SPDX 3.0 - a migration journey

Gary O’Neall, co-lead SPDX tech, maintainer SPDX Java libs
Agenda

SPDX 2 to SPDX 3

- Why SPDX 3?
- The approach to creating the spec
- Overview of the changes

Impact on the Java Libraries

- Current (soon to be previous) Java Library architecture
- Breaking changes in the model causing breaking changes in the library
- Reducing the impact of the breakage
- Reducing errors translating from the spec
Why SPDX 3.0?

Simplify - profiles

Flexibility - relationship structure

Interest in SPDX for non-licensing scenarios

- supporting **security** and **safety critical** application compliance requirements.
- **AI/ML** and **datasets** increasing need for transparency

Consolidate efforts between the SPDX community & OMG/CISQ efforts
SPDX-LICENSE-IDENTIFIER: CC-BY-4.0

Agenda

- SPDX 2 to SPDX 3
- Impact on the Java Libraries
  - The goal - or why the (breaking) changes?
  - Overview of the changes
  - The approach to creating the spec
  - Java Library architecture
  - Breaking changes in the model causing breaking changes in the library
  - Techniques to reduce the breakage
  - Generating the library from the schema

Timeline of SPDX Evolution - Use Case by Use Case

I AM THE Cavalry

Legislation: proposed software transparency, upatability & bill of material as reqts in safety critical sectors (automotive & healthcare)

NTIA: Software Transparency begins

OMG/CISQ

3T-SBOM: OMG/CISQ begins w/ CONOPs for Tool-to-Tool SBOM

Transition of SBOM work to DHS

Executive Order 14028

2010

SPDX begins

2011

SPDX 1.0

2013

SPDX 1.2

2015

SPDX 2.0

2018

2.1 to ISO

2019

2020

SPDX 2.3

2021

SPDX 3.0

2022

2023

SPDX 3rc1

2024+

SPDX 3.0

Standardized Single Package Information: Machine and human readable formats

Compliance Use Cases: Additional Project and License Information

Package Relations: 30+ additional use cases supported for complex packaging relationships and distribution scenarios

Security use cases: External references for vulnerabilities and product identification

SPDX & 3T-SBOM efforts merge: SPDX revises charter as an SDO

Free ISO Standard: ISO/IEC 5962 SPDX available

Format Interoperability

Profiles: New areas of use cases:
  - Build
  - Data
  - Security
  - AI
  - Lite

Profiles: New areas of use cases:
  - Services
  - Hardware
  - Safety

© 2024 The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. Case No: 23-01614-54
Specification Evolution

The SPDX Document

SPDX v1.2 File
- Creation information
- Package information
- Other licensing information
- File information
- Review information

SPDX v2.0 File
- Document creation information
- Package information
- File information
- Other licensing information
- Relationships
- Annotations

Serialization into common formats
- .json
- .json-ld
- .yaml
- .xml
- .rdf
- .spdx
- .xls

Online Specification
PAS ISO Submission
SPDX 3.0 Specification Infrastructure

Specification is being transformed into markdown describing

- Classes, Properties, Enumerations
- Metadata (type & cardinality) and description for each element.
- Will be able to automatically generate schema from this version (for JSON, YAML, RDF, XML, tag-value, etc.) and reduce errors.

Profiles can add their own Classes and Properties and may also restrict other profiles (e.g. values, cardinalities, ...)

See: https://github.com/spdx/spdx-3-model
Approach to Creating the Spec

Conceptual model (the “big picture”)
Approach to Creating the Spec

Conceptual model (the “big picture”)

Write the spec in Markdown using a specific formation
Approach to Creating the Spec

Initial / Temporary Conceptual model (the “big picture”)

Write the spec in Markdown using a specific formation

Tools / Generators

Website
Approach to Creating the Spec

Initial / Temporary Conceptual model (the “big picture”)

Tools / Generators

Website

Schema OWL/SHACL

Write the spec in Markdown using a specific formation
Approach to Creating the Spec

Initial / Temporary Conceptual model (the “big picture”)

Write the spec in Markdown using a specific formation

Tools / Generators

Website

Schema OWL/SHACL

Serialization Schemas
Approach to Creating the Spec

Initial / Temporary Conceptual model (the “big picture”)

Write the spec in Markdown using a specific formation

Tools / Generators

Ensures Consistency

Website

Schema OWL/SHACL

Serialization Schemas
Why SHACL / OWL?

- Captures semantic constraints as well as syntax constraints
- A superset of serialization schema functionality
  - Can be used to generate any of the schemas we’ve identified
- Tools exist in most language ecosystems to validate
Structural Changes

Profiles

- Conformance Requirements
  - Additional restrictions on properties (e.g. required license information in the licensing profile)

- Namespace
  - Organizes the vocabulary into more logical digestible units (e.g. you don’t have to know all the licensing terms if you’re only interested in security)

- Organization
  - SPDX work groups are organized around profiles
Structural Changes
External Document References

SPDX 2.3

SPDX Document

ExternalDocumentRef

Element

Element

Element

Element

Element

SPDX 3.0

Optional SPDX Documents

Element

Namespace Map

Imports

Element

Optional SPDX Documents
Structural Changes

Relationships

**SPDX 2.3**

- Element
  - relationship
  - relationship
  - Element

**SPDX 3.0**

- Relationship Element
  - Completeness
  - from
  - to
  - linkType
  - scope
  - requirement
  - Element
  - Element
Other SPDX 3.0 changes

- Better model for “Entities” (person, organization, tools)
- Renamed or removed confusing properties (e.g. filesAnalyzed)
- Added some useful classes and properties (e.g. Artifact, packageUrl)
- Profile specific classes and properties

Details: https://docs.google.com/document/d/1-oIHRnX1CssUS67Psv_sAg9Vd-pc81HF8MM0hA7M0hg
Changes “Big Picture”

- More flexibility with profiles, new relationship structure, independent annotations, external map structure

- Simpler with profiles, simpler snippets, no more filesAnalyzed, clearer naming

- More use cases with profiles

- For the details - see the Migration Document:
  https://docs.google.com/document/d/1-olHRnX1CssUS67Psv_sAq9Vd-pc81HF8MM0hA7M0hg/edit?usp=sharing
Impact on Java Libraries
Current Java Library Architecture

SPDX Java Library

Model

Utility
Current Java Library Architecture

SPDX Java Library

Model

Copy Manager

Storage Interface

Utility

Model Stores

JacksonStore (JSON, YAML, XML)

TagValueStore (SPDX Tag/Value)

RdfStore (JSON-LD, TTL, RDF/XML, …)

SpreadsheetStore (XLSX, XLS)
Breaking Changes in the Java Library

Storage Interface

- Expanded namespaces in profiles
  - Can no longer assume property names are unique
  - Added a “namespace” parameter in addition to the property name

```java
public Optional<Object> getValue(String documentUri, String id, String propertyName)
```

```java
public Optional<Object> getValue(String documentUri, String id, String namespace, String propertyName)
```
Breaking Changes

Model

- Compatibility package if you’re dealing with an SPDX 2.X document
- Relationships and Annotation structure - independent elements
- ExternalDocumentRef structure
- Replace Person/Organization strings with Agent
- Snippet simplifications
- Properties -> Relationships (e.g. packageFileName, concludedLicense)
- Plus a few other miscellaneous changes (see the migration document for details)
Reducing the Breakage

● Compatibility package with the version 2.X model
  ○ “CopyManager” handles upgrades from 2.X to 3.0

● SpdxModelFactory to direct you to the correct model based on SPDX spec version
  ○ getElements, getModelObject (by type), createModelObject (by type)

● “Relationship” list properties
  ○ Treats a relationship as a list property within a class
  ○ Can also access the relationships independently
  ○ Example: getConcludedLicense

● Compatible storage interface which looks up the namespace for property names
Reducing the Errors

Generating the Model Files

- Generated from the OWL/SHACL file
  - Currently implemented in Java using Mustache templates
  - Investigating switching to the Python generator written by Joshua Watt
  - Consistency from the spec markdown files all the way through the code
- Generate serialization specific schemas from the OWL/SHACL file
  - JSON Schema
  - XML Schema
- Generate verification code based on the OWL/SHACL schema
- For RDF serializations, verify directly against the OWL/SHACL schema
  - Not implemented with other serialization formats due to the “weight” of the RDF dependencies
SPDX 3.0 Java Library Architecture

SPDX Java Library

Model
- Generated
- Copy
- Compatibility

Utility

Storage Interface

Model Stores
- JacksonStore (JSON, YAML, XML)
- TagValueStore (SPDX Tag/Value)
- RdfStore (JSON-LD, TTL, RDF/XML, …)
- SpreadsheetStore (XLSX, XLS)