Annex A: SPDX license expressions

(normative)

A.1 Overview

Often a single license can be used to represent the licensing terms of a source code or binary file, but there are situations where a single license identifier is not sufficient. A common example is when software is offered under a choice of one or more licenses (e.g., GPL-2.0-only OR BSD-3-Clause). Another example is when a set of licenses is needed to represent a binary program constructed by compiling and linking two (or more) different source files each governed by different licenses (e.g., LGPL-2.1-only AND BSD-3-Clause).

SPDX License Expressions provide a way for one to construct expressions that more accurately represent the licensing terms typically found in open source software source code. A license expression could be a single license identifier found on the SPDX License List; a user defined license reference denoted by the LicenseRef-[idString]; a license identifier combined with an SPDX exception; or some combination of license identifiers, license references and exceptions constructed using a small set of defined operators (e.g., AND, OR, WITH and +). We provide the definition of what constitutes a valid SPDX License Expression in this section.

The exact syntax of license expressions is described below in ABNF.

```
idstring = 1*(ALPHA /DIGIT /"-" /"." )

license-id = <short form license identifier in Annex A.1>

license-exception-id = <short form license exception identifier in Annex A.2>

license-ref = ["DocumentRef-"(idstring)":"]"LicenseRef-"(idstring)

addition-ref = ["DocumentRef-"(idstring)":"]"AdditionRef-"(idstring)

simple-expression = license-id /license-id"+" /license-ref

addition-expression = license-exception-id /addition-ref

compound-expression = (simple-expression /

  simple-expression "WITH" addition-expression /

  compound-expression "AND" compound-expression /

  compound-expression "OR" compound-expression /

  "(" compound-expression ")" )

license-expression = (simple-expression /compound-expression)
```

In the following sections we describe in more detail <license-expression> construct, a licensing expression string that enables a more accurate representation of the licensing terms of modern-day software.

A valid <license-expression> string consists of either:

(i) a simple license expression, such as a single license identifier; or

(ii) a more complex expression constructed by combining smaller valid expressions using Boolean license operators.

There MUST NOT be white space between a license-id and any following +. This supports easy parsing and backwards compatibility. There MUST be white space on either side of the operator "WITH". There MUST be white space and/or parentheses on either side of the operators AND and OR.

In the tag:value format, a license expression MUST be on a single line, and MUST NOT include a line break in the middle of the expression.

A.2 Case sensitivity

License expression operators (AND, OR and WITH) should be matched in a *case-sensitive* manner.

License identifiers (including license exception identifiers) used in SPDX documents or source code files should be matched in a *case-insensitive* manner. In other words, MIT, Mit and mIt should all be treated as the same identifier and referring to the same license.

However, please be aware that it is often important to match with the case of the canonical identifier on the SPDX License List. This is because the canonical identifier's case is used in the URL of the license's or exception's entry on the List, and because the canonical identifier is translated to a URI in RDF documents.

A.3 Simple license expressions

A simple <license-expression> is composed one of the following:

- An SPDX License List Short Form Identifier. For example: CDDL-1.0
- An SPDX License List Short Form Identifier with a unary "+" operator suffix to represent the current version of the license or any later version. For example: CDDL-1.0+
- An SPDX user defined license reference: ["DocumentRef-"1*(idstring)":"]"LicenseRef-"1*(idstring)

Some examples:

```
LicenseRef-23

LicenseRef-MIT-Style-1

DocumentRef-spdx-tool-1.2:LicenseRef-MIT-Style-2
```

The current set of valid license identifiers can be found in spdx.org/licenses.

A.4 Composite license expressions

A.4.1 Introduction

More expressive composite license expressions can be constructed using "OR", "AND", and "WITH" operators similar to constructing mathematical expressions using arithmetic operators.

For the tag:value format, any license expression that consists of more than one license identifier and/or LicenseRef, may optionally be encapsulated by parentheses: "( )".

Nested parentheses can also be used to specify an order of precedence which is discussed in more detail in D.4.5.

A.4.2 Disjunctive "OR" operator

If presented with a choice between two or more licenses, use the disjunctive binary "OR" operator to construct a new license expression, where both the left and right operands are valid license expression values.

For example, when given a choice between the LGPL-2.1-only or MIT licenses, a valid expression would be:

```
LGPL-2.1-only OR MIT
```

The "OR" operator is commutative, meaning that the above expression should be considered equivalent to:

```
MIT OR LGPL-2.1-only
```

An example representing a choice between three different licenses would be:

```
LGPL-2.1-only OR MIT OR BSD-3-Clause
```

A.4.3 Conjunctive "AND" operator

If required to simultaneously comply with two or more licenses, use the conjunctive binary "AND" operator to construct a new license expression, where both the left and right operands are a valid license expression values.

For example, when one is required to comply with both the LGPL-2.1-only or MIT licenses, a valid expression would be:

```
LGPL-2.1-only AND MIT
```

The "AND" operator is commutative, meaning that the above expression should be considered equivalent to:

```
MIT AND LGPL-2.1-only
```

An example where all three different licenses apply would be:

```
LGPL-2.1-only AND MIT AND BSD-2-Clause
```

A.4.4 Additive "WITH" operator

Sometimes license texts are found with additional text, which might or might not modify the original license terms.

In this case, use the binary "WITH" operator to construct a new license expression to represent the special situation. A valid `<license-expression>` is where the left operand is a `<simple-expression>` value and the right operand is a `<addition-expression>` that represents the additional text.

The `<addition-expression>` can be either a `<license-exception-id>` from the SPDX License List, or a user defined addition reference in the form ["DocumentRef-"(idstring)":"]"AdditonRef-"(idstring)

For example, when the Bison exception is to be applied to GPL-2.0-or-later, the expression would be:

```
GPL-2.0-or-later WITH Bison-exception-2.2
```

The current set of valid license exceptions identifiers can be found in spdx.org/licenses.

A.4.5 Order of precedence and parentheses

The order of application of the operators in an expression matters (similar to mathematical operators). The default operator order of precedence of a `<license-expression>` a is:

```
+
WITH
AND
OR
```

where a lower order operator is applied before a higher order operator.

For example, the following expression:

```
LGPL-2.1-only OR BSD-3-Clause AND MIT
```

represents a license choice between either LGPL-2.1-only and the expression BSD-3-Clause AND MIT because the AND operator takes precedence over (is applied before) the OR operator.

When required to express an order of precedence that is different from the default order a `<license-expression>` can be encapsulated in pairs of parentheses: ( ), to indicate that the operators found inside the parentheses takes precedence over operators outside. This is also similar to the use of parentheses in an algebraic expression e.g., (5+7)/2.

For instance, the following expression:

```
MIT AND (LGPL-2.1-or-later OR BSD-3-Clause)
```

states the OR operator should be applied before the AND operator. That is, one should first select between the LGPL-2.1-or-later or the BSD-3-Clause license before applying the MIT license.

A.4.6 License expressions in RDF

A conjunctive license can be expressed in RDF via a `<spdx:ConjunctiveLicenseSet>` element, with an spdx:member property for each element in the conjunctive license. Two or more members are required.

```
<spdx:ConjunctiveLicenseSet>
    <spdx:member  rdf:resource="http://spdx.org/licenses/GPL-2.0-only"/>
    <spdx:ExtractedLicensingInfo rdf:about
      ="http://example.org#LicenseRef-EternalSurrender">
        <spdx:extractedText>
            In exchange for using this software, you agree to give
            its author all your worldly possessions. You will not
            hold the author liable for all the damage this software
            will inevitably cause not only to your person and
            property, but to the entire fabric of the cosmos.
        </spdx:extractedText>
        <spdx:licenseId>LicenseRef-EternalSurrender</spdx:licenseId>
    </spdx:ExtractedLicensingInfo>
</spdx:ConjunctiveLicenseSet>
```

A disjunctive license can be expressed in RDF via a `<spdx:DisjunctiveLicenseSet>` element, with an

spdx:member property for each element in the disjunctive license. Two or more members are required.

```
<spdx:DisjunctiveLicenseSet>
    <spdx:member  rdf:resource="http://spdx.org/licenses/GPL-2.0-only"/>

    <spdx:member>
        <spdx:ExtractedLicensingInfo rdf:about
          ="http://example.org#LicenseRef-EternalSurrender">
            <spdx:extractedText>
                In exchange for using this software, you agree to
                give its author all your worldly possessions. You
                will not hold the author liable for all the damage
                this software will inevitably cause not only to
                your person and property, but to the entire fabric
                of the cosmos.
            </spdx:extractedText>
            <spdx:licenseId>LicenseRef-EternalSurrender</spdx:licenseId>
        </spdx:ExtractedLicensingInfo>
    </spdx:member>
</spdx:DisjunctiveLicenseSet>
```

A License Exception can be expressed in RDF via a `<spdx:LicenseException>` element. This element has the following unique mandatory (unless specified otherwise) attributes:

- `comment` - An `rdfs:comment` element describing the nature of the exception.
- `seeAlso` (optional, one or more)- An `rdfs:seeAlso` element referencing external sources of information on the exception.
- `example` (optional) - Text describing examples of this exception.
- `name` - The full human readable name of the item.
- `licenseExceptionId` - The identifier of an exception in the SPDX License List to which the exception applies.
- `licenseExceptionText` - Full text of the license exception.

```
<rdf:Description rdf:about
  ="http://example.org#SPDXRef-ButIdDontWantToException">
    <rdfs:comment>This exception may be invalid in some
      jurisdictions.</rdfs:comment>
    <rdfs:seeAlso>http://dilbert.com/strip/1997-01-15</rdfs:seeAlso>
    <spdx:example>So this one time, I had a license exception
      …</spdx:example>
    <spdx:licenseExceptionText>
        A user of this software may decline to follow any subset of
        the terms of this license upon finding any or all such terms
        unfavorable.
    </spdx:licenseExceptionText>
    <spdx:name>&amp;quot;But I Don&amp;apos;t Want To&amp;quot; Exception</spdx:name>
    <spdx:licenseExceptionId>SPDXRef-
ButIdDontWantToException</spdx:licenseExceptionId>
    <rdf:type rdf:resource
      ="http://spdx.org/rdf/terms#LicenseException"/>
</rdf:Description>
```

Annex B: Using SPDX license list short identifiers in source files

(Informative)

B.1 Introduction

Identifying the license for open source software is critical for both reporting purposes and license compliance. However, determining the license can sometimes be difficult due to a lack of information or ambiguous information. Even when licensing information is present, a lack of consistent notation can make automating the task of license detection very difficult, thus requiring vast amounts of human effort.

Short identifiers from the SPDX License List can be used to indicate license info at the file level. The advantages of doing this are numerous but include:

- It is precise.
- It is concise.
- It is language neutral.
- It is easy and more reliable to machine process.
- Leads to code that is easier to reuse.
- The license information travels with the file (as sometimes not entire projects are used or license files are removed).
- It is a standard and can be universal. There is no need for variation.
- An SPDX short identifier is immutable.
- Easy look-ups and cross-references to the SPDX License List website.

If using SPDX short identifiers in individual files, it is recommended to reproduce the full license in the projects LICENSE file and indicate that SPDX short identifiers are being used to refer to it. For links to projects illustrating these scenarios, see https://spdx.dev/ids-where.

B.2 Format for SPDX-License-Identifier

The SPDX-License-Identifier tag declares the license the file is under and should be placed at or near the top of the file in a comment.

The SPDX License Identifier syntax may consist of a single license (represented by a short identifier from the SPDX license list) or a compound set of licenses (represented by joining together multiple licenses using the license expression syntax).

The tag should appear on its own line in the source file, generally as part of a comment.

```
SPDX-License-Identifier: <SPDX License Expression>
```

B.3 Representing single license

A single license is represented by using the short identifier from SPDX license list, optionally with a unary "+" operator following it to indicate "or later" versions may be applicable.

Examples:

```
SPDX-License-Identifier: CDDL-1.0+
SPDX-License-Identifier: MIT
```

D.4 Representing multiple licenses

Multiple licenses can be represented using an SPDX license expression as defined in Annex D. A set of licenses may optionally be enclosed in parentheses, but are not required to be enclosed. As further described there:

1. When there is a choice between licenses ("disjunctive license"), they should be separated with "OR". If presented with a choice between two or more licenses, use the disjunctive binary "OR" operator to construct a new license expression.
2. Similarly when multiple licenses need to be simultaneously applied ("conjunctive license"), they should be separated with "AND". If required to simultaneously comply with two or more licenses, use the conjunctive binary "AND" operator to construct a new license expression.
3. In some cases, a set of license terms apply except under special circumstances, in this case, use the "WITH" operator followed by one of the recognized exception identifiers.
4. The expression MUST be on a single line, and MUST NOT include a line break in the middle of the expression.

Examples:

```
SPDX-License-Identifier: GPL-2.0-only OR MIT
SPDX-License-Identifier: LGPL-2.1-only AND BSD-2-Clause
SPDX-License-Identifier: GPL-2.0-or-later WITH Bison-exception-2.2
```

Please see Annex D for more examples and details of the license expression specific syntax.

If you can't express the license(s) as an expression using identifiers from the SPDX list, it is probably best to just put the text of your license header in the file (if there is a standard header), or refer to a neutral site URL where the text can be found. To request a license be added to the SPDX License List, please follow the process described here: https://github.com/spdx/license-list-XML/blob/master/CONTRIBUTING.md.

Alternatively, you can use a `LicenseRef-` custom license identifier to refer to a license that is not on the SPDX License List, such as the following:

```
SPDX-License-Identifier: LicenseRef-my-special-license
```

The `LicenseRef-` format is defined in Annex D. When using a custom `LicenseRef-` identifier, you will also need to provide a way for others to determine what license text corresponds to it. Version 3.0 of the REUSE Software Specification provides a standardized format that can optionally be used for providing the corresponding license text for these identifiers.