーネント記述から以下のデータフィールド（5.2 節参照）を抽出し、参照元 BOM に加えて参照元 SBOM に追加しなければならない：

- コンポーネント作成者
- 部品名
- コンポーネントバージョン

注記：参照が解決できない場合、または参照された BOM へのアクセスに使用できない場合、参照されたコンポーネントを識別するためにこの情報が必要である。

参照元 BOM と参照先 BOM の両方に矛盾するコンポーネント情報が存在する場合、参照先 BOM の情報が優先される。

前述のデータフィールド（作成者、名称、バージョン）および他 BOM への参照以外の情報は、矛盾を避けるため参照元 SBOM に記録すべきではない。ただし、当該フォーマットで有効な SBOM を生成するために SBOM フォーマット仕様で要求される追加データフィールドは除きます。

3.2.6 実行可能ファイル

本技術ガイドラインで使用される「実行可能ファイル」とは、コンピュータによって直接またはランタイムシステムを介して実行されるコードを含むあらゆるファイルを指す。

3.2.7 依存関係

本技術ガイドラインにおいて、依存関係とは、あるコンポーネントから別のコンポーネントへの指向性のある要求を指す。これは、コンポーネントが（技術用語としての）依存関係であるか、あるいはコンポーネントに静的リンクまたは埋め込みなどの形で含まれているかどうかにかかわらず適用される。

3.2.8 ライセンス情報

本技術ガイドラインでは、ライセンス情報を以下の 3 つのカテゴリーに分類する：

- **オリジナルライセンス**
  オリジナルライセンスとは、コンポーネントの作成者によって付与されたすべてのライセンスを指す。

- **配布ライセンス**
  配布ライセンスとは、現在の SBOM に記録されたコンポーネントをライセンシーが使用できるすべてのライセンスを指す。
- **実効ライセンス**
  有効ライセンスとは、現在の SBOM の作成者であるライセンシーがコンポーネントを使用する際に適用されるライセンスを指す。

「締結済みライセンス」および「宣言済みライセンス」という用語は、異なる SBOM 標準や実装間で定義や使用法が統一されていない。不一致や混乱を避け、容易なマッピングを可能にするため、本技術ガイドラインでは異なる用語を使用する。

注記：「有効なライセンス」は現在の SBOM の作成者によって設定されるため、SBOM をマージする際には新たに設定する必要がある。ただし、このプロセスにより同じ「有効なライセンス」が結果として得られる場合がある。

3.2.9 ベンダー/サプライヤーと作成者の区別

本技術ガイドラインでは、「ベンダー」と「作成者」を区別する。なぜなら「製造事業者」という用語は、ドイツ語の「Hersteller」が通常解釈されるように、「自社作成ソフトウェアのベンダー」という意味でこれら二つの役割を包含すると解釈される可能性があるためである。

「ベンダー」（ドイツ語:「Anbieter」）は、ソフトウェアまたはコンポーネントを提供する事業体の役割を指す。代替として、必ずしも商業的背景を伴わない場合、「サプライヤー」（ドイツ語:「Lieferant」）という用語が使用される。

「作成者」（ドイツ語：Ersteller）は、ソフトウェアまたはコンポーネントを著作者または作成した事業体の役割を指す。代替として「作者」（ドイツ語：Autor）という用語が用いられる。

CRA は市場アクセス規制であり、本技術ガイドラインは技術要件を規定するため、 本技術ガイドラインでは、CRA で定義された用語とは異なる用語を使用する。したがって、本技術要件はこれら当事者の役割とは無関係であるため、本技術ガイドラインでは「経済的事業者」、「頒布事業者」、「輸入事業者」といった用語は言及しない。これらの技術要件は、それを満たす役割とは独立している。

4 SBOM フォーマット

新規生成または更新された SBOM は、JSON 形式または XML 形式でなければならず、かつ以下の仕様のいずれか、指定されたバージョンのいずれかに準拠した有効な SBOM でなければならない：

- CycloneDX[9] 、バージョン 1.6 以上
- System Package Data Exchange (SPDX)[10] 、バージョン 3.0.1 以上これらの仕様は、公式にリリースされたバージョンのみを使用しなければならない。

本技術ガイドラインの移行システムは第 7 節に規定されている。

第 5.1 節および第 3.2.5 節の情報モデルと、前述の SBOM フォーマット仕様のデータフィールドとの対応関係は、附属書第 8.2 節に記載されている。

[9] https://cyclonedx.org/specification/overview/
[10] https://spdx.dev/specifications/

5 内容要件

5.1 詳細レベル

本技術ガイドラインに準拠する SBOM では、再帰的依存関係解決を、少なくとも納入範囲に含まれる各コンポーネントについて**、各**下位パス（附属書 8.1.12 参照）**において**、納入範囲外の最初のコンポーネント（附属書 8.1.11 参照；「納入品目 SBOM」については附属書 8.3.4 参照）まで含めて実施しなければならない。。

注記：納品範囲外の依存関係の最初のレベルは（「識別されたコンポーネント」については 3.2.4 節参照）、これらの依存関係を明確に関連付けるために識別されなければならない。この識別がなされない場合、例えば納品項目の SBOM とそれが実行される環境の SBOM というように、2 つの SBOM を正しく連結するために必要な情報が欠落する。

プライマリコンポーネントが異なるメタ情報を持つコンポーネントの複数インスタンスに依存する場合、これら全てのインスタンスは個別のメタ情報と共に個別に記載されなければならない。

使用コンポーネントの SBOM は、本技術ガイドラインに準拠している場合に限り、プライマリコンポーネントの SBOM に統合する代わりに参照することが許容される。プライマリコンポーネントの SBOM プロバイダは、参照される SBOM の可用性について責任を負う。

SBOM に記録すべき必須データフィールドを決定する際には、以下の順序が適用される：

1. 参照コンポーネント（3.2.5 節参照）
2. 識別コンポーネント（3.2.4 節参照）
3. 完全記述コンポーネント（5.2 節参照）

参照される SBOM 内のコンポーネントは、参照コンポーネントであるか、プライマリコンポーネントの SBOM の詳細レベルで要求される全てのデータフィールドを提供するプロバイダであるか、または両方を満たさなければならない。

本技術ガイドラインに準拠する SBOM は、ビルドプロセス中に利用可能な情報、またはビルドプロセスが存在しない場合には同等の情報を含まなければならない（ビルド SBOM に関する詳細は附属書 8.4.3 節を参照）。

注記：人間による識別を容易にするため、SBOM 自体のファイル名はプライマリコンポーネントの名称を反映してもよい。

5.2 データフィールド

5.2.1 SBOM 自体に必要な必須データフィールド

各 SBOM は、少なくとも以下の情報を含まなければならない：

表 *2: SBOM* 自体に必要なデータフィールド

データフィールド	説明
SBOM の作成者	SBOM を作成した事業体のメールアドレス。メールアドレスが利用できない場合、これは「Uniform Resource Locator (URL)」でなければならない。例：作成者のホームページまたはプロジェクトのウェブページ。
タイムスタンプ	仕様書（第 4 節参照）に基づく SBOM データコンパイルの日時。

データフィールド	説明
	注記：タイムスタンプは UTC（「ズールー」時間）のみを使用することを推奨する。

注記：SBOM フォーマット仕様によっては、当該フォーマットで有効な SBOM を生成するために追加のデータフィールドが必要となる場合がある。

5.2.2 各コンポーネントに必要なデータフィールド

SBOM に含まれる各コンポーネントについて、少なくとも以下の情報をプロバイダしなければならない：

表 *3*：各コンポーネントに必要なデータフィールド

データフィールド	説明
コンポーネント作成者	該当コンポーネントを作成し、該当する場合維持管理する事業体のメールアドレス。メールアドレスが利用できない場合、これは「Uniform Resource Locator (URL)」でなければならない。例：作成者のホームページまたはプロジェクトのウェブページ。 附属書の 8.1.8 節も参照のこと。 注記：コンポーネントの作成者が当該コンポーネントを現在も維持している場合で、コンポーネント統合時とは異なる名称で活動している場合は、現在の名称を使用してもよい。
コンポーネント名	コンポーネント作成者がコンポーネントに割り当てた名称。名称が割り当てられていない場合、実際のファイル名でなければならない。 注記：コンポーネント名がバージョン間で変更された場合、コンポーネント統合時に有効であったコンポーネント名を使用しなければならない。
コンポーネントバージョン	作成者が、以前に作成されたバージョンに対するコンポーネントの変更を指定するために使用する識別。バージョンを決定するには、以下の点をこの順序で適用する： 1. 既存の識別子はこの目的で変更してはならない。 2. セマンティックバージョニング（[11]）またはカレンダーバージョニング（[12]）に基づく識別子を、バージョン管理方式を決定する者が使用すべきである。これは通常、コンポーネント作成者である。 3. バージョンが割り当てられていない場合、これは RFC 3339[13] セクション 5.6 に準拠した日時形式で表されるファイルの変更日時でなければならない。作成時刻を決定するには、ファイルのメタデータを参照しなければならない。 附属書のセクション 8.1.9 も参照のこと。
コンポーネントのファイル名	コンポーネントの実際のファイル名（ファイルシステム上のパスではない）。3.2.1 節も参照のこと。

[11] https://semver.org/
[12] https://calver.org/
[13] https://www.rfc-editor.org/rfc/rfc3339

データフィールド	説明
他のコンポーネントへの依存関係	セクション 5.1 の要件に基づき、このコンポーネントが直接依存するすべてのコンポーネント、またはセクション 3.2.1 の要件に基づきこのコンポーネントが含むすべてのコンポーネントの列挙。 さらに、この列挙の完全性は明確に示されなければならない。
配布ライセンス	ライセンシーが使用できる、コンポーネントの配布ライセンス。詳細はセクション 6.1 および 8.1.13 を参照。
展開可能なコンポーネントのハッシュ値	暗号的に安全なチェックサム（ハッシュ値） 展開済み／デプロイ可能なコンポーネント（例：大容量記憶装置上のファイル）の暗号的に安全なチェックサム（ハッシュ値）を SHA-512 で表したもの。3.2.1 節も参照のこと。
実行可能プロパティ	コンポーネントが実行可能かどうかを記述する。可能な値は 「実行可能」および「非実行可能」；附属書セクション 8.1.4 も参照
アーカイブ属性	コンポーネントがアーカイブであるかどうかを記述する。可能な値は 「archive」および「no archive」；附属書の 8.1.5 節も参照のこと
構造化プロパティ	コンポーネントが構造化ファイルであるかどうかを記述する。すなわち、内容のメタデータが依然として存在するかどうか（セクション 3.2.1 参照）。可能な値は「構造化」 および「非構造化」。附属書の 8.1.6 節も参照のこと コンポーネントが構造化部分と非構造化部分の両方を含む場合、値「構造化」を使用しなければならない。

注記：SBOM フォーマット仕様では、このフォーマットで有効な SBOM を生成するために追加のデータフィールドが必要となる場合がある。

5.2.3 SBOM 自体に関する追加データフィールド

各 SBOM は、以下の情報が存在し、かつ特定のデータフィールドに対する SBOM フォーマット仕様の要件を満たす場合、追加で以下の情報を含まなければならない：

表 *4: SBOM* 自体に対する追加データフィールド

データフィールド	説明
SBOM-URI	この SBOM の「Uniform Resource Identifier (URI)」

5.2.4 各コンポーネントに対する追加データフィールド

SBOM に含まれる各コンポーネントについて、以下の情報は、存在し、かつ特定のデータフィールドに対する SBOM フォーマット仕様の要件を満たす場合に、追加で提供されなければならない：

表 *5*：各コンポーネントの追加データフィールド

データフィールド	説明
ソースコード URI	コンポーネントのソースコードの「一意のリソース識別（URI）」、例えばリポジトリ内の利用されたソースコードバージョンの URL、またはバージョンを指定できない場合は利用されたソースコードリポジトリ自体。
コンポーネントのデプロイ可能な形式の URI	コンポーネントのデプロイ可能（例：ダウンロード可能）な形式を直接指す「一意のリソース識別子（URI）」。
その他の一意の識別子	コンポーネントを識別したり、関連データベース（例：共通プラットフォーム一覧表（CPE）や Package URL (purl)）で検索するために使用できるその他の識別子。
元のライセンス	コンポーネントの作成者が付与したライセンス。詳細は 6.1 節および 8.1.13 節を参照。

5.2.5 各コンポーネントのオプションデータフィールド

各 SBOM は、以下の情報が存在し、かつ特定のデータフィールドに対する SBOM フォーマット仕様の要件を満たす場合、追加で含めることができる：

表 *6*：各コンポーネントのオプションデータフィールド

データフィールド	説明
有効なライセンス	現在の SBOM の作成者であるライセンシーがコンポーネントを使用する際に適用されるライセンス。詳細は 6.1 節および 8.1.13 節を参照。
コンポーネントのソースコードのハッシュ値	コンポーネントのソースコードの暗号的に安全なチェックサム（ハッシュ値）。複数のソースファイルまたはソースコードツリーのハッシュ値を生成する方法、およびその際に使用するハッシュアルゴリズムについては、まだ決定されていない。
security.txt の URL[14]	コンポーネント作成者の **security.txt** の「Uniform Resource Locator (URL)」を含む。

注：このリストは網羅的なものではない。

[14] https://www.rfc-editor.org/rfc/rfc9116

6 表現仕様

本節では、第 5 節の内容を詳細に明示的に規定する。

6.1 ライセンス識別子と表現

ライセンス識別に関する原則：

1. ライセンスは、適切な SPDX ライセンス識別子またはそのような識別子に基づくライセンス表現によって（「名前を付ける」という意味で）「参照」されなければならない。この参照は、いわゆる「ライセンスおよびライセンス表現の識別」[15] によって行われる。

   SPDX および CycloneDX 仕様では、SBOM にコンポーネントのライセンステキストを含めることが許可されているが、これはライセンス識別子の代替として使用してはならない。

2. SPDX で定義されていないライセンス識別子の表示

   適切なライセンス識別子が、SPDX が管理するリストにも見当たらず、定義済みライセンス識別子に基づく SPDX ライセンス表現によって構築することもできない場合、次にライセンスデータベース「Scancode LicenseDB AboutCode」[16] を参照しなければならない。このデータベースの識別子は、その起源を示すため、SPDX ライセンス識別子において接頭辞 **LicenseRef-scancode-[...]** を使用する。

3. 既存の識別子または既存の識別子に基づくライセンス表現に割り当てられないライセンス

   確立された識別子リストのいずれからも適切なライセンス識別子が見つからない場合、「附属書 B. SPDX ライセンス表現」[17] に従い、プレフィックス **LicenseRef-<licence_inventorising_entity>-[...]** を使用し、（各 LicenseRef 名前空間内で）一意のライセンス識別子を割り当てる必要がある。

ライセンス識別においては、以下のライセンス類似性の原則を適用しなければならない：

- ライセンスは、「附属書 C. ライセンスリスト照合ガイドライン」[18] に従い、他のライセンスにマッピング可能な場合に類似しているとみなされる。
- ライセンス文中のプレースホルダーとテンプレート

  ライセンステキスト内にプレースホルダーやテンプレートが存在する場合、これらのプレースホルダーやテンプレートを個別化された内容で置換することは、ライセンスの変更と解釈してはならず、同じライセンス識別子を割り当てる必要がある。

- SPDX 演算子とライセンス連結

  複数ライセンス付与、ライセンス選択、ライセンス例外のライセンス表現は、SPDX 演算子を用いてマッピングされなければならない。ライセンス演算子はライセンス識別子を連結する。許可される演算子は「附属書 B. SPDX ライセンス表現」に従って選択されなければならない。

- ライセンスの例外条項

[15] https://spdx.org/licenses/
[16] https://scancode-licensedb.aboutcode.org/
[17] https://spdx.github.io/spdx-spec/v3.0.1/附属書/spdx-license-expressions/
[18] https://spdx.github.io/spdx-spec/v3.0.1/annexes/license-matching-guidelines-and-templates/

ライセンス条文に例外条項が付随する場合、許可された SPDX 演算子に従い **WITH** を使用してライセンス識別子に付随させなければならない。例外条項の名称は、ライセンス識別に関する要件[19] に準じた識別子で記述しなければならない。

- テキストの修正

  既知のライセンス識別子を持つライセンスのテキストと比較して、ライセンステキストの文言がわずかに変更されている場合、変更が重大でない限り、変更されたライセンスにはこのライセンス識別子を使用すべきである。例としては、責任条項の追加または削除、既存の商標に関する注記などが挙げられる。

[19]https://spdx.org/licenses/exceptions -index.html

7 移行システム

BSI のウェブサイトで入手可能な本技術ガイドラインのバージョン番号が最も高いバージョンを「最新版」と呼ぶ。本技術ガイドラインに準拠するためには、SBOM 生成に最新版を使用しなければならない。ただし、表 1 の文書履歴に基づき新たな「最新版」が発行された後 6 か月間は、直前のバージョンに限り本技術ガイドラインに準拠した SBOM 生成に適用することが許容される。それ以前のバージョンは適用してはならない。

特に定義されていない限り、本 SBOM バージョンの提供日時点で本技術ガイドラインに準拠していた SBOM バージョンは、その後本技術ガイドラインの新バージョンが発行された場合でも、引き続き本技術ガイドラインに準拠しているものとみなされる。したがって、SBOM の利用者は、提供時点において本技術ガイドラインに準拠していた SBOM バージョンを解釈できるべきである。

8 附属書

本節では、補足的な説明情報を提供する。

第 3 節から第 5 節ではソフトウェア部品表（SBOM）の内容、範囲、形式に関する規範的要件を定めているが、本節ではこれらの要件に関する有益な情報を提供する。これには特定の要件の理由、使用される用語の追加説明、および例が含まれる。

8.1 説明

本節では、本技術ガイドラインにおける特定の用語または定義に関する詳細情報を提供する。

8.1.1 機械処理可能

本技術ガイドラインでは、SBOM は機械処理可能なファイルと定義される。これは、機械がコンテンツを作成、読み取り、変更、処理、分析、評価し、データに基づいて動作できることを意味する。コンテンツ自体は明確に定義され、構造化されている。「機械可読」という用語は複数の解釈が可能であるため、使用されません。

8.1.2 コンポーネント

本技術ガイドラインにおいて「コンポーネント」とは、通常単一の実行可能ファイル[20] またはアーカイブファイルを指す。ただし、他の SBOM へのリンクを目的とした抽象化レベルとして論理コンポーネントを使用することも推奨される。

注記：

- アーカイブも実行可能ファイルとなり得る。
- 本技術ガイドラインが適用される対象を明確に定義するため、「ファイル」という用語の使用が必要である。

8.1.3 論理コンポーネント

論理コンポーネントは、抽象レベルでコンポーネントを扱う必要がある場合に、オペレーティングシステム、アプリケーション、枠組み、コンテナなどとして定義される。例：インストール済みオペレーティングシステム（論理コンポーネント）への依存関係（インストール可能イメージ（アーカイブファイル）との対比）。

別の例として、コンテナ内の PHP アプリケーションの SBOM は、抽象化レベルなしではコンテナ自体と各実行可能 PHP ファイルのみを列挙する。元の PHP アプリケーションとしての製品情報は失われる。論理コンポーネントは、その情報を表現する場を提供し、PHP アプリケーションの既存 SBOM を参照することも可能にする。

8.1.4 実行可能プロパティ

実行可能ファイルの例には、コンパイル済みバイナリ（「実行可能ファイル」）、インタプリタ型コード（例：Python、Shell、Java）、共有ライブラリが含まれます。この用語の対象外となるものには、設定ファイル、グラフィックファイル、ドキュメントなどがある。

注記: このプロパティは、悪意のあるコードを含む可能性のあるファイルを識別するために重要である。

8.1.5 アーカイブプロパティ

アーカイブは複数のコンポーネントの集合体である。

[20]参照: https://en.wikipedia.org/wiki/Executable

注記：

- アーカイブの圧縮はこのプロパティを変更しない。
- このプロパティは、分解が必要なファイルを識別するために重要である。

8.1.6 構造化プロパティ

構造化アーカイブにはメタデータが含まれており、後から元の構成要素を特定できる。例としてはコンテナ、パッケージ、ISO イメージ、.zip、.tar、.tar.gz、.7z 形式のアーカイブなどがある。

非構造化アーカイブはこのようなメタデータを含まず、つまり構造に関する情報がファイルに埋め込まれていない。例としてはファームウェアイメージや、元の構成要素に分解できないその他のアーカイブが挙げられる。

注記:

- 静的リンクされたライブラリを含むバイナリは非構造化の実行可能ファイルである。ただし、自己解凍型アーカイブは例外で、これらは構造化アーカイブでありながら実行可能ファイルであり、静的リンクされた解凍アルゴリズムを含む可能性がある。
- この特性は、分解可能なファイルを識別するために重要である。

8.1.7 実行可能、アーカイブ、構造化プロパティの例

表 *7:* 実行可能、アーカイブ、構造化プロパティの例

例	実行可能	アーカイブ	構造化	コメント
設定ファイル	–	–	– / √	SBOM に記載する必要はない。
rpm パッケージ	–	√	√	
Python スクリプト	√	–	–	
SBOM を統合したバイナリ	√	–	√	
ファームウェアイメージ	√	– / √	–	
自己解凍アーカイブ	√	√	√	

8.1.8 コンポーネント作成者

コンポーネントのソースファイルが複製された場合、コンポーネントのメンテナがプロジェクト名を変更した場合、またはコンポーネントのソースコードの内容（該当する場合）が変更された場合、メンテナはコンポーネント作成者の役割を引き継ぎます。

8.1.9 コンポーネントのバージョン

一般的に、広く認知されたバージョン管理方式の使用が推奨される。目的は、バージョン番号が単調増加するシーケンスを維持することである。コンポーネントの作成者はバージョン管理方式を完全に任意に選択できるが、サプライチェーン下流の事業体はこれを変更すべきではない。

8.1.10 URI、IRI、Punycode の関係

注記：国際化リソース識別（IRI）のみが利用可能な場合、それを URI に変換する必要がある。完全修飾ドメイン名（FQDN）部分には、Punycode[21] を使用しなければならない。

8.1.11 納入範囲

ソフトウェア製品の納入範囲には、当該ソフトウェア製品と共に納入される、供給者またはサードパーティによって作成されたソフトウェアの全構成要素が含まれる。別途取得または入手が必要なソフトウェア構成要素は、この用語には含まれない。

8.1.12 コンポーネントへのパス

コンポーネントへのパスとは、当該コンポーネントに至るノードを接続するエッジの組として定義される。パスはプライマリコンポーネントから始まり、当該コンポーネントで終わる。複数の互いに排他的コンポーネントが当該コンポーネントに依存する場合、単一コンポーネントへの複数のパスが存在することに留意すること。パスは 2 つのノードを接続し、他のノードを経由する場合がある。したがってパスは特定のノードの連鎖を明確に示す。結果として、コンポーネントが n 個の直接依存関係を持つ場合、このノードを経由するパスは少なくとも n 個存在し、それぞれが当該コンポーネントの異なる直接依存関係を経由する。可視化については 8.3 節を参照のこと。

8.1.13 ライセンス情報

コンポーネントに必要なデータフィールドにはライセンス情報が含まれる。一方で、ライセンス情報の取り扱いは SBOM の一般的なユースケースである。他方で、ライセンス情報は脆弱性対応において極めて重要となり得る。例えば、コンポーネントのライセンスが脆弱性緩和のためのコード修正をユーザーに許可しない場合などである。

上流コンポーネントの元のライセンスは、そのコンポーネントを利用する下流コンポーネントの作成者が割り当てる配布ライセンスと異なる場合がある。例として、コンポーネント作成者が主要なライセンシーに対し、互いに排他的な複数のライセンスセットから選択することを強制する場合が挙げられる。典型例が Qt であり、 の主要ライセンシーは GPL とプロプライエタリライセンスのいずれかを選択せねばならず、選択したライセンスのみがサプライチェーン下流へ継承される。したがって配布ライセンスは元のライセンスと異なる場合がある。

セクション 5.1 における異なるメタ情報を持つ複数コンポーネントインスタンスに関する記述は、ライセンス情報のみが異なる場合にも適用される。

一般的に、「宣言済みライセンス」と「締結済みライセンス」の両方のライセンスフィールドは、使用されているフォーマット仕様（特定のバージョンの CycloneDX または SPDX）に従って記入する必要がある。これらのフォーマット仕様は「宣言されたライセンス」と「締結されたライセンス」の定義が若干異なる点に留意してください。ただし、両仕様ともライセンスフィールドを区別する際に、この情報がどのように取得されたかに焦点を当てている。BSI TR-03183-2 は、SBOM 作成者から原典（サプライチェーンの「先端」）までのライセンス状況、および SBOM 作成者からサプライチェーン下流（「下流ライセンス」）へのライセンス状況に重点を置いている。これは、ソフトウェア（コンポーネント）の取得者が、その使用・分析・変更・配布の自由度を判断し、またソフトウェア（コンポーネント）を元のソースからどのライセンスで入手できるかを決定する上で関連するライセンス情報だからである。

したがって、本技術ガイドラインでは、ライセンス情報に関して意図的に異なる用語と定義を両フォーマット仕様と区別して規定している。これは「宣言されたライセンス」および「締結されたライセンス」という用語に第三の定義を追加して過負荷をかけることを避けるためである。BSI TR-03183-2 で規定されるライセンス情報は、以下の通り対応するフォーマッ

[21] https://www.rfc-editor.org/rfc/rfc3492

ト仕様にマッピングされなければならないが、利用するフォーマット仕様のバージョンによるこれらのフィールドの定義に違反しないよう注意が必要である。

8.1.14 脆弱性情報

本技術ガイドラインにおける SBOM の定義では、脆弱性情報は SBOM に含まれないと規定している。特定のソフトウェアバージョンの脆弱性は時間とともに変化するが、SBOM の重要な情報（依存関係など）は静的である。脆弱性情報を SBOM に含めると、脆弱性情報が更新されるたびに、この静的データが変更されずにソフトウェアサプライチェーン全体に不必要に伝播されることになる。したがって、SBOM フォーマット仕様がサポートしている場合でも、SBOM に脆弱性情報を含めないことが求められる。脆弱性情報を配布するための推奨フォーマットは CSAF（プロファイルとして VEX も含む）である。

8.1.15 デジタル署名

理想的には、SBOM はデジタル署名され、取得者がその真正性を検証できるようにすべきである。

8.1.16 他の BOM への参照

SBOM の作成者が外部 BOM を参照して SBOM 情報を参照する場合、当該作成者は参照先 SBOM 情報の可用性（例：当該情報の独自コピーの提供）について責任を負う。目的は、SBOM 作成者が完全な情報を自ら提供したかのように SBOM 情報を提供することである。

8.2 個々のデータフィールドのマッピング

以下の表は、セクション 5 で規定されたデータフィールドの必須情報を、SPDX および CycloneDX フォーマットにマッピングしたものである。両フォーマットの構造が異なるため、このマッピングにはいくつかの課題がある。例えば、SPDX はファイルとパッケージを暗黙的に区別するのに対し、CycloneDX はコンポーネントタイプを含みます。また、両フォーマットとも本技術ガイドラインで要求される値の一部が欠落している。この問題を解決するため、該当するデータフィールドは BSI の分類体系[22] を用いてキーと値のペアとして表現される。

注記：

- フォーマット仕様（SPDX または CycloneDX）やその他の規制で要求されるデータフィールドは維持される。さらに、これらの要求事項および本技術ガイドラインの要求事項を超えるデータフィールドは、SBOM 作成者の裁量で含めることができる。これには、ハッシュ値などのデータフィールドの追加インスタンスも含まれる。
- 以下の表における「OR」および「XOR」の記述は、規範的テキストのセクション 5.2 における「or」または「(n)either / (n)or」の記述、もしくは対応する SBOM フォーマット仕様から直接導出されたものである。例えば CycloneDX では、主要コンポーネントのデータフィールドは他のコンポーネントとは異なるタグ階層位置で記述される。
- 以下の表における「...」（省略記号）は、SBOM 作成者が挿入すべきデータフィールドの値を示すプレースホルダーである。
- 以下の表における「example_URI_xx」は、SPDX 要素間の関連性を示す「spdxId」のプレースホルダーである。これらの「spdxId」は SBOM 作成者が挿入する必要がある。

表 *8: SBOM* 自体に必要なデータフィールドのマッピング

[22] https://github.com/BSI-Bund/tr-03183-cyclonedx-property-taxonomy

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

SBOM の作成者
SBOM の作成者
	{
	"type": "CreationInfo",
	"createdBy": ["*example_URI_01*"]
	},
	{
	"type": "Person",
	XOR
	"Organization",
	"spdxId": "*example_URI_01*",
	"externalIdentifiers": [{
	"type": "外部識別子",
	"externalIdentifierType": "email",
	識別: "*...@...*"
	XOR
	"type": "ExternalIdentifier",
	"externalIdentifierType": "urlScheme",
	識別: "*https://...*"
	}]
	}
		{
		"metadata": {
		"製造事業者": [{
		"url": "*...*"
		XOR
		"contact" [{
		"email": "*...*"
		}]
		}]
		}
		}

タイムスタンプ	{
	"type": "CreationInfo",
	"created": "*...*"
	}
		{
		"metadata": {
		"timestamp": "*...*"
		}
		}

表 *9:* 各コンポーネントに必要なデータフィールドのマッピング

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

コンポーネント作成者	{
	"type": "ソフトウェアパッケージ",
	"originatedBy": ["*example_URI_01*"]
		{
		"metadata" {
		"component": {

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

	},
	{
	"type": "Person",
	XOR
	"組織",
	"spdxId": "*example_URI_01"*,
	"externalIdentifiers": [{
	"type": "外部識別子",
	"externalIdentifierType": "email",
	識別: "*...@...*"
	XOR
	"type": "ExternalIdentifier",
	"externalIdentifierType": "other",
	識別: "*https://...*"
	}]
	}
		"製造事業者": [{
		"url": "*...*"
		XOR
		"contact" [{
		"email": "*...*"
		}]
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"製造事業者": [{
		"url": "*...*"
		XOR
		"contact" [{
		"email": "*...*"
		}]
		}]
		}]
		}

コンポーネント名	{
	"type": "software_Package",
	"name": "*...*"
	}
		{
		"metadata" {
		"component": {
		"name": "*...*",
		"group": "*...*"
		}
		}
		}
		XOR
		{

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

		"components": [{
		"name": "*...*",
		"group": "*...*"
		}]
		}
		注記: 「group」は本 TR および CycloneDX 仕様のいずれにおいても必須データフィールドではない。ただし、異なるプロジェクトによって構築された同名のコンポーネントを区別することは一般的に合理的と考えられる。

コンポーネントバージョン	{
	"type": "software_Package",
	"software_packageVersion": "*...*"
	}
		{
		"metadata" {
		"component": {
		"version": "*...*"
		}
		}
		}
		XOR
		{
		"components": [{
		"version": "*...*"
		}]
		}

ファイル名
 	
コンポーネント
	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "software_File",
	"spdxId": "*example_URI_02*",
	"name": "*...*"
	},
	{
		{
		"metadata" {
		"component": {
		"プロパティ": [{
		"name": "bsi:component:filename",
		"value": "*...*"
		}]
		}
		}
		}

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

	"type": "Relationship",
	"from": "*example_URI_01"*,
	"relationshipType": "hasDistributionArtifact",
	"to": ["*example_URI_02"*],
	"completeness": "complete"
	}
		XOR
		{
		"components": [{
		"プロパティ": [{
		"name": "bsi:component:filename",
		"value": "*...*"
		}]
		}]
		}

他のコンポーネントへの依存関係	{
	"type": "Relationship",
	"from": "*example_URI_01"*,
	"relationshipType": "contains",
	XOR
	"dependsOn",
	"to": [...]
	"completeness": "complete"
	XOR
	"incomplete"
	XOR
	"noAssertion" }
		{
		"dependencies": [{
		"ref": *...*,
		"dependsOn": [*...*]
		}]
		},
		{
		"metadata" {
		"component": {
		"components": [{*...*}]
		}
		}
		},
		{
		"コンポーネント": {
		"components": [{*...*}]
		}
		},
		{
		"compositions": [{
		"ref": *...*,
		"aggregate": "complete",
		XOR

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*
		"incomplete", *XOR* "不明", "アセンブリ": [...], *OR* "依存関係": [...] }] } 注記（CycloneDX v1.6 仕様を参照）[23] ：依存関係グラフ内で独自の 依存関係を持つコンポーネントは、依存関係グラフ内で空の要素として宣言されなければならない。 注記：「構成要素」は「依存関係」と「アセンブリ」の完全性を記述するためにのみ使用される。

配布ライセンス	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "Relationship",
	"from": "*example_URI_01*",
	"relationshipType": "hasConcludedLicense",
	"to": ["*example_URI_02*"],
	"completeness": "complete"
	},
	{
	"type": "simpleLicensing_LicenseExpression",
	"spdxId": "*example_URI_02*",
	"licenseExpression": "*...*"
		{
		"metadata" {
		"component": {
		"licenses": [{
		"expression": "*...*",
		"acknowledgement": "concluded"
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"licenses": [{
		"expression": "*...*",
		"acknowledgement": "concluded"
		}]

[23] https://cyclonedx.org/docs/1.6/json/#dependencies

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*
	}	}] }

展開可能なコンポーネントのハッシュ値	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "software_File",
	"spdxId": "*example_URI_02"*,
	"verifiedUsing": [{
	"type": "Hash",
	"algorithm": "sha512",
	"hashValue": "*...*"
	}]
	},
	{
	"type": "Relationship",
	"from": "*example_URI_01"*,
	"relationshipType": "hasDistributionArtifact",
	"to": ["*example_URI_02"*],
	"completeness": "complete"
	}
		{
		"metadata" {
		"component": {
		"外部参照": [{
		"type": "distribution",
		"hashes": [{
		"alg": "SHA-512",
		"content": "*...*"
		}]
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"externalReferences": [{
		"type": "distribution",
		"hashes": [{
		"alg": "SHA-512",
		"content": "*...*"
		}]
		}]
		}]
		}

実行可能プロパティ	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
		{
		"metadata" {
		"component": {
		"properties": [{

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

	{
	"type": "software_File",
	"spdxId": "*example_URI_02"*,
	"software_additionalPurpose": ["..."],
	"comment": "software_additionalPurpose フィールドは、BSI TR-03183-2」
	},
	{
	"type": "関係",
	"from": "*example_URI_01"*,
	"relationshipType": "hasDistributionArtifact",
	"to": ["*example_URI_02"*],
	"completeness": "complete"
	}
	
	注：コンポーネントが実行可能である場合、コンポーネントが実行可能である場合のみ「software_additionalPurpose」リストに追加してください。それ以外の場合は追加しないでください。
		"name": "bsi:component:executable",
		"value": "..."
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"プロパティ": [{
		"name": "bsi:component:executable",
		"value": "..."
		}]
		}]
		}

アーカイブプロパティ	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "software_File",
	"spdxId": "*example_URI_02"*,
	"software_additionalPurpose": ["..."],
	"comment": "software_additionalPurpose
		{
		"metadata" {
		"component": {
		"properties": [{
		"name": "bsi:component:archive",
		"value": "..."
		}]
		}
		}
		}

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

	フィールドは、BSI
	TR-03183-2」
	},
	{
	"type": "Relationship",
	"from": "*example_URI_01"*,
	"relationshipType": "hasDistributionArtifact",
	"to": ["*example_URI_02"*],
	"completeness": "complete"
	}
	
	注記: コンポーネントがアーカイブである場合、
	コンポーネントがアーカイブである場合のみ「software_additionalPurpose」リストに追加してください。それ以外の場合は追加しないでください。
		XOR
		{
		"components": [{
		"プロパティ": [{
		"name": "bsi:component:archive",
		"value": "..."
		}]
		}]
		}

構造化プロパティ	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "software_File",
	"spdxId": "*example_URI_02"*,
	"software_additionalPurpose": ["..."],
	"comment": "software_additionalPurpose
	フィールドは、BSI
	TR-03183-2」
	},
	{
	"type": "Relationship",
	"from": "*example_URI_01"*,
		{
		"metadata" {
		"component": {
		"properties": [{
		"name": "bsi:component:structured",
		"value": "..."
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"プロパティ": [{
		"name": "bsi:component:structured",
		"value": "..."

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*
	"relationshipType": "hasDistributionArtifact", "to": ["*example_URI_02"*], "completeness": "complete" } 注：コンポーネントが構造化ファイルの場合は、 コンポーネントが構造化ファイルの場合に限り、「software_additionalPurpose」リストに「container」を追加してください。それ以外の場合は追加しないでください。 注記：コンポーネントが構造化ファイルでない場合、「firmware」を コンポーネントが構造化ファイルでない場合、「software_additionalPurpose」リストに「firmware」を追加してください。それ以外の場合は追加しないでください。	}] }] }

表 *10: SBOM* 自体に対する追加データフィールドのマッピング

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

SBOM-URI	{
	"type": "SpdxDocument",
	"rootElement": "*...*"
	},
	{
	"type": "software_Sbom",
	"spdxId": "*...*",
	"rootElement": "*example_URI_01*"
	},
	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
		{
		"serialNumber": "*...*"
		}

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*
	} 注記: 「spdxId」が「example_URI_01」である「software_Package」は、SBOM の主要要素を指す。	

表 *11*：各コンポーネントの追加データフィールドのマッピング

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

ソースコード
URI
	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "software_SoftwareArtifact",
	"spdxId": "*example_URI_02"*,
	"software_primaryPurpose": "source",
	"externalRef": [{
	"type": "ExternalRef",
	"externalRefType": "SourceArtifact",
	"locator": "*...*"
	}]
	},
	{
	"type": "Relationship",
	"from": "*example_URI_02"*,
	"relationshipType": "generates",
	"to": ["*example_URI_01"*],
	"completeness": "complete"
	}
		{
		"metadata" {
		"component": {
		"外部参照": [{
		"type": "source-distribution",
		"url": "*...*"
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"externalReferences": [{
		"type": "source-distribution",
		"url": "*...*"
		}]
		}]
		}

デプロイ可能なコンポーネントの	{ "type": "software_Package",	{ "metadata" {

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

コンポーネントのデプロイ可能	"spdxId": "*example_URI_01*"
	},
	{
	"type": "software_File",
	"spdxId": "*example_URI_02"*,
	"binaryArtifact": "..."
	},
	{
	"type": "Relationship",
	"from": "*example_URI_01"*,
	"relationshipType": "hasDistributionArtifact",
	"to": ["*example_URI_02"*],
	"completeness": "complete"
	}
		"component": {
		"外部参照": [{
		"type": "distribution",
		"url": "*...*"
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"externalReferences": [{
		"type": "distribution",
		"url": "*...*"
		}]
		}]
		}

その他の一意の識別子	{
	"type": "software_Package"
	"externalIdentifiers": [{
	"externalIdentifierType": "cpe22",
	または
	"cpe23",
	OR
	"swid",
	または
	"packageURL", "identifier": "..."
	}]
	}
		{
		"metadata" {
		"component": {
		"cpe": "*cpe:/...*"
		OR
		"swid": {
		"tagId": "*...*"
		}
		OR
		"purl": "*...*"
		}
		}
		}
		XOR
		{

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

		"components": [{
		"cpe": "*cpe:/...*"
		OR
		"swid": {
		"tagId": "*...*"
		}
		または
		"purl": "*...*"
		}]
		}

オリジナルライセンス	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "Relationship",
	"from": "*example_URI_01*",
	"relationshipType": "hasDeclaredLicense",
	"to": ["*example_URI_02*"],
	"completeness": "complete"
	},
	{
	"type": "simpleLicensing_LicenseExpression",
	"spdxId": "*example_URI_02*",
	"licenseExpression": "*...*"
	}
		{
		"metadata" {
		"component": {
		"licenses": [{
		"expression": "*...*",
		"acknowledgement": "declared"
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"licenses": [{
		"expression": "*...*",
		"acknowledgement": "declared"
		}]
		}]
		}

表 *12:* 各コンポーネントのオプションデータフィールドのマッピング

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

有効なライセンス	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "関係",
	"from": "*example_URI_01*",
	"relationshipType": "other",
	"to": ["*example_URI_02*"],
	"comment": "有効なライセンスを保持",
	"completeness": "complete"
	},
	{
	"type": "simpleLicensing_LicenseExpression",
	"spdxId": "*example_URI_02*",
	"licenseExpression": "*...*"
	}
		{
		"metadata" {
		"component": { "properties": [{ "name": "bsi:component:effectiveLicense",
		"value": "*...*"
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"プロパティ": [{
		"name": "bsi:component:effectiveLicense",
		"value": "*...*"
		}]
		}]
		}

コンポーネントのソースコードのハッシュ値
コンポーネントのソースコード
	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "software_SoftwareArtifact",
	"spdxId": "*example_URI_02*",
	"software_primaryPurpose": "source",
	"verifiedUsing": [{
	"type": "Hash",
	"algorithm": "sha512",
	"hashValue": "*...*"
		{
		"metadata" {
		"component": {
		"外部参照": [{
		"type": "source-distribution",
		"hashes": [{
		"alg": "SHA-512",
		"content": "*...*"
		}]
		}]
		}
		}

データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

	}]
	},
	{
	"type": "Relationship",
	"from": "*example_URI_02"*,
	"relationshipType": "generates",
	"to": ["*example_URI_01"*],
	"completeness": "complete"
	}
		}
		XOR
		{
		"components": [{
		"externalReferences": [{
		"type": "source-distribution",
		"hashes": [{
		"alg": "SHA-512",
		"content": "*...*"
		}]
		}]
		}]
		}

security.txt の URL	{
	"type": "software_Package",
	"externalRef": [{
	"type": "ExternalRef",
	"externalRefType": "securityOther",
	"locator": "*...*"
	}]
	}
		{
		"metadata" {
		"component": {
		"externalReferences": [{
		"type": "rfc-9116",
		"url": "*...*"
		}]
		}
		}
		}
		XOR
		{
		"components": [{
		"externalReferences": [{
		"type": "rfc-9116",
		"url": "*...*"
		}]
		}]
		}

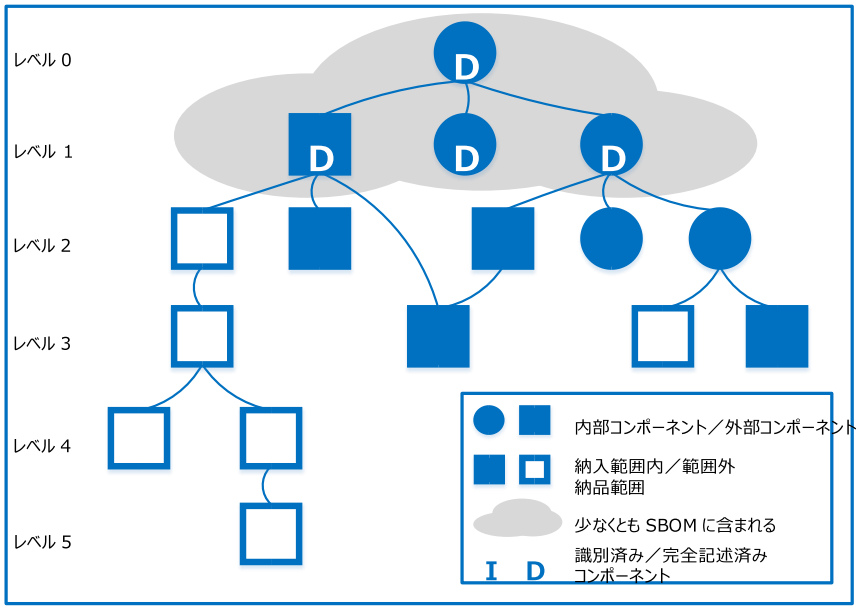
データフィールド	*SPDX v3.0.1 (JSON)*	*CycloneDX v1.6 (JSON)*

別の BOM への参照	{
	"type": "software_Package",
	"spdxId": "*example_URI_01*"
	},
	{
	"type": "software_Sbom",
	"spdxId": "*example_URI_02"*,
	"externalRef": [{
	"type": "ExternalRef",
	"externalRefType": "buildMeta",
	"locator": "*...*"
	}]
	},
	{
	"type": "Relationship",
	"from": "*example_URI_02"*,
	"relationshipType": "describes",
	"to": ["*example_URI_01"*],
	"completeness": "complete"
	}
		{
		"components": [{
		"外部参照": [{
		"type": "bom",
		"url": "*...*"
		}]
		}]
		}

8.3 SBOM の詳細レベル

構造上、SBOM は異なる詳細レベルで作成可能であり、例えば以下の分類に従う。

8.3.1 トップレベル SBOM



主要コンポーネントの完全な記述に加え、SBOM には主要コンポーネントが直接依存する全てのコンポーネントの完全な記述が含まれる。

図　*1:トップレベル SBOM*

8.3.2 n 階層 SBOM

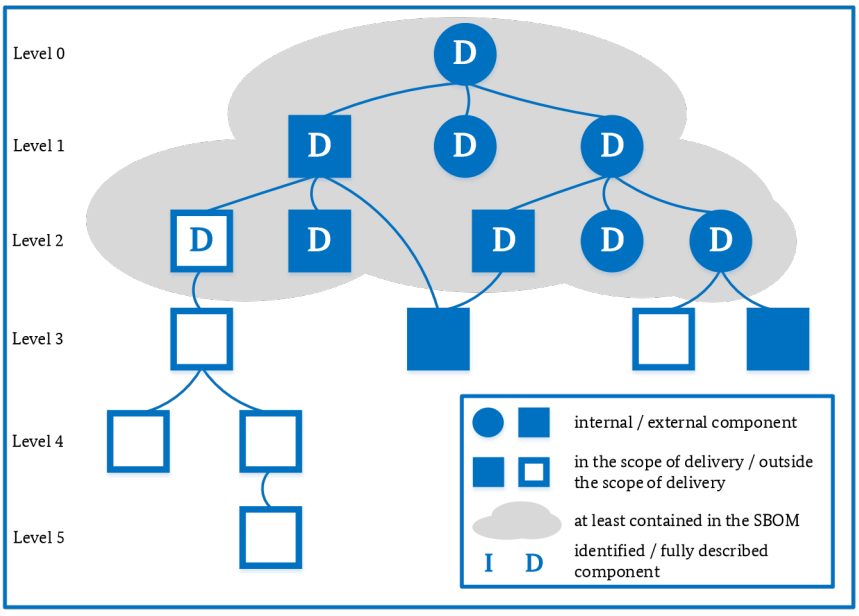
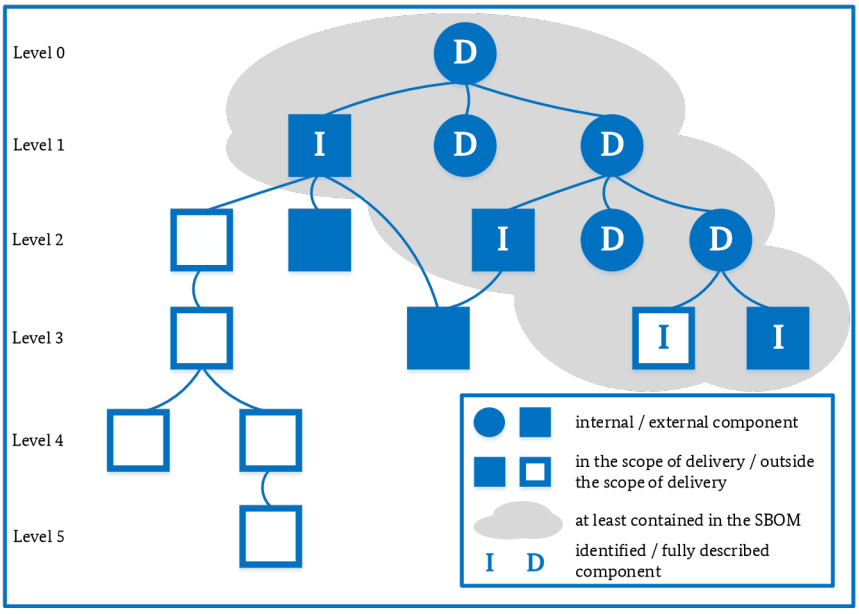


図 *2: n=2 における n レベル SBOM*

主要コンポーネントの完全な記述に加え、SBOM には主要コンポーネントが n レベルを通じて直接的または推移的に依存する全てのコンポーネントの完全な記述が含まれます。これは、推移的依存関係の再帰的解決が深さ n ステップに制限されることを意味する。主要コンポーネントからのパスが n レベル未満の場合、このパス上の全てのコンポーネントを解決し、結果として完全に記述する必要がある。

最上位レベルの SBOM は 1 レベル SBOM である。

8.3.3 推移的 SBOM

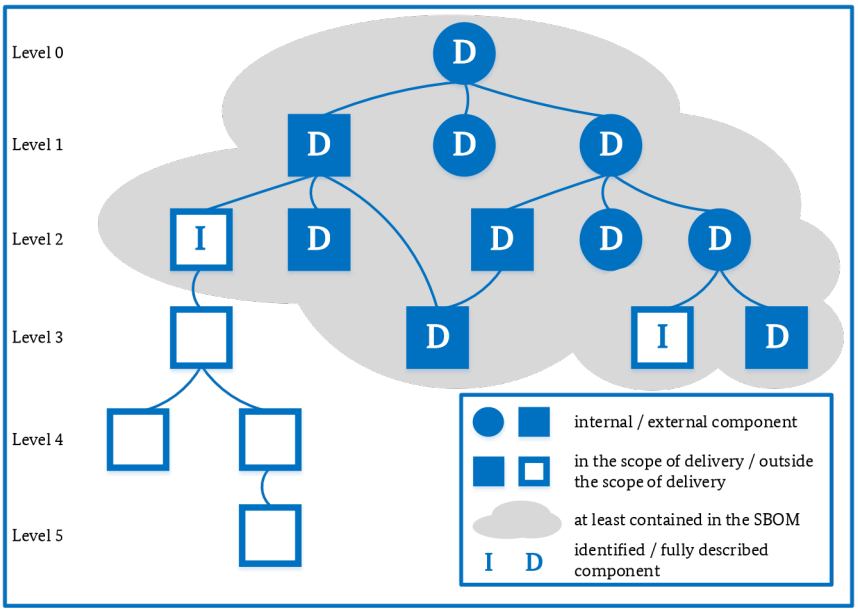


SBOM には、主要コンポーネントの完全な記述に加え、主要コンポーネントが直接的または推移的に依存するすべてのコンポーネントに関する情報が含まれます。コンポーネントとその依存関係の完全な記述および再帰的解決は、各パスにおいて少なくとも最初の外部コンポーネント（すなわちサードパーティ製コンポーネント）まで行われます。この依存関係を明確に関連付けるため、最初の外部コンポーネントは少なくとも SBOM 内で識別されなければならない。したがって、その依存関係はさらに解決する必要はない。

図 *3:* 推移的 *SBOM*

最初の外部コンポーネントを含む n 階層 SBOM と比較して、推移的 SBOM はこの外部コンポーネントに関する情報が少なくなる可能性がある。推移的 SBOM は外部コンポーネントを完全に記述する必要がないためである。

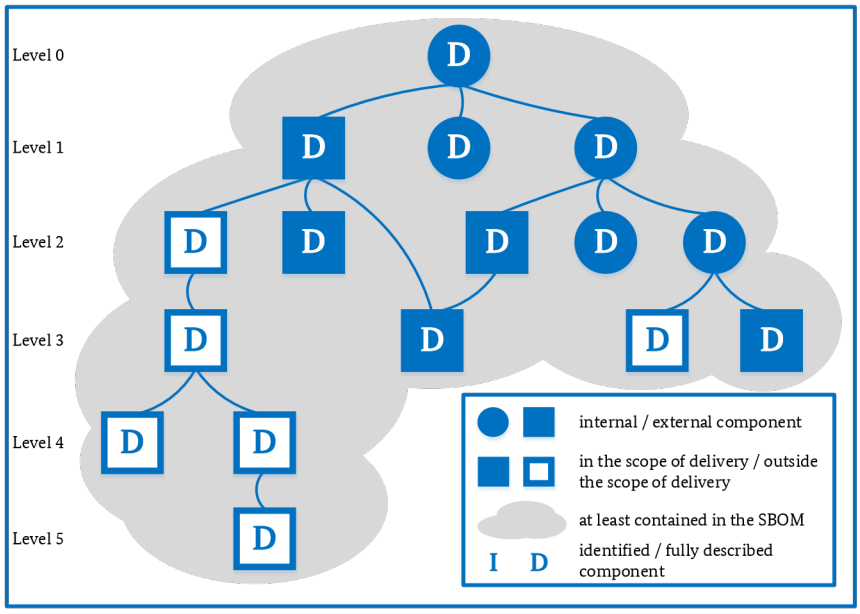
8.3.4 納入品目 SBOM



SBOM には、主要コンポーネントの完全な記述に加え、納品範囲に属し、主要コンポーネントが直接的または推移的に依存するすべてのコンポーネントの完全な記述が含まれる。コンポーネントとその依存関係の完全な記述および再帰的解決は、各パスにおいて、少なくとも納品範囲外の最初のコンポーネントまで行われる。納品範囲外の最初のコンポーネントは、この依存関係を明確に関連付けるために、少なくとも SBOM 内で識別されなければならない。したがって、その依存関係はさらに解決する必要はない。

図 *4*：納品物 *SBOM*

8.3.5 完全な SBOM



主要コンポーネントの完全な記述に加え、SBOM には主要コンポーネントが直接的または間接的に依存するすべてのコンポーネントの完全な記述が含まれます。コンポーネントとその依存関係の完全な記述および再帰的解決が完全に実施される。

Figure 5: Complete SBOM

8.4 SBOM の分類

コンポーネントの開発、提供、インストール、実行プロセスの一環として SBOM が作成される方法や時期によって、その特定の状況で利用可能なデータは異なる。

その結果、SBOM に集約可能な情報も異なる。一般的な区別として、以下の SBOM クラスが挙げられる。

8.4.1 設計 SBOM

新規ソフトウェア成果物に含まれる計画上のコンポーネント群に基づいて作成される SBOM である。コンポーネントは現時点で存在していなくても構いない。

8.4.2 ソース SBOM

SBOM は、開発環境、ソースファイル、およびそれらが使用する依存関係から作成される。

8.4.3 ビルド SBOM

SBOM は、ビルドプロセスの一部として、ソースファイル、依存関係情報、既に作成されたコンポーネント、揮発性のビルドプロセスデータ、その他の SBOM などを基に作成される。

注記：

- 実行可能バイナリコンポーネント（事前コンパイル済みコードなど）を捕捉可能にするため、ビルド SBOM の作成ではコンパイラ実行ではなく、翻訳済みコードのリンカ実行に焦点を当てます。
- ハッシュ値がコンポーネントを明確に識別できるようにするには、再現可能なビルドを採用する必要がある。
- インタプリタコードの場合、ソースコードのみが存在するため、各実行ファイルをコンポーネントとしてリストする必要がある。インタプリタは、合理的に可能な限り依存関係として明示する必要がある。

8.4.4 解析済み SBOM

SBOM は、ビルドプロセス後に実行ファイル、パッケージ、コンテナ、仮想マシンイメージなどのアーティファクトを分析して作成される。このタイプは「サードパーティ SBOM」とも呼ばれます。

8.4.5 展開済み SBOM

この SBOM は、システム上のソフトウェアのインベントリを提供する。これは、構成オプションを考慮し、（シミュレーションされた可能性のある）展開環境における実行動作の検証を踏まえた、他の SBOM の統合版となる場合がある。

8.4.6 ランタイム SBOM

この SBOM は、ソフトウェアを実行するシステムを用いて作成され、実行中のコンポーネント（実行中のコンポーネント）だけでなく、その外部呼び出しや動的にロードされたコンポーネントを、実行時のみ（メモリ内）に捕捉する。このタイプは「動的 SBOM」とも呼ばれる場合がある。

8.5 詳細情報

8.5.1 NTIA からの情報

「米国商務省」傘下の「国家電気通信情報局（NTIA）」は、https://ntia.gov/sbom で SBOM に関する詳細な情報を提供している。

8.5.2 CISA からの情報

米国国土安全保障省傘下のサイバーセキュリティ・インフラセキュリティ庁（CISA）も、SBOM に関する追加情報を提供している。詳細は https://cisa.gov/sbom をご覧ください。

8.5.3 CycloneDX からの情報

CycloneDX も初心者向けのガイドとリソースを https://cyclonedx.org/guides/で提供している。

8.5.4 SPDX からの情報

SPDX は、SPDX ドキュメントおよび SBOM の作成および／または消費に関する詳細情報を https://spdx.dev/learn/overview/で提供している。