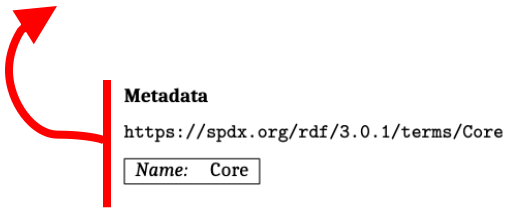
8 Core ~~Profile~~

Summary

The basis for all SPDX profiles.

Description

The Core namespace defines foundational concepts serving as the basis for all SPDX-3.0 profiles. ~~Figure 5 below shows the logical model for Core profile, for the Software profile, and the non-element classes, enumerations, and data types for both~~.



Metadata

`https://spdx.org/rdf/3.0.1/terms/Core`

Name:	Core

9 Software ~~Profile~~

Summary

Everything having to do with software.

Description

The Software namespace defines concepts related to software artifacts. ~~Figure 6 below shows the logical model for Core profile, for the Software profile, and the non-element classes, enumerations, and data types for both.~~

Metadata

`https://spdx.org/rdf/3.0.1/terms/Software`

10 Security ~~Profile~~

Summary

The Security Profile captures security related information.

Description

The Security Profile captures security related information. ~~Figure 7 below shows the logical model for the Security profile and its enumerations.~~

Metadata

Name	Security

https://spdx.org/rdf/3.0.1/terms/Security

11 Licensing ~~Profile~~

Summary

The Licensing Profile defines a minimum set of license information to facilitate compliance with typical license use cases.

Description

The Licensing profile only contains the additional requirement that any Software Artifact must have a concludedLicense Relationship.

Classes and Property restrictions are defined in the SimpleLicensingProfile (Classes and Properties associated with string license expressions) and in the ExpandedLicensingProfile (Classes and Properties used for a fully parsed syntax tree of license expressions).

There are 2 relationship types related to licensing - declaredLicense and concludedLicense.

A declaredLicense identifies the license information actually found in the Software Artifact, for example as detected by use of automated tooling.

This field is not intended to capture license information obtained from an external source, such as a package's website. Such information can be included, as needed, in the concludedLicense field.

A declaredLicense may be expressed differently in practice for different types of Software Artifacts. For example:

- for Packages:
- would include license info for the Package as a whole, found in the Package itself (e.g., LICENSE file, README file, metadata in the Package, etc.)
- would not include any license information that is not in the Package itself (e.g., license information from the project's website or from a third party repository or website)
- for Files:
- would include license info found in the File itself (e.g., license header or notice, comments indicating the license, SPDX-License-Identifier expression)
- would not include license info found in a different file (e.g., LICENSE file in the top directory of a repository)
- for Snippets:
- would include license info found in the Snippet itself (e.g., license notice, comments, SPDX-License-Identifier expression)
- would not include license info found elsewhere in the File or in a different File (e.g., comment at top of File if it is not within the Snippet, LICENSE file in the top directory of a repository)

A declaredLicense relationship to NoneLicense indicates that the corresponding Package, File or Snippet contains no license information whatsoever.

A declaredLicense relationship to NoAssertionLicense indicates that one of the following applies: * the SPDX data creator has attempted to but cannot reach a reasonable objective determination; * the SPDX data creator has made no attempt to determine this field; or * the SPDX data creator has intentionally provided no information (no meaning should be implied by doing so).

If a declaredLicense relationship is not present, no assumptions can be made about whether or not a declaredLicense exists. Note that a missing declaredLicense is not the same as a relationship to NoAssertionLicense since the latter is a "known unknown" whereas no assumptions can be made from a missing declaredLicense relationship.

A concludedLicense is the license identified by the SPDX data creator, based on analyzing the license information in the Software Artifact and other information to arrive at a reasonably objective conclusion as to what license governs the Software Artifact.

A concludedLicense relationship to NoneLicense indicates that the SPDX data creator has looked and did not find any license information for this Software Artifact.

A concludedLicense relationship to NoAssertionLicense indicates that one of the following applies: * the SPDX data creator has attempted to but cannot reach a reasonable objective determination; * the SPDX data creator has made no attempt to determine this field; or * the SPDX data creator has intentionally provided no information (no meaning should be implied by doing so).

If a concludedLicense is not present, no assumptions can be made about whether or not a concludedLicense exists. Note that a missing concludedLicense is not the same as a relationship to a NoAssertionLicense since the latter is a "known unknown" whereas no assumptions can be made from a missing concludedLicense relationship.

A written explanation of a relationship to a NoAssertionLicense MAY be provided in the comment field for the relationship.

If the concludedLicense for a Software Artifact is not the same as its declaredLicense, a written explanation SHOULD be provided in the concludedLicense relationship comment field.

Figure 8 below shows the logical model for the Simple and Expanded Licensing profiles.

Metadata

Name	Licensing

`https://spdx.org/rdf/3.0.1/terms/Licensing`

11.1 SimpleLicensing ~~Profile~~

Summary

Additional metadata relating to software licensing.

Description

The SimpleLicensing profile provides classes and properties to express licenses as a license expression string. It also provides the base abstract class, AnyLicenseInfo, used for references to license information. The SimpleLicensingText class provides a place to record any license text found that does not match a license on the SPDX license list.

The ExpandingLicensing profile can be used to represent the complete parsed license expressions.

Metadata

`https://spdx.org/rdf/v3/SimpleLicensing`

Name	SimpleLicensing

~~SimpleLicensing~~ Classes

11.1.1 AnyLicenseInfo

Summary

Abstract class representing a license combination consisting of one or more licenses (optionally including additional text), which may be combined according to the SPDX license expression syntax.

Description

An AnyLicenseInfo is used by licensing properties of software artifacts. It can be a NoneLicense, a NoAssertionLicense, single license (either on the SPDX License List or a custom-defined license); a single license with an "or later" operator applied; the foregoing with additional text applied; or a set of licenses combined by applying "AND" and "OR" operators recursively.

Metadata

`https://spdx.org/rdf/v3/SimpleLicensing/AnyLicenseInfo`

Name	AnyLicenseInfo
Instantiability	Abstract
SubclassOf	/Core/Element

Properties

Property	Type	minCount	maxCount

12 Dataset Profile

Summary

Everything having to do with datasets.

Description

The Dataset profile provides meta-data about data files. Figure 9 below shows the logical model for the Dataset profile with its classes and enumerations.

Metadata

Name	Dataset

https://spdx.org/rdf/3.0.1/terms/Dataset

13 AI Profile

Summary

Additional metadata based on software profile, that is useful for ai applications and models.

Description

The AI profile namespace defines concepts related to AI application and model artifacts. Figure 101 below shows the logical model for the AI profile with its classes and enumerations.

Metadata

Name	AI

https://spdx.org/rdf/3.0.1/terms/AI

14 Build ~~Profile~~

Summary

The Build Profile defines the set of information required to describe an instance of a Software Build.

Description

A Software Build is defined here as the act of converting software inputs into software artifacts using software build tools. Inputs can include source code, config files, artifacts that are build environments, and build tools. Outputs can include intermediate artifacts to other build inputs or the final artifacts.

The Build profile provides a subclass of Element called Build. It also provides a minimum set of required Relationship Types from the Core profile:

- hasInputs: Describes the relationship from the Build element to its inputs.
- hasOutputs: Describes the relationship from the Build element to its outputs.
- invokedBy: Describes the relationship from the Build element to the Agent that invoked it.

In addition, the following Relationship Types may be used to describe a Build.

- hasHost: Describes the relationship from the Build element to the build stage or host.
- configures: Describes the relationship from a configuration to the Build element.
- ancestorOf: Describes a relationship from a Build element to Build eelements that describe its child builds.
- decendentOf: Describes a relationship from a child Build element to its parent.
- usesTool: Describes a relationship from a Build element to a build tool.

All relationships in the Build Profile are scoped to the "build" LifecycleScopeType period.

The hasInputs relationship can be applied to a config file or a build tool if the nature of these inputs are not known at the creation of an SPDX document.

Metadata

Name	Build

`https://spdx.org/rdf/3.0.1/terms/Build`

15 Lite ~~Profile~~

Summary

The SPDX Lite profile defines a subset of the SPDX specification, from the point of view of use cases in some industries. SPDX Lite aims at the balance between the SPDX standard and actual workflows in some industries.

Description

The SPDX Lite profile consists of mandatory fields from the Document Creation and Package Information sections and other basic information.

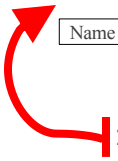
The mandatory part of the Package information in SPDX Lite is basic but useful for complying with licenses. It is easy to understand licensing information by reading an SPDX Lite file. It is easy to create manually an SPDX Lite file by anyone who does not have enough knowledge about licensing information, so that tools are not necessarily required to create an SPDX Lite file.

SPDX Lite has affinity with SPDX tools due to its containing the mandatory part of the Document Creation and Package Information in the SPDX Lite definition.

An SPDX Lite document can be used in parallel with SPDX documents in software supply chains.

Metadata

Name	Lite



`https://spdx.org/rdf/3.0.1/terms/Lite`

16 Extension ~~Profile~~

Summary

Everything having to do with SPDX extensions.

Description

The Extension namespace defines the abstract Extension class serving as the base for all defined extension subclasses. ~~Figure 12 below shows the logical model for the Extension profile~~.

Metadata

Name	Extension

`https://spdx.org/rdf/3.0.1/terms/Extension`