SPDX 2.3 to SPDX 3.0 Compatibility Analysis

Purpose and Background

This document is a work in progress description of changes in the SPDX 3.0 spec that will impact compatibility between SPDX 2.3 and SPDX 3.0 “Documents” (to use the SPDX 2.3 terminology).

As a major version upgrade, the 3.0 spec is allowed to have breaking changes. As the 3.0 spec will include several new use cases in the security field as well as machine learning and safety, breaking changes are likely. We will also address some of the deficiencies identified in the 2.3 spec based on usage in the real world. It is also a goal to minimize the changes and reduce the effort required by existing SPDX uses to convert from the 2.3 version documents to the 3.0 version documents.

This document lays out currently identified differences between the 2.3 spec and the 3.0 spec to help guide discussions on balancing the goal of minimizing breaking changes with the other goals for the SPDX 3.0 spec.

Please note that the 3.0 spec is a work in progress and will likely change in response to this analysis, so it is probably a bit soon to take action in any existing tooling.

The document is organized in the following sections:

- Structural Differences - These are the most significant breaking changes requiring a change in logic to handle a different model or structure for the information. Each structural difference will describe the change, describe an approach to translate from 2.3 to 3.0, and provide a rationale for the change.
- Properties removed - Any properties which are longer supported. This section will include rationale for removing the properties.
- Renamed entities - Any properties, types or classes which have been renamed. In addition to the old and new names, a rationale for the name change will be included.

This document is based on the following references:

- 3.0 Migration Mapping Spreadsheet
- SPDX 3 Model Diagram
- SPDX 3.0 Model Repository
- SPDX 2.3 RDF Ontology
- SPDX 2.3 Specification
- SPDX 2.3 JSON schema
- SPDX 2.3 Examples

Update History

Date	Who	Summary of Changes
24 Feb 2023	Gary O'Neall	Initial Document
8 May 2023	Several contributors	RC1 Draft - updated justifications, many corrections
31 Jan 2024	Gary O'Neall	Update for RC2 release
5 April 2024	Gary O'Neall	Update for final release

Structural Differences

External Document Reference

Description of Change

The purpose of the SPDX 2.3 structure "ExternalDocumentRef" is now covered by two separate structures:

- NamespaceMap which maps short identifiers used in serializations to full namespace URI's to support terseness in serialization of element identifiers
- ExternalMap which maps an element identifier for an element defined externally to verification and location information

The externalDocumentRef property on the SpdxDocument has been replaced by import property and namespace property.

Another change is the SPDX document checksum field has been replaced with a "verifiedUsing" property on the ElementCollection. The "verifiedUsing" which has 0 or more "IntegrityMethod" which should be the checksum of the SPDX document.

Translating from 2.3 to 3.0

Each ExternalDocumentRef instance will translate as follows:

- An entry would be created in the Namespace map for the external document namespace
    - The value of the DocumentRef-[idstring] would be used for the prefix property in the NamespaceMap.
    - The value of the documentNamespace appended with a "#" would be used for the namespace in the NamespaceMap.
- An entry would be created in the ExternalMap for the external document ref

    - A string identifier consisting of the DocumentRef-[idstring] (the same value as the prefix in the NamespaceMap) concatenated with a ":" and then concatenated with "SPDXRef-DOCUMENT" would be used for the externalSpdxId in the ExternalMap.
    - An integrity method of "Hash" will be created with the same information as the checksum property and will be referenced using the "verifiedUsing" property on the ExternalMap entry.
- An entry would be created in the ExternalMap for each element referenced in the current SpdxDocument that is originally specified in the referenced SpdxDocument.
    - A string identifier consisting of the DocumentRef-[idstring] (the same value as the prefix in the NamespaceMap) concatenated with a ":" and then concatenated with the local portion of the element identifier would be used for the externalSpdxId in the ExternalMap
    - A "definingDocument" property would be specified containing a string identifier consisting of the DocumentRef-[idstring] concatenated with a ":" and then concatenated with "SPDXRef-DOCUMENT". This is a shortcut linkage to tie the referenced element to its defining SpdxDocument for verification and location information.

Rationale

A key difference between SPDX 2.3 and SPDX 3.0 is that in SPDX 2.3 elements are always expressed within or referenced in relation to a single enclosing SpdxDocument while in SPDX 3.0 a key design principle is that all elements may be expressed and referenced independent of any other element including SpdxDocument. This independence is required to support a variety of content exchange and analysis use cases.

For example, in SPDX 2.3 if you wish to express even a single package you specify it within an SpdxDocument and its identifier namespace is restricted to the namespace of the SpdxDocument. In SPDX 3.0 you could specify a single package within an SpdxDocument element (or any other subclass of ElementCollection such as Bundle, Bom, Sbom, etc.) but you could also simply specify it on its own without any enclosing collection element. In addition, in SPDX 3.0 the identifier of the package may share a namespace with an enclosing collection element such as SpdxDocument if desired but it is equally valid for it to have any namespace desired unconstrained by any other element namespace whether it is expressed within a collection element such as SpdxDocument or not.

In this example, in SPDX 2.3 if you referenced the package within the same SpdxDocument that it is defined in you would utilize the local portion of its identifier and presume that the namespace is the same as the SpdxDocument namespace. If you referenced it from an SpdxDocument other than the one it is defined in you would use an ExternalDocumentRef to specify a prefix name for the other SpdxDocument to be used within the current SpdxDocument, the URI namespace/identifier for the other SpdxDocument, and a checksum for the other SpdxDocument. To reference the package you would then use an identifier combining the external document ref prefix and the local portion of the identifier.

The ExternalDocumentRef structure in SPDX 2.3 is based on the presumptions that elements are always defined within SpdxDocuments, that external elements can always be referenced via a containing SpdxDocument and that element identifiers have a namespace from their original containing SpdxDocument. None of these three presumptions hold true for SPDX 3.0 so a slightly modified structure is necessary to support the two use cases previously covered by ExternalDocumentRef in SPDX 2.3: 1) the ability to specify identifier namespace prefixes and accompanying namespaces for SPDX elements to support more terse serialized expression of content with integrity across serialization forms, 2) the ability to specify which elements in the current subclass of ElementCollection (e.g., SpdxDocument) are only referenced from that collection and defined elsewhere, along with details regarding their verification and location.

The Namespace map structure in SPDX 3.0 fully supports the namespace prefixing use case for SpdxDocuments previously covered by ExternalDocumentRef but also equally covers the same use case capability for all element types and for any number of element identifier namespaces (in SPDX 3.0 all elements within an SpdxDocument are not required to have the same namespace and can actually be any desired mix of namespaces) to support this capability required in SPDX 3.0.

The ExternalMap structure in SPDX 3.0 fully supports the external element (including SpdxDocument elements) referencing use case for SpdxDocuments previously covered by ExternalDocumentRef but also equally covers the same use case capability for any elements whether they were originally defined within an SpdxDocument or not to support this capability required in SPDX 3.0. The ExternalMap structure in SPDX 3.0 provides the ability to specify verification and location details for any element, not just SpdxDocuments, if appropriate but also provides simple linkage, using the “definingDocument" property, from element entries in the ExternalMap to SpdxDocument entries in the ExternalMap where the elements were defined within the SpdxDocument and verification of the elements can be achieved via proxy to the SpdxDocument “verifiedUsing” information (this is how the SPDX 2.3 ExternalDocumentRef structure currently works).

Agent

Description of Change

The creator property in SPDX 2.3 has been replaced by createdBy and createdUsing properties with a type Agent and Tool resp. The supplier property has been replaced by a property suppliedBy with a type Agent. Additional suppliers can be provided with a a relationship to an availableFrom relationship. The originator property type has been replaced with the originatedBy property with a type Agent.

An Agent can be a Person, Organization, or Software Agent. It can also just be an Agent if it is not known what specific type an Agent is.

Translating from 2.3 to 3.0

The SPDX 2.3 creator string would be parsed and the appropriate Person, Organization or Tool would be created depending on if the prefix is “Person: ”, “Organization:” or “Tool: ” resp. The required createdBy field for Agent or Tool may point to itself if no other information is available. The createdUsing property would be used for Tool whereas the createdBy property would be used for Person and Organization. The name would map to the “name” property. If an email address is present, it would translate to an external identifier.

Note that in 3.0 the createdBy is a required field. There will be situations where only a Tool is provided. In that case, createdBy should point to a SoftwareAgent should be created using the same information as the Tool.

Rationale

The 3.0 format is more machine readable and structured (e.g. you do not need to parse the type from the string value). It is also more flexible in that an Agent can be used even if it is not known what the Agent type is.

File Type

Description of Change

The FileType enumeration has been replaced by two fields, the media type string as maintained by IANA for the content of the file and an enumeration of SoftwarePurpose for the purpose of the file.

The property name fileType has been replaced by a property name contentType.

Translating from 2.3 to 3.0

Rationale

One of the things that we identified is that `FileType` was being used for two things:

1. Describing the purpose of the file.
2. Describing the type of content in the file.

For SPDX 3.0 we split this into two properties:

- `SoftwarePurpose` to capture the purpose (which is of type `SoftwarePurpose`).
- `ContentType` to capture the type of content (which is of type `MediaType`).

The name `ContentType` was chosen to mirror the Content-Type header in HTTP (which is also of type MediaType) and to express that this is describing the type of content (as opposed to

metadata, headers, or something else). For example, if (and not saying we would) we extended `File` in the future to be able to capture the type of executable header a file has (e.g. ELF), that could also be of type `MediaType` but the property name might be `ExecutableHeaderType`.

An example conversion table from SPDX 2.3 `FileType` to SPDX 3.0 `ContentType` or `SoftwarePurpose` can look like this:

SPDX 2 File Type	**SPDX 3 Software Purpose**	**SPDX 3 Content Type**
ARCHIVE	Archive	
BINARY		application/octet-stream
SOURCE	Source	
TEXT		text/plain
APPLICATION	Application	
AUDIO		audio/*
IMAGE		image/*
VIDEO		video/*
DOCUMENTATION	Documentation	
SPDX		text/spdx
OTHER	Other	

(Based on https://github.com/spdx/spdx-3-model/issues/82#issuecomment-1441476885)

Package File Name

Description of Change

The packageFileName property and packageChecksum property has been replaced by a relationship from a Package to a File. A relationship type of hasDistributionArtifact should be used.

Translating from 2.3 to 3.0

Create an SPDX File with the name from the packageFileName and a verifiedUsing value from the packageChecksum for a single file. If the packageFileName is a directory, then the SPDX File is created with the directory name and is verified using the gitoid property on the File and a

fileKind of directory. Create a hasDistributionArifact relationship from the SPDX Package to the SPDX File.

Rationale

Providing a File relationship to the download location will include more detailed and complete information about the package file name used.

External Identifiers

Description of Change

In SPDX 3.0, a new property externalIdentifiers and a new type ExternalIdentifier is introduced. This is in addition to retaining the ExternalRef property and classes.

In SPDX 2.3, both identifiers and references were captured in the externalRef property for packages.

In addition to the structural changes, the “url” ExternalRef type was removed and is replaced by the “securityOther” ExternalRef type.

Translating from 2.3 to 3.0

The following ExternalRef Types should be converted to ExternalIdentifiers:

- cpe22Type
- cpe23Type
- swid
- purl
- swh
- gitoid

All other ExternalRef types should remain as ExternalRef’s.

The url ExternalRef type should be converted to a “securityOther”.

Rationale

Distinguishing identifiers from references is key to several integrity and provenance use cases. Creating a separate property and type enables easier identification of identifiers.

Package URL

Description of Change

In SPDX 3.0, Package URL is a new property for Artifact which is a superclass of Package.

Package URL is an External Ref type in SPDX 2.3.

Translating from 2.3 to 3.0

If there is a single ExternalReference of type purl without the optional ExternalRef comment property, place that in the packageUrl property.

Rationale

Package URL is a very common method of identifying software packages. Moving this to a property makes it significantly simpler to find and correlate Package URL identifiers.

Annotation

Description of Change

Annotations are now subclasses of Element, so it inherits a number of new optional properties including names, annotations, and its own relationships.

Annotations are no longer a property of an Element. It is now a standalone element with a “subject” field which points to the Element being annotated.

Translating from 2.3 to 3.0

A new Annotation element would be created for every annotation property in an element (Package, File or Snippet). The subject property would point to the Element which has the Annotation as a property.

The annotator from SPDX 2.3 should be translated to one of the creators for the creationInfo for the Annotation and the annotationDate should be translated to the created field in the same creationInfo. The creationInfo for the Annotation should be the creationInfo of the SPDX 2.3 document.

The SPDX 2.3 “comment” should use the statement field in SPDX 3.0.

Rationale

Changing from a property to a standalone element allows for relationships to exist outside the element itself (e.g. you can now create an amended SPDX document which has a new annotation for an element defined in the original document). This also supports third parties' ability to assert Annotations on Elements that they did not create.

Relationship

Description of Change

The structure of the Relationship class has changed to have a single direction and allow more than one related SPDX Elements. Relationships are now subclasses of Element, so it inherits a number of new optional properties including names, annotations, and its own relationships.

Relationships are no longer a property of an Element. It is now a standalone element with a “from” and “to” field.

A new property “completeness' ' complements the use of NONE and NOASSERTION for the related SPDX elements.

Translating from 2.3 to 3.0

The “from” property would be populated by the SPDX Element which has the relationship property. The “to” property will be the relatedSpdxElement.

When translating the relationshipType, the “from” and “to” may need to be swapped - the table below will have a “Y” in the “Swap to and from?” column when this is necessary.

The completeness property would be constructed based on the following:

- “to” value is NONE: complete
- “to” value is NOASSERTION: noAssertion
- “to” value is an SPDX element: No value for the completeness - uses the default

Relationship migration is being worked out in the relationships spreadsheet. Once completed, the following table will reflect the translation for relationship types from SPDX 2.3 to SPDX 3.0:

SPDX 2.3 Relationship Type	SPDX 3.0 Relationship Type	Swap to and from?	LifecycleScopeType
AMENDS	amendedBy	Y	
ANCESTOR_OF	ancestorOf		
BUILD_DEPENDENCY_OF	dependsOn		build

BUILD_TOOL_OF	usesTool		build (all lifecycle scope could be appropriate)
CONTAINED_BY			
CONTAINS	contains		
COPY_OF	copiedTo		
DATA_FILE_OF	hasDataFile		
DEPENDENCY_MANIFEST_OF	hasDependencyManifest		
~~DEPENDENCY_OF~~			
DEPENDS_ON	dependsOn		LifecycleScopeType
DESCENDANT_OF	decendentOf		
~~DESCRIBED_BY~~			
DESCRIBES	describes		
DEV_DEPENDENCY_OF	dependsOn		development
DEV_TOOL_OF	usesTool		development
DISTRIBUTION_ARTIFACT	hasDistributionArtifact		
DOCUMENTATION_OF	hasDocumentation		
DYNAMIC_LINK	hasDynamicLink		build, runtime
EXAMPLE_OF	hasExample		
EXPANDED_FROM_ARCHIVE	expandsTo		
FILE_ADDED	hasAddedFile		
FILE_DELETED	hasDeletedFile		
FILE_MODIFIED	modifiedBy		
~~GENERATED_FROM~~			
GENERATES	generates		
HAS_PREREQUISITE	hasPrequisite		lifecycle scope
METAFILE_OF	hasMetadata		
OPTIONAL_COMPONENT_OF	hasOptionalComponent		
OPTIONAL_DEPENDENCY_OF	hasOptionalDependency		lifecycle scope
OTHER	other		
PACKAGE_OF	packagedBy		

PATCH_APPLIED	patchedBy		
~~PATCH_FOR~~			
~~PREREQUISITE_FOR~~			
PROVIDED_DEPENDENCY_OF	hasProvidedDependency		lifecycle scope
REQUIREMENT_DESCRIPTION_FOR	hasRequirement		lifecycle scope
RUNTIME_DEPENDENCY_OF	dependsOn		runtime
SPECIFICATION_FOR	hasSpecification		lifecycle scope
STATIC_LINK	hasStaticLink		lifecycle scope
TEST_CASE_OF	hasTestCase		
TEST_DEPENDENCY_OF	dependsOn		test
TEST_OF	hasTest		lifecycle scope
TEST_TOOL_OF	usesTool		test
VARIANT_OF	hasVarient		

Rationale

The addition of the completeness attribute is clearer than the use of NONE and NOASSERTION.

Changing from a property to a standalone element allows for relationships to exist outside the element itself (e.g. you can now create an amended SPDX document which has a new relationship for an element defined in the original document). This enables primary Element creating parties as well as third parties to express significantly greater contextual detail among content they create as well as content created by others.

Note: A proposed change in cardinality for the “to” property is being tracked in issue #129.

Snippet

Description of Change

Byte and line range types have been changed from a StartEndPointer type to a PositiveIntegerRange. Byte range is now optional.

Translating from 2.3 to 3.0

Iterate through the “ranges” property. Any startPointer and endPointer with a property of “offset” would be translated to a snippetByteRange property. Any startPointer and endPointer with a property of “lineNumber” would translate to a snippetLineRange property.

A new Relationship would be created with the “from” pointing to the snippetFromFile and the “to” pointing to the Snippet. They relationshipType would be CONTAINS.

Rationale

Using the W3C Pointer standard introduced significant complexity in the SPDX 2.X specification. Although there may be some benefit in using a published standard, we have not found any instances where the W3C Pointer ontology was useful for SPDX use cases.

Changing the snippetFromFile from a property to a relationship [to be filled in].

SpecVersion

Description of Change

The type of SpecVerion is changed from a simple string without constraints to a SemVer string which must follow the Semantic Versioning format.

This adds a constraint where a patch version is required. Previous usage of the SpecVersiononly included the major and minor version.

Translating from 2.3 to 3.0

Add a patch version of “0” to any previous spec version.

Rationale

The additional constraints align with best practices for versioning strings.

LicenseListVersion

Description of Change

The type of LicenseListVersion is changed from a simple string without constraints to a SemVer string which must follow the Semantic Versioning format.

This adds a constraint where a patch version is required. Previous usage of the SPDX license list only included the major and minor version.

Translating from 2.3 to 3.0

Add a patch version of “0” to any previous license list version.

Rationale

The additional constraints align with best practices for versioning strings.

Properties Removed

Below is a list of properties present in 2.3 and not present in 3.0. The Range / Where used is where the property was used in the SPDX 2.3 model.

example

SPDX 2.3 Model Name

example

Tag/Value Name

Not used

Range / Where Used

LicenseException

Rationale

This field has not been used.

LicenseInfoInFiles

SPDX 2.3 Model Name

licenseInfoInFiles

Tag/Value Name

LicenseInfoInFiles

Range / Where Used

Package

Rationale

This field is redundant with the declaredLicense property in the Files contained in the Package. It is recommended that the licenseInfoInFiles can be added as an Annotation to the Package in the format: “SPDX 2.X LicenseInfoInFiles: [expression1], [expression2]” where the [expressions] are the string representation of the license expressions.

FilesAnalyzed

SPDX 2.3 Model Name

filesAnalyzed

Tag/Value Name

FilesAnalyzed

Range / Where Used

Package

Rationale

Many users of the SPDX 2.X spec reported this property as very confusing.
NOTE: This is being tracked in Issue #84

Naming Differences

Below is a list of properties and classes where the name has been changed from 2.3 to 3.0. The Range / Where used is where the property was used in the SPDX 2.3 model.

Release Date

SPDX 2.3 Model Name

releaseDate

Tag/Value Name

ReleaseDate

New Name

releaseTime

Range / Where Used

Package

Rationale

Better reflects the granularity of the field.

Build Date

SPDX 2.3 Model Name

buildDate

Tag/Value Name

BuildDate

New Name

buildTime

Range / Where Used

Package

Rationale

Better reflects the granularity of the field.

Valid Until Date

SPDX 2.3 Model Name

validUntilDate

Tag/Value Name

ValidUntilDate

New Name

validUntilTime

Range / Where Used

Package

Rationale

Better reflects the granularity of the field.

External Document Reference

SPDX 2.3 Model Name

externalDocumentRef

Tag/Value Name

ExternalDocumentRef

New Name

import

Range / Where Used

SpdxDocument (Creation Information)

Rationale

Feedback from SPDX 2.X usage is that externalDocumentRef is confusing due to the similar externalRef property.

NOTE: See structural changes related to this property

Checksum Class / Data Type

SPDX 2.3 Model Name

Checksum class name and checksum property name

Tag/Value Name

FileChecksum, PackageChecksum

New Name

verifiedUsing property and Hash class

Range / Where Used

Package, File

Rationale

More general concept allowing for different verification algorithms for different scenarios.
Note: Being tracked in issue #90.

Checksum Algorithm

SPDX 2.3 Model Name

checksumAlgorithm

Tag/Value Name

N/A - parsed from a string following the Checksum: keyword.

New Name

hashAlgorithm

Range / Where Used

Package, File

Rationale

The term “hash” better represents the intent of this property which is to validate the integrity of the data whereas the term “checksum” is typically for the purpose of error checking.

Name

SPDX 2.3 Model Name

packageName, fileName

Tag/Value Name

PackageName, FileName

New Name

name

Range / Where Used

Package, File

Rationale

In the SPDX 2.3 RDF Ontology, both spdx:fileName and spdx:packageName are sub-properties of spdx:name. The OWL has a restriction that spdx:File has exactly one spdx:fileName and spdx:Package has exactly one spdx:packageName.

Changing these restrictions to just spdx:name would simplify the model.

Version

SPDX 2.3 Model Name

versionInfo

Tag/Value Name

PackageVersion

New Name

packageVersion

Range / Where Used

Package

Rationale

This change would make the Tag/Value and RDF values consistent.

Home Page

SPDX 2.3 Model Name

doap:homepage

Tag/Value Name

PackageHomePage

New Name

homePage

Range / Where Used

Rationale

This is being tracked in issue #132.

Annotation Comment

SPDX 2.3 Model Name

rdfs:comment

Tag/Value Name

AnnotationComment

New Name

statement

Range / Where Used

Element (Package, File, Snippet)

Rationale

The rdfs:comment property is optional and has slightly different semantics in other uses (e.g. comments on Elements). Changing the property name clearly distinguishes this usage as a mandatory property for an Annotation.

With Exception Operator

SPDX 2.3 Model Name

WithExceptionOperator
member property in WithExceptionOperator
licenseException property in WithExceptionOperator

Tag/Value Name

With (part of License Expression)

New Name

WithAdditionOperator

subjectLicense
subjectAddition

Range / Where Used

Package, File, Snippet

Rationale

Custom Additions have been added in SPDX 3.0 which operate in a similar manner to listed License Exceptions. The new type and property names are more general to accommodate both custom additions and listed license Exceptions.

License Exception

SPDX 2.3 Model Name

LicenseException
licenseExceptionId property in LicenseException
licenseExceptionText property in LicenseException
name property in LicenseException

Tag/Value Name

Not used in Tag/Value

New Name

ListedLicenseException
additionId
additionText
additionName

Range / Where Used

Package, File, Snippet

Rationale

Custom Additions have been added in SPDX 3.0 which operate in a similar manner to listed License Exceptions. The new type and property names are more general to accommodate both custom additions and listed license Exceptions.

ExtractedLicenseInfo

SPDX 2.3 Model Name

ExtractedLicenseInfo

Tag/Value Name

ExtractedText

New Name

CustomLicense

Range / Where Used

Package, File, Snippet, Document

Rationale

The SPDX 2.X term implied that the only property was text when in fact there are several properties in common with the listed licenses. See issue #233 for context.

licenseComment

SPDX 2.3 Model Name

licenseComment

Tag/Value Name

LicenseName

New Name

name

Range / Where Used

License, ListedLicense, ExtractedText

Rationale

“name” is used in the Element class. Since License is a type of (subclass of) Element, it should use the same field otherwise there would be redundant fields for the same purpose.

LicenseComment

SPDX 2.3 Model Name

licenseComment

Tag/Value Name

LicenseComment

New Name

comment

Range / Where Used

License, ListedLicense

Rationale

“comment” is used in the Element class. Since License is a type of (subclass of) Element, it should use the same field otherwise there would be redundant fields for the same purpose.

LicenseID

SPDX 2.3 Model Name

licenseId

Tag/Value Name

LicenseId

New Name

spdxId

Range / Where Used

License, ListedLicense

Rationale

“spdxId” is used in the Element class. Since License is a type of (subclass of) Element, it should use the same field otherwise there would be redundant fields for the same purpose.

Range / Where Used

License, ListedLicense

Rationale

Primary Package Purpose

SPDX 2.3 Model Name

primaryPackagePurpose

Tag/Value Name

PrimaryPackagePurpose

New Name

primaryPurpose

Range / Where Used

Package

Rationale

The purpose property is now available for files and snippets in addition to Package resulting in a more general name of primaryPurpose.

Note that additional purposes can be added using the additionalPurpose property.

JSON Format Changes

Please note that at this time, the SPDX 3.0 serialization formats are still in flux, the information below is subject to change.

Pluralization

In SPDX 2.3, any property names which had a JSON Array as the value type were pluralized. In 3.0, we will consistently use the same name as in the model which is the singular form.

The rationale for this change is consistency in the formats and the fact that some plural forms were not translated correctly.