Complying with CRA SBoM Requirements using the Yocto Project

Joshua Watt

Open Source Summit

May 18, 2026

About Me

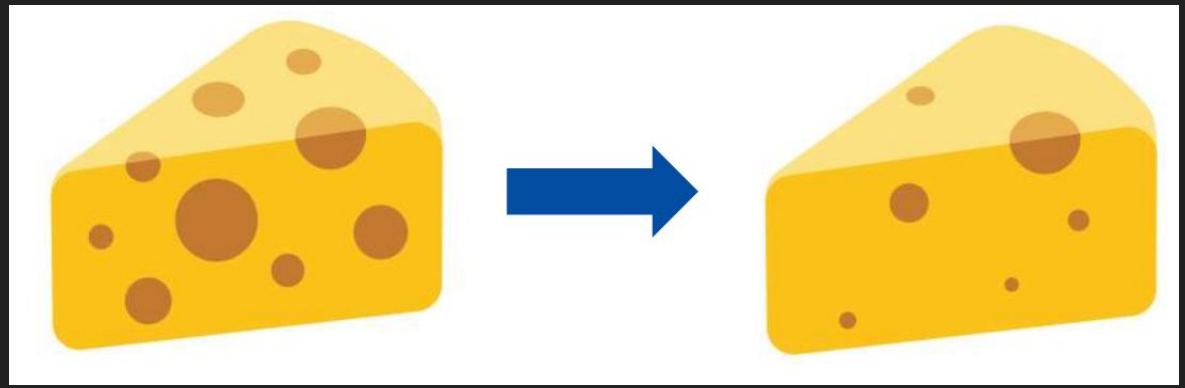
- Worked at Garmin since 2009
- Using OpenEmbedded & Yocto Project since 2016
- Member of the OpenEmbedded Technical Steering Committee (TSC)
- Member of Yocto TSC
- Member of the SPDX Technical Team
- Joshua.Watt@garmin.com
- JPEWhacker@gmail.com
- IRC (OFTC or libera): JPEW
- X/Twitter: @JPEW_dev
- LinkedIn: joshua-watt-dev

Disclaimers

- Opinions are my own
- I Am Not a Lawyer!
  - All interpretations of documents are my own educated guess; please consult with a legal expert

What is the CRA?

- Cyber Resilience Act
- Ratified in 2024
- Goes into effect in December 2027 (with some parts sooner)
- Intended to improve the security posture of Software used in the EU
- Requires SBoMs (and many other things)



https://www.cisa.gov/sites/default/files/2023-09/EU%20Commission%20SBOM%20Work_508c.pdf

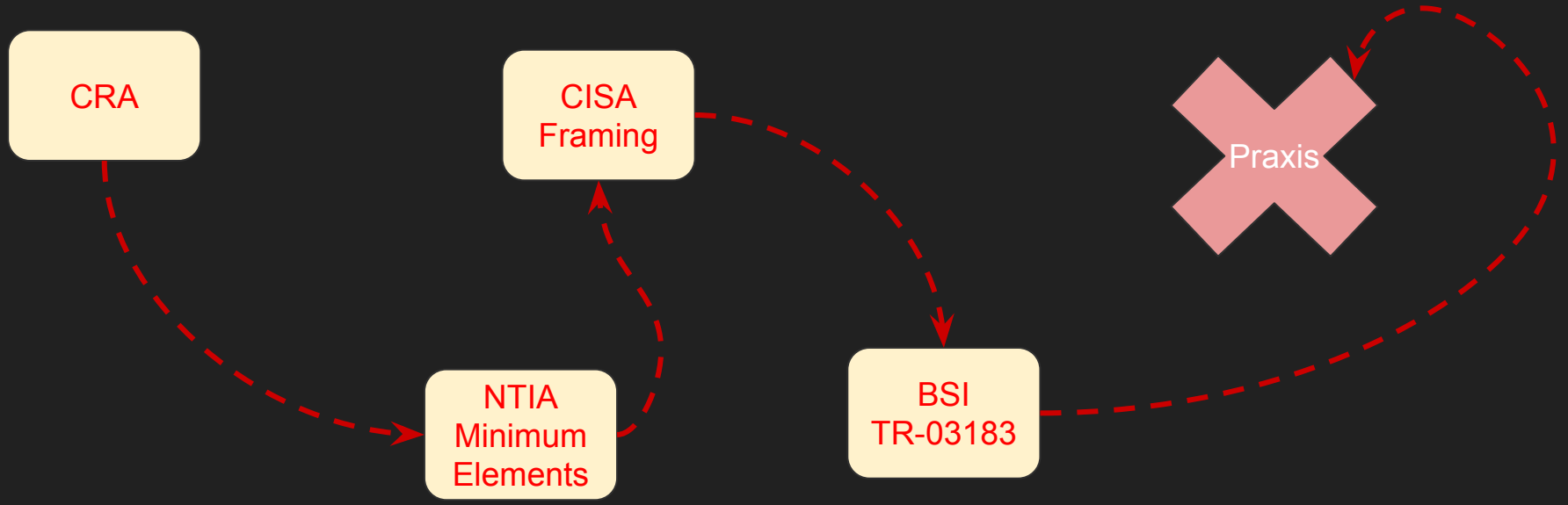
CRA SBoM Requirements

- The primary CRA text has the only *actual* currently known requirements
  - Summary
    - Use a standard SBoM format
    - Cover at least one level of dependencies
    - SBoM is expected be used to scan for known vulnerabilities
- Maybe there will be more this Summer (2026)?

So… we're done then?

rick, CC BY 2.0, via Wikimedia Commons

Our Journey



The CRA isn't likely to "adopt" any of these, but it can be useful to get a feel for what might happen.

Yocto Support for SBoM Requirements

Only covering what upstream Yocto can provide automatically; SPDX has support for the rest.

Legend

- ✅ - Provided automatically
- ⚠️ - Provided automatically, with configuration
- 🚧 - Possible in the future with code changes
- ⛔ - Not provided automatically (requires post-processing)

NTIA Minimum Elements

NTIA Minimum Elements

- Published by the U.S. National Telecommunications and Information administration in 2021
- Per Executive Order 14028
- New version published for comments on August 22, 2025
  - Not covered in this presentation since it not official yet
- Describes the minimum elements required when supplying a SBoM to the U.S. Government
- All text in orange are direct quotes from the NITA Minimum Elements Document
- https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom

NTIA Minimum Elements

Field	Description	Yocto Support
Supplier Name	The name of an entity that creates, defines, and identifies components	⚠️
Component Name	Designation assigned to a unit of software defined by the original supplier.	✅
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version.	✅
CPE identifier*	Common Platform Enumeration (CPE)	✅
purl identifier*	Package URL (purl)	✅/⚠️

* Combined in the Other Unique Identifiers field

NTIA Minimum Elements

Field	Description	Yocto Support
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y.	✅
Author of SBOM Data	The name of the entity that creates the SBOM data for this component.	⚠️
Timestamp	Record of the date and time of the SBOM data assembly.	⚠️

NTIA Recommended Elements

Field	Description	Yocto Support
Hash of the Component	Hashes also offer confidence that a specific component was used.	✅
Lifecycle Phase	SBOM may have some differences depending on when and where the data was created	✅
Other Component Relationships	Other types of dependency relationships can be captured, and have been implemented in some SBOM standards	✅
License Information	SBOMs can convey data about the licenses for each component	✅

NTIA SBoM Detail Level

An SBOM should contain all primary (top level) components, with all their transitive dependencies listed. At a minimum, all top-level dependencies must be listed with enough detail to seek out the transitive dependencies recursively.

Yocto SBoM describe the top level image, all installed packages and their runtime dependencies, which should be sufficient ✅.

More on this later

CISA Framing Software Component Transparency

CISA Framing Software Component Transparency

- Third edition published in 2024 by the U.S. Cybersecurity and Infrastructure Security Agency Industry Working Group
- Provides minimum expected SBoM information, recommended best practices, and aspirational goals
- Effectively an "extension" of the NTIA document; only additions will be covered here
- Mostly likely superseded by new NTIA minimum elements
- **General Consensus is that CRA requirements will be closest to this**
- All text in orange are direct quotes from the Framing Document
- https://www.cisa.gov/resources-tools/resources/framing-software-component-transparency-2024

CISA Recommendations for SBoM Metadata

Field	Description	Yocto Support
Type Attribute	The Type Attribute provides context for how and why the SBOM was created	✅
Primary Component	The Primary Component, or root of Dependencies, is the subject of the SBOM or the foundational Component being described in the SBOM	✅

CISA Recommendations for Components

Field	Description	Yocto Support
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.	🚧(?)
Relationship Completeness	Cover(s) the range of an author's knowledge about another Supplier's Components	🚧
Copyright Notice	The Component Copyright Holder identifies the entity that holds exclusive and legal rights to the listed Component in the SBOM	🚧(?)

BSI TR-03183

BSI TR-03183

- Published by German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik)
- Cyber Resilience Requirements for Manufacturers and Products
- 3 Parts:
  - General CRA Requirements
  - SBoM Requirements
  - Vulnerability Reports
- https://bsi.bund.de/dok/TR-03183
- Version 2.1.0 of Part 2 (SBoMs) released in August, 2025
- All text in orange are direct quotes from TR-03183

BSI TR-03183 SBoM Requirements

- SBoM requirements are based on the latest version of BSI TR-03183 *at the time of SBoM delivery* (with 6 months grace)
  - You'll need to keep an eye on changes to TR-03183!
  - This presentation is based on v2.1.0
- Format Requirements
  - JSON or XML
  - CycloneDX >= 1.6
  - SPDX >= 3.0.1

BSI TR-03183 SBoM Requirements

Field	Description	Yocto Support
Creator of the SBOM	Email address of the entity that created the SBOM. If no email address is available this MUST be a “Uniform Resource Locator (URL)”, e.g. the creator’s home page or the project’s web page.	⚠️
Timestamp	Date and time of the SBOM data compilation according to the specification of the formats	⚠️

BSI TR-03183 Component Requirements

Field	Description	Yocto Support
Component creator	Email address of the entity that created and, if applicable, maintains the respective component. If no email address is available this MUST be a “Uniform Resource Locator (URL)”, e.g. the creator’s home page or the project’s web page.	✅/🚧 (`${HOMEPAGE}` from recipe)
Component name	Name assigned to the component by its creator. If no name is assigned this MUST be the actual filename.	✅ (`${PN}`, filename, etc.)
Component version	Identifier used by the creator to specify changes in the component to a previously created version. Identifiers according to Semantic Versioning or alternatively Calendar Versioning SHOULD be used if one determines the versioning scheme. Existing identifiers MUST NOT be changed for this purpose. If no version is assigned this MUST be the creation date of the file expressed as full-date according to RFC 3339 section 5.6. To determine the creation time the file metadata MUST be consulted.	✅/⛔/🚧 (`${PV}`, but may not match the upstream version)

BSI TR-03183 Component Requirements

Field	Description	Yocto Support
Filename of Component	The actual filename of the component (i.e. not its path)	✅
Dependencies on other components	Enumeration of all components on which this component is directly dependent	✅
Distribution Licences	Distribution licence(s) of the component under which it can be used by a licensee (Declared License)	✅
Hash value of the deployable component	Cryptographically secure checksum (hash value) of the deployed/deployable component (i.e. as a file on a mass storage device) as SHA-512	🚧

BSI TR-03183 Component Requirements

Field	Description	Yocto Support
Executable property	Describes whether the component is executable	??/ 
Archive property	Describes whether the component is an archive	??/ 
Structured property	Describes whether the component is a structured file, i.e. metadata of the contents is still present	??/ 

BSI TR-03183 Additional Information

- Required if the information exists

BSI TR-03183 Additional SBoM Requirements

Field	Description	Yocto Support
SBOM-URI	“Uniform Resource Identifier (URI)” of this SBOM	⛔

BSI TR-03183 Additional Component Requirements

Field	Description	Yocto Support
Source code URI	“Uniform Resource Identifier (URI)” of the source code of the component, e.g. the URL of the utilised source code version in its repository, or if a version cannot be specified the utilised source code repository itself.	✅
URI of the deployable form of the component	“Uniform Resource Identifier (URI)”, which points directly to the deployable (e.g. downloadable) form of the component.	⛔
CPE identifier*	Common Platform Enumeration (CPE)	✅
purl identifier*	Package URL (purl)	✅/⚠️

* Combined in the Other unique identifiers field

BSI TR-03183 Additional Component Requirements

Field	Description	Yocto Support
Original licenses	The licence(s) that have been assigned by the creator of the component (Declared License)	✅

BSI TR-03183 Optional Information

- MAY be include if information exists
- There is no optional information for SBoM (only components)

BSI TR-03183 Optional Component Information

Field	Description	Yocto Support
Effective license	The licence(s) under which the component is used by the licensee that is the creator of the current SBOM (Concluded License)	⚠️
Hash value of the source code of the component	Cryptographically secure checksum (hash value) of the component source code. A specific algorithm how to create a hash value of multiple source files or a source code tree, and which hash algorithm is utilised for that has not yet been determined.	🚧

Digitally Signing SBoMs

- Ideally, SBOMs should be digitally signed so recipients can verify their authenticity.
- This is outside the scope of Yocto ⛔

BSI TR-03183 SBoM Classifications

Design SBoM

The SBOM is created based on the planned set of included components of a new software artefact. The components do not have to exist yet.

Source SBoM

The SBoM is created from the development environment, the source files and the dependencies it uses.

Build SBoM

The SBoM is created as part of the build process based on e.g. source files, dependency information, already created components, volatile build process data and other SBOMs.

Notes

- In order to enable capturing executable, binary components that already exist (i.e. precompiled code), creating a Build SBOM focuses on the linker run for translated code, not the compiler run.
- In order to let hash values unambiguously identify components, reproducible builds have to be employed.
- In the case of interpreted code, only the source code exists; each executable file has to be listed as a component. The interpreter has to be specified as a dependency, as far as reasonably possible.

Analysed SBoM

The SBOM is created after the build process by analysing artefacts such as executables, packages, containers and virtual machine images. This type is also referred to as “3rd party SBOM”.

Deployed SBOM

The SBOM provides an inventory of the software on a system. This can be a compilation of other SBOMs, taking into account configuration options and examination of execution behaviour in a (possibly simulated) deployment environment.

Runtime SBoM

The SBOM is created using the system executing the software to capture running (i.e. executing) components as well as their external calls and dynamically loaded components at runtime only (i.e. in memory). This type may also be referred to as “Dynamic SBOM”.

Classification Conformance

SBOM MUST contain the same information as available during the build process or equivalent information where the build process does not exist.

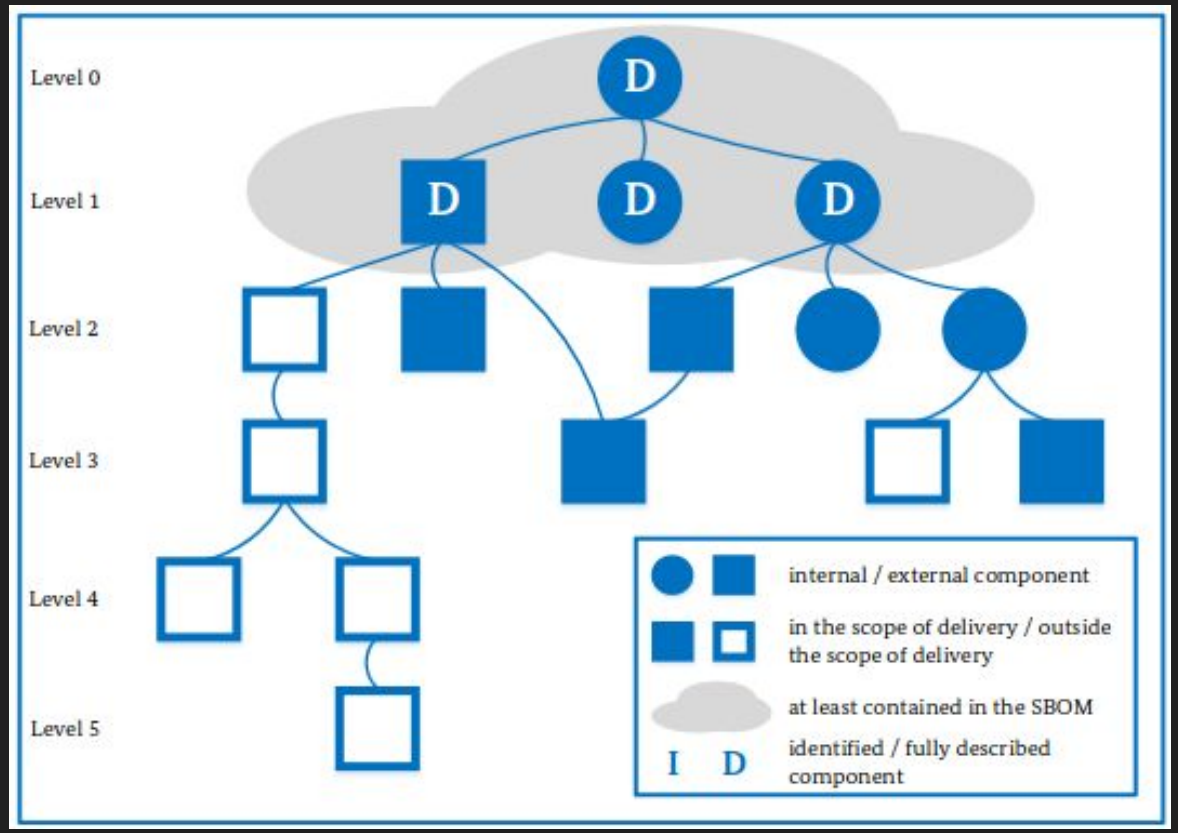
Yocto produces a Build SBoM, so this should conform ✅

BSI TR-03183 SBoM Detail Levels

Top-level

In addition to the full description of the primary component, the SBOM contains the full description of all components, which the primary component directly depends on.

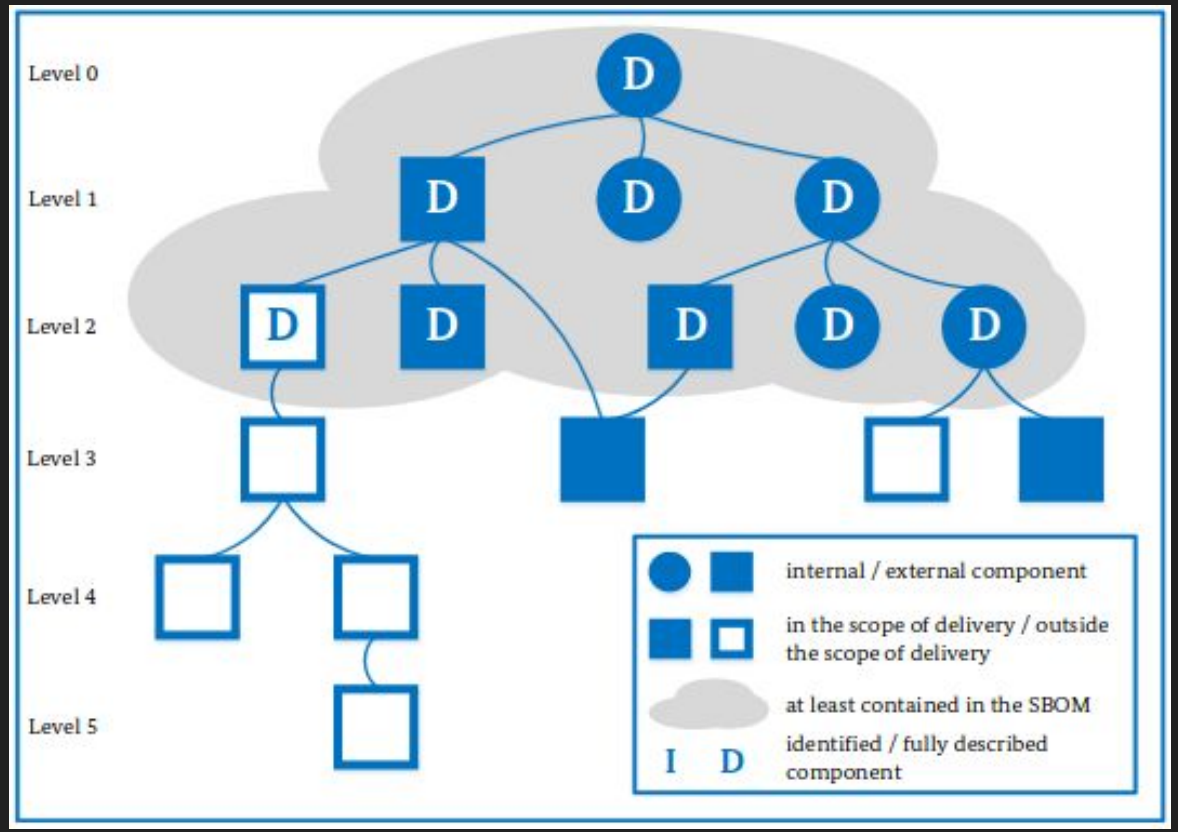
This is what is required by the CRA (but may change)



n-level

In addition to the full description of the primary component, the SBOM contains the full description of all components, which are directly or transitively depended upon via n levels by the primary component. This means that the recursive resolution of the transitive dependencies is limited to *n* steps in depth. If the path from the primary component is shorter than n levels, all components on this path have to be resolved and, consequently, fully described.

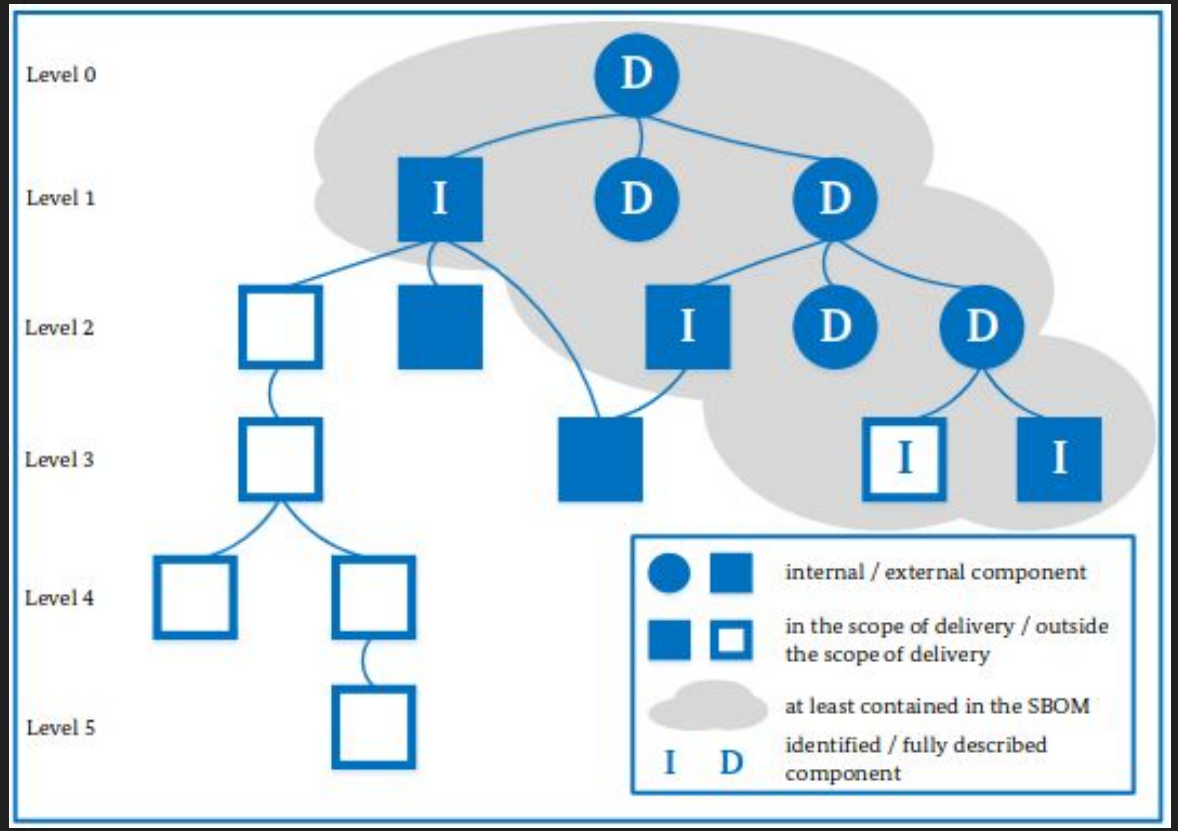
A top-level SBOM is a 1-level SBOM.



Transitive

Recursively describes components up to and including the first external component (i.e. third-party component).

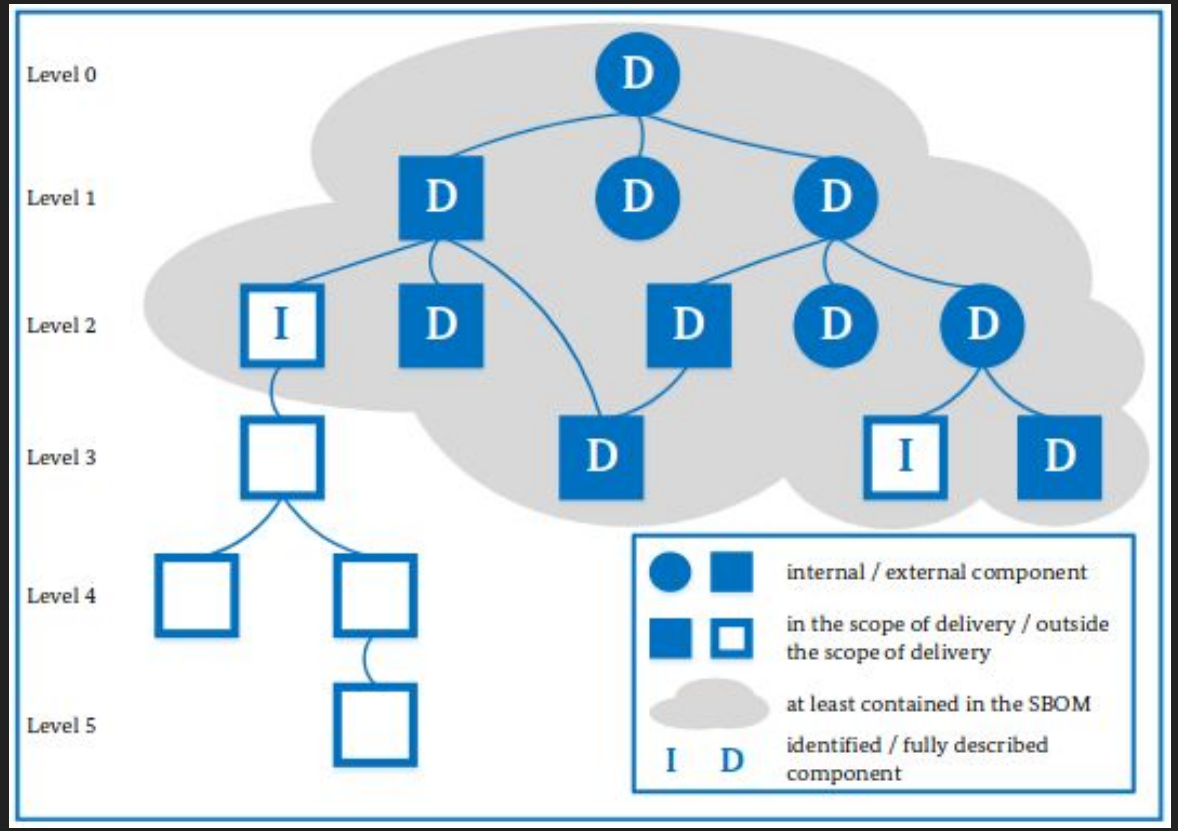
This is what Yocto generates, and what NTIA requires



Delivery Item

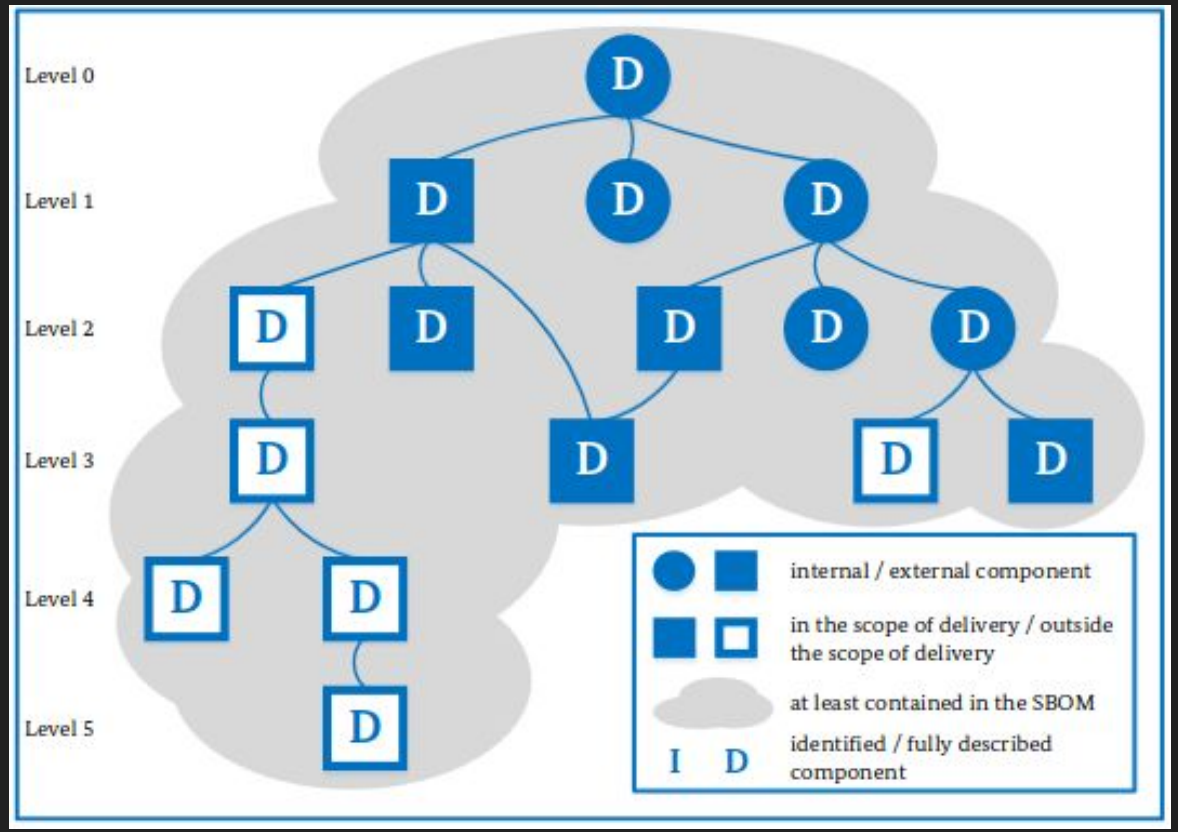
Recursively describes components up to and including the first component outside of the scope of the delivery.

This is what is required by TR-03183 ⛔



Complete

In addition to the full description of the primary component, the SBOM contains the full description of all components, which are directly or transitively depended upon by the primary component. The full description and recursive resolution of the components and their dependencies is carried out completely.



Summary

Result Summary

Standard	✅	⚠️	🚧	⛔
NTIA Minimum Elements	10	4	0	0
CISA Framing Document	12	4	3	0
BSI TR-03183	11	4	6	5

My Analysis & Thoughts

- We got most of the required basics covered
- Hopefully when the actual requirements are published, we will be able to quickly pivot to implement anything missing
- It looks like we should be able to get most of a valid SBoM directly from the build, with little to no manual annotation afterwards (unless CRA borrows too much from TR-03183)

SBoM Praxis
(a.k.a. "What do I *actually* need to do")

What Yocto versions?

- Yocto 5.1 and later
- SPDX 3.0
- Possible backport to older Yocto version in the future

CRA SBoM Praxis - Creator of the SBOM

You **MUST** define you (or your company) as the creator of the SBoM in question using the `SPDX_AUTHORS` variable (e.g. in local.conf or similar). There are a few ways to do this:

Publish an SPDX document with your information and reference it (preferred):

```
# Import an SPDX ID from an external URL
SPDX_IMPORTS += "mycompany"
SPDX_IMPORTS_mycompany_spdxid = "https://your-company-spdx-id"
SPDX_IMPORTS_mycompany_uri = "https://spdxdocument-url.spdx.json"
SPDX_IMPORTS_mycompany_hash_sha256 = "e66225…"

# Reference the SPDX ID as the author
SPDX_AUTHORS += "myauthor"
SPDX_AUTHORS_myauthor_import = "mycompany"
```

CRA SBoM Praxis - Creator of the SBOM

Create a new SPDX object for your company in each document:

```
SPDX_AUTHORS += "myauthor"
SPDX_AUTHORS_myauthor_name = "my company"
SPDX_AUTHORS_myauthor_type = "organization"
SPDX_AUTHORS_myauthor_id_email = "foo@mycompany.com"
SPDX_AUTHORS_myauthor_id_urlScheme = "http://mycompany.com"
```

Note: This will result in many author components in the final document, since each SPDX fragment will create a new one

CRA SBoM Praxis - Component Creator/Supplier

You (or your company) are the "Creator" or "Supplier" of a given component in an SBoM (since you are building it and responsible for the binary output). As such, you **MUST** define you (or your company) as the package supplier using the `SPDX_PACKAGE_SUPPLIER` variable (e.g. in local.conf or similar). This variable has the same structure as a single `SPDX_AUTHORS` entry.

Reference an existing agent created by another variable:

```
SPDX_PACKAGE_SUPPLIER_ref = "SPDX_AUTHORS_myauthor"
```

Concluded License

The Concluded License for each package can be assigned with:

```
SPDX_CONCLUDED_LICENSE = "Applicable license"
```

CRA SBoM Praxis - Timestamps

Enabling Timestamps can be done by setting `SPDX_INCLUDE_TIMESTAMPS = "1"` in local.conf or similar.

WARNING: This will make your SBoMs non reproducible

CRA SBoM Praxis - purls

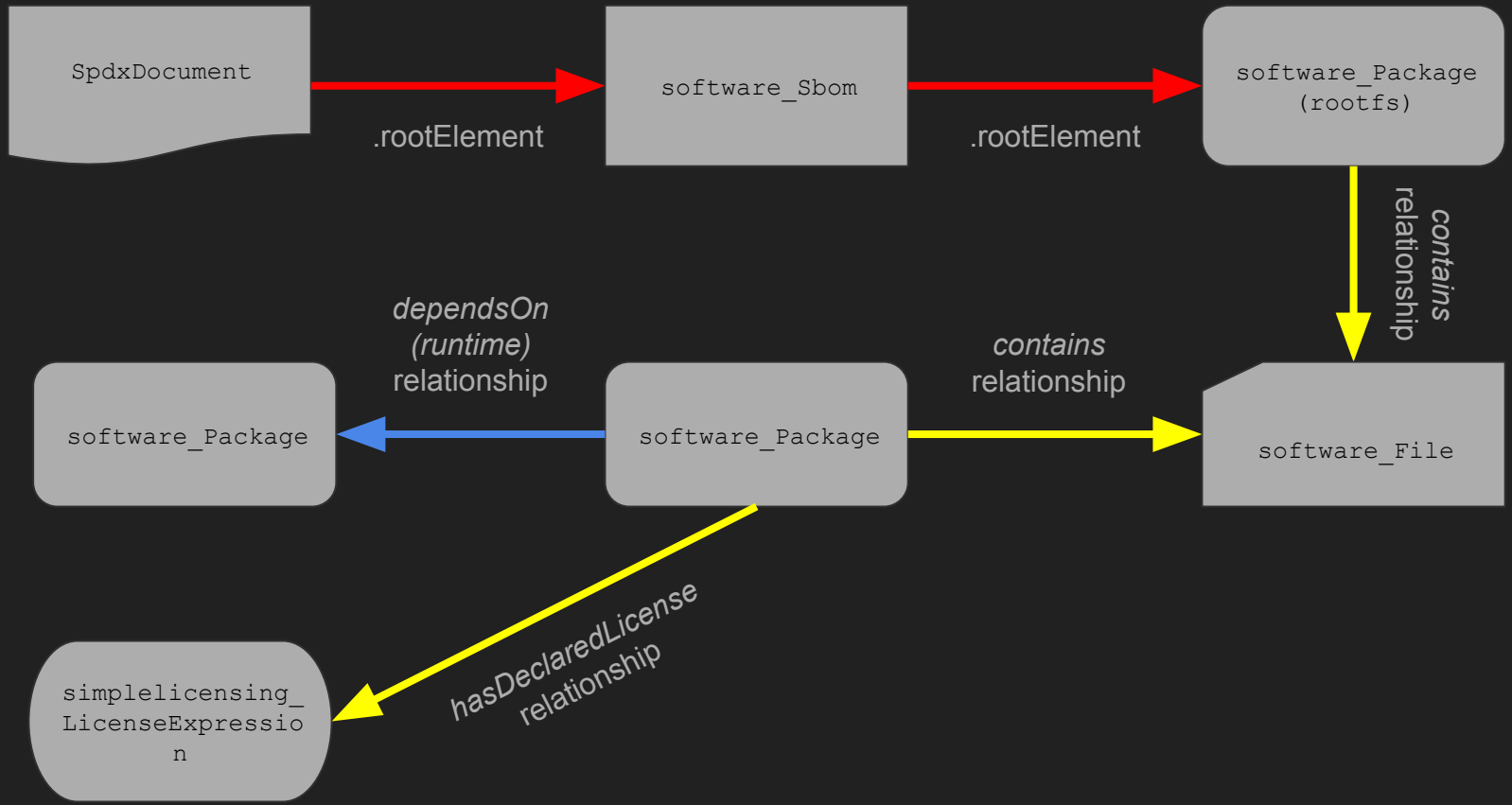
- Custom purls can be specified by appending to the `SPDX_PACKAGE_URLS` variable.
- PURLS are automatically assigned for:
  - All recipes using a `pkg:yocto/` scheme
  - CPAN recipes using a `pkg:cpan/` scheme
  - Cargo recipes using a `pkg:cargo/` scheme
  - Npm recipes using a `pkg:npm/` scheme
  - Golang recipes using `pkg:golang/` scheme
  - Pypi recipes using a `pkg:pypi/` scheme

SBoM Detail Level

All documents appears to be primarily concerned with the *runtime* dependencies

- Yocto SBoMs describe all runtime packages in an image, and their runtime dependencies. For a "traditional" self-contained embedded device, this may be sufficient
  - If you have 3rd party binaries or code not captured in the recipe: ⛔
  - Otherwise if Yocto knows about all the dependencies: ✅
- If your image is only one part of a larger system (e.g. a container image, micro-service, etc.) you'll likely need to add the runtime dependencies on the "first component outside the scope of delivery (BSI TR-03183)" ⛔

Navigating the Yocto SBoM



More Information

- Hours of SBoMs: My SBoM YouTube playlist

Questions?