

Standard ECMA-427

1st Edition / December 2025

Package-URL (PURL) specification

Standard



COPYRIGHT PROTECTED DOCUMENT

Contents		Page
1	Scope	1
2	Conformance	1
3	Normative references	2
4	Overview	2
5	Package-URL specification	2
5.1	A PURL is a URL	3
5.2	Permitted characters	4
5.3	Separator characters	4
5.4	Character encoding	4
5.5	Case folding	4
5.6	Rules for each PURL component	5
5.6.1	Scheme	5
5.6.2	Type	5
5.6.3	Namespace	5
5.6.4	Name	5
5.6.5	Version	5
5.6.6	Qualifiers	6
5.6.7	Subpath	6
6	Package-URL Type Definition Schema	6
6.1	PURL type	7
6.2	Type name	8
6.3	Description	8
6.4	Repository	8
6.4.1	Use repository	8
6.4.2	Default repository URL	8
6.4.3	Note	9
6.5	Namespace definition	9
6.5.1	Namespace requirement	9
6.5.2	Component optional requirement	10
6.5.3	Component required requirement	10
6.5.4	Component prohibited requirement	10
6.5.5	Permitted characters in this PURL component	10
6.5.6	Case sensitive	10
6.5.7	Normalization rules	10
6.5.8	Native name	11
6.5.9	Note	11
6.6	Name definition	11
6.6.1	Name component requirement	12
6.6.2	Component required requirement	12
6.6.3	Permitted characters in this PURL component	12
6.6.4	Case sensitive	12
6.6.5	Normalization rules	12
6.6.6	Native name	12
6.6.7	Note	13
6.7	Version definition	13
6.7.1	Version requirement	13
6.7.2	Component optional requirement	13
6.7.3	Permitted characters in this PURL component	14
6.7.4	Case sensitive	14
6.7.5	Normalization rules	14
6.7.6	Native name	14
6.7.7	Note	14
6.8	Qualifiers definition	14
6.8.1	Qualifiers definition	15
6.9	Subpath definition	16
6.9.1	Subpath requirement	17

6.9.2	Component optional requirement	17
6.9.3	Permitted characters in this PURL component	17
6.9.4	Case sensitive	17
6.9.5	Normalization rules	17
6.9.6	Native name	17
6.9.7	Note	18
6.10	PURL examples	18
6.11	Note	18
6.12	Reference URLs	18
Annex A (normative)	PURL Type Definition	19
Bibliography		27
Colophon		29
Software License		31

Introduction

Software ecosystems have evolved into highly interconnected networks of components, packages, and dependencies. Managing this complexity demands a robust, uniform mechanism to identify and track software packages across diverse ecosystems and tools. Package-URL (PURL) was developed to address this challenge by providing a simple, consistent, and flexible approach to identifying software packages with precision and clarity.

PURL introduces a standardized URL-based syntax that uniquely identifies software packages, independent of their ecosystem or distribution channel. Unlike traditional identification methods, PURL embeds critical metadata directly into its structure, enabling efficient, accurate package identification at scale. This standardization ensures interoperability between tools and ecosystems, fostering greater collaboration and reducing ambiguity in software supply chain management.

Challenges addressed by PURL:

- **Ambiguity in Package Identification:** With diverse naming conventions across ecosystems, identifying software packages reliably has historically been a challenge. PURL eliminates this ambiguity by creating a universal identifier with a predictable structure.
- **Cross-Ecosystem Interoperability:** Developers, organizations, and tools often work across multiple ecosystems, each with its own package management systems. PURL harmonizes these differences, enabling seamless interoperability.
- **Enhanced Traceability and Risk Management:** In an era where supply chain security is critical, PURL provides the foundation for identifying and tracing packages to their origins, dependencies, and potential vulnerabilities.
- **Tooling and Automation:** By standardizing package identification, PURL simplifies tooling development, automation, and integration for tasks such as software composition analysis, vulnerability management, and license compliance.

As software supply chain security becomes a global priority, formalizing PURL as an international standard ensures its adoption and consistent implementation. Standardization under Ecma International Technical Committee 54 (TC54) positions PURL as a foundational building block for secure, transparent, and efficient software ecosystems worldwide.

By enabling a universally recognized and implementable specification, PURL aligns with global efforts to improve the security, reliability, and accountability of software supply chains. Its adoption ensures that organizations and developers can rely on a common language to manage software packages across the diverse and rapidly evolving software landscape.

This Ecma Standard was developed by Technical Committee 54 and was adopted by the General Assembly of December 2025.

About this specification

The document at <https://tc54.org/purl/> is the most accurate and up-to-date Package-URL specification.

This document is available as [a single page](#) and as [multiple pages](#).

Contributing to this specification

This specification is developed on GitHub with the help of the Package-URL community. There are a number of ways to contribute to the development of this specification:

GitHub Repository: <https://github.com/Ecma-TC54/ECMA-427>

Issues: [All Issues](#), [File a New Issue](#)

Pull Requests: [All Pull Requests](#), [Create a New Pull Request](#)

Editors:

- [John Horan](#)
- [Michael Herzog](#)
- [Philippe Ombredanne](#)
- [Steve Springett](#)

Community:

- Chat: [Slack Channel](#)

Refer to the [colophon](#) for more information on how this document was created.

COPYRIGHT NOTICE

© 2025 Ecma International

By obtaining and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

This document may be copied, published and distributed to others, and certain derivative works of it may be prepared, copied, published, and distributed, in whole or in part, provided that the above copyright notice and this Copyright License and Disclaimer are included on all such copies and derivative works. The only derivative works that are permissible under this Copyright License and Disclaimer are:

- (i) works which incorporate all or portion of this document for the purpose of providing commentary or explanation (such as an annotated version of the document),*
- (ii) works which incorporate all or portion of this document for the purpose of incorporating features that provide accessibility,*
- (iii) translations of this document into languages other than English and into different formats and*
- (iv) works by making use of this specification in standard conformant products by implementing (e.g. by copy and paste wholly or partly) the functionality therein.*

However, the content of this document itself may not be modified in any way, including by removing the copyright notice or references to Ecma International, except as required to translate it into languages other than English or into a different format.

The official version of an Ecma International document is the English language version on the Ecma International website. In the event of discrepancies between a translated version and the official version, the official version shall govern.

The limited permissions granted above are perpetual and will not be revoked by Ecma International or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and ECMA INTERNATIONAL DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.



Package-URL (PURL) specification

1 Scope

This Standard defines the Package-URL (PURL) syntax for identifying software packages independently from their ecosystem or distribution channel. PURL is used to identify software packages across software supply chains supporting many use cases, identifying software packages in Software Bills of Materials, vulnerability databases, vulnerability advisories, vulnerability disclosures and exploitability reports, and managing software package dependencies.

A PURL is a valid URL and URI composed of seven components to identify a software package. The PURL type component defines the ecosystem-specific structure and meaning for the other PURL components. This Standard specifies the syntax for PURLs and the schema for defining PURL types, but it does not include any specific PURL type definitions, such as maven, pypi or npm.

2 Conformance

A conforming implementation of Package-URL (PURL) shall fully implement and support all elements defined within this Standard, including the syntax, components, and semantic requirements for constructing and interpreting valid PURLs.

A conforming implementation of PURL shall adhere to the syntax defined in this Standard, ensuring that all PURLs are parsed, constructed, and validated according to the prescribed rules. The implementation shall provide full support for ecosystem-agnostic behaviour, enabling PURLs to function consistently and reliably across diverse environments.

All required components of a PURL, such as the scheme, type, and name, shall be present and validated according to the rules defined in this Standard. Additionally, optional components, including qualifiers and subpaths, shall be handled appropriately if provided, in full compliance with their specified behaviours.

Implementations shall ensure that equivalent PURLs are consistently resolved to the same canonical representation. This includes strict adherence to normalisation and equivalence rules. Furthermore, implementations shall process URI encoding and decoding for PURL components according to the standards outlined in RFC 3986.

Invalid PURLs that fail to conform to the specification shall be identified and rejected by any conforming implementation. This guarantees the integrity and reliability of PURLs in all supported contexts.

A conforming implementation of PURL may extend its functionality by providing ecosystem-specific validation, processing, or metadata handling, as long as these extensions do not violate the core specification. Additionally, implementations may offer auxiliary tools or features, such as utilities for constructing or validating PURLs, provided they align with the standard's requirements.

A conforming implementation shall not redefine or alter the core syntax, components, or semantics defined by this Standard. Any prohibited extensions explicitly identified in the specification shall not be implemented. Furthermore, behaviours that compromise the interoperability of PURLs across tools, platforms, or ecosystems are strictly disallowed.

3 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ECMA-262, *ECMAScript® language specification*

<https://ecma-international.org/publications-and-standards/standards/ecma-262/>

RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*

<https://datatracker.ietf.org/doc/html/rfc3986>

The Unicode Standard

<https://www.unicode.org/versions/latest/>

4 Overview

This Clause contains a non-normative overview of the Package-URL specification.

The Package-URL (PURL) specification defines a lightweight, universal syntax for identifying software packages. By leveraging a URL-based format, PURL provides a consistent and interoperable mechanism for referencing software packages across a wide range of ecosystems and tools. Its design addresses the challenges of ambiguity, inconsistency, and fragmentation in software package identification, enabling better interoperability and traceability in modern software supply chains.

This Standard focuses on the core aspects of PURL, including its syntax, required components, optional attributes, and conformance requirements. It does not cover ecosystem-specific types or extensions such as PURL Version Ranges (VERS). However, the flexibility of PURL allows it to be extended to meet the needs of diverse package ecosystems without compromising its universal applicability.

The primary audience for this Standard includes developers, tool implementers, and organisations involved in software composition analysis, dependency management, and supply chain security. PURL is foundational to a variety of use cases, from software bill of materials (SBOM) generation and license compliance to vulnerability tracking and software artefact exchange.

While this document serves as the authoritative reference for implementing PURL, it is complemented by various ecosystem-specific guidance documents, examples, and related standards. These resources provide additional context and practical insights for leveraging PURL effectively.

This overview is non-normative and serves to provide context for the specification's intent, purpose, and audience. For detailed requirements and conformance criteria, refer to the normative clauses of this Standard.

5 Package-URL specification

PURL stands for **Package-URL**.

A PURL is a URL composed of seven components:

```
scheme:type/namespace/name@version?qualifiers#subpath
```

Components are separated by a specific character for unambiguous parsing.

Table 1 — Components of a PURL

Component	Requirement	Description
scheme	Required	The URL scheme with the constant value of "pkg". One of the primary reasons for this single scheme is to facilitate the future official registration of the "pkg" scheme for Package-URLs.
type	Required	The package "type" or package "protocol" such as maven, npm, nuget, gem, pypi, etc.
namespace	Optional	A name prefix such as a Maven groupid, a Docker image owner, a GitHub user or organization. Namespace is type-specific.
name	Required	The name of the package.
version	Optional	The version of the package.
qualifiers	Optional	Qualifier data for a package such as OS, architecture, repository, etc. Qualifiers are type-specific.
subpath	Optional	Subpath within a package, relative to the package root.

Components are designed such that they form a hierarchy from the most significant on the left to the least significant components on the right.

A PURL shall not contain a URL Authority, i.e. there is no support for **username**, **password**, **host** and **port** components. A **namespace** segment may sometimes look like a **host**, but its interpretation is specific to a **type**.

Example 1 (Informative): Debian

pkg:deb/debian/curl@7.50.3-1?arch=i386&distro=jessie

Example 2 (Informative): Maven

pkg:maven/org.apache.xmlgraphics/batik-anim@1.9.1?packaging=sources

Example 3 (Informative): Node Package Manager (NPM)

pkg:npm/foobar@12.3.1

5.1 A PURL is a URL

- A PURL is a valid URL and URI that conforms to the URL definitions or specifications at:
 - <https://tools.ietf.org/html/rfc3986>
 - <https://en.wikipedia.org/wiki/URL#Syntax>
 - https://en.wikipedia.org/wiki/Uniform_Resource_Identifier#Syntax
 - <https://url.spec.whatwg.org/>
- A PURL is a valid URL because it is a locator even though it has no Authority URL component: each type has a default repository location when defined.
- The PURL components are mapped to these URL components:
 - PURL **scheme**: this is a URL **scheme** with a constant value: **pkg**
 - PURL **type**, **namespace**, **name** and **version** components: these are collectively mapped to a URL **path**
 - PURL **qualifiers**: this maps to a URL **query**
 - PURL **subpath**: this is a URL **fragment**

In a PURL, there is no support for a URL Authority (e.g. no **username**, **password**, **host** and **port** components).

- Special URL schemes as defined in <https://url.spec.whatwg.org/> such as **file://**, **https://**, **http://** and **ftp://** are not valid PURL types. They are valid URL or URI schemes but they are not a valid PURL scheme. They may be used to reference URLs in separate attributes outside of a PURL or in a PURL

qualifier.

- Version control system (VCS) URLs such **git://**, **svn://**, **hg://** or as defined in Python pip or SPDX download locations are not valid PURL types. They are valid URL or URI schemes but they are not a valid PURL scheme. They are a closely related, compact and uniform way to reference VCS URLs. They may be used as references in separate attributes outside of a PURL or in a PURL qualifier.

5.2 Permitted characters

A canonical PURL is composed of these permitted ASCII characters:

- the Alphanumeric Characters: **A to Z**, **a to z**, **0 to 9**
- the Punctuation Characters: **.** **-** **_** **~** (period '.', dash '-', underscore '_' and tilde '~')
- the Percent Character: **%** (percent sign '%')
- the Separator Characters: **:** **/** **@** **?** **=** **&** **#** (colon ':', slash '/', at sign '@', question mark '?', equal sign '=', ampersand '&' and hash sign '#')

5.3 Separator characters

This is how each of the Separator Characters is used:

- ':' (colon) is the separator between **scheme** and **type**
- '/' (slash) is the separator between **type**, **namespace** and **name**
- '/' (slash) is the separator between **subpath** segments
- '@' (at sign) is the separator between **name** and **version**
- '?' (question mark) is the separator before **qualifiers**
- '=' (equals) is the separator between a **key** and a **value** of a **qualifier**
- '&' (ampersand) is the separator between **qualifiers** (each being a **key=value** pair)
- '#' (hash sign) is the separator before **subpath**

5.4 Character encoding

- In the [Rules for each PURL component](#) clause, each component defines when and how to apply percent-encoding and decoding to its content.
- When percent-encoding is required by a component definition, the component string shall first be encoded as UTF-8.
- In the component string, each "data octet" shall be replaced by the percent-encoded "character triplet" applying the percent-encoding mechanism defined in [RFC 3986 section 2.1](#), including the RFC definition of "data octet" and "character triplet", and using these definitions for RFC's "allowed set" and "delimiters":
 - "allowed set" is composed of the Alphanumeric Characters and the Punctuation Characters
 - "delimiters" is composed of the Separator Characters
- The following characters shall not be percent-encoded:
 - the Alphanumeric Characters
 - the Punctuation Characters
 - the Separator Characters when being used as PURL separators
 - the colon ':', whether used as a Separator Character or otherwise
 - the percent sign '%' when used to represent a percent-encoded character
- Where the space ' ' is permitted, it shall be percent-encoded as '%20'.
- With the exception of the percent-encoding mechanism, the rules regarding percent-encoding are defined by this Standard alone.

5.5 Case folding

References to "lowercase" in this Standard refer to the **culture-invariant** full case mapping defined in [Section 3.13.2 of the Unicode Standard](#).

When applied to the ASCII character set, this operation converts uppercase Latin letters (**A to Z**) to their corresponding lowercase forms (**a to z**). All other ASCII characters remain unchanged.

5.6 Rules for each PURL component

A PURL string is an ASCII URL string composed of seven components. Except as expressly stated otherwise in this Clause, each component:

- may be composed of any of the characters defined in the [Permitted characters](#) clause
- shall be encoded as defined in the [Character encoding](#) clause

The "lowercase" rules are defined in the [Case folding](#) clause.

The rules for each component are:

5.6.1 Scheme

- The **scheme** is a constant with the value "pkg".
- The **scheme** shall be followed by an unencoded colon ':'.
- PURL parsers shall accept URLs where the **scheme** and colon ':' are followed by one or more slash '/' characters, such as 'pkg://', and shall ignore and remove all such '/' characters.

5.6.2 Type

- The package **type** shall be composed only of ASCII letters and numbers, period '.', and dash '-'.
- The **type** shall start with an ASCII letter.
- The **type** shall not be percent-encoded.
- The **type** is case insensitive. The canonical form is lowercase.

5.6.3 Namespace

- The **namespace** is optional, unless required by the package's **type** definition.
- If present, the **namespace** may contain one or more segments, separated by a single unencoded slash '/' character.
- All leading and trailing slashes '/' are not significant and should be stripped in the canonical form. They are not part of the **namespace**.
- Each **namespace** segment shall be a percent-encoded string.
- When percent-decoded, a segment:
 - shall not contain any slash '/' characters
 - shall not be empty
 - may contain any Unicode character other than '/' unless the package's **type** definition provides otherwise
- A URL host or Authority shall not be used as a **namespace**. Use instead a **repository_url** qualifier. Note however, that for some types, the **namespace** may look like a host.

5.6.4 Name

- The **name** is prefixed by a single slash '/' separator when the **namespace** is not empty.
- All leading and trailing slashes '/' are not significant and should be stripped in the canonical form. They are not part of the **name**.
- A **name** shall be a percent-encoded string.
- When percent-decoded, a **name** may contain any Unicode character unless the package's **type** definition provides otherwise.

5.6.5 Version

- The **version** is prefixed by a '@' separator when not empty.
- This '@' is not part of the **version**.
- A **version** shall be a percent-encoded string.
- When percent-decoded, a **version** may contain any Unicode character unless the package's **type**

definition provides otherwise.

- A **version** is a plain and opaque string.

5.6.6 Qualifiers

- The **qualifiers** component shall be prefixed by an unencoded question mark '?' separator when not empty. This '?' separator is not part of the **qualifiers** component.
- The **qualifiers** component is composed of one or more **key=value** pairs. Multiple **key=value** pairs shall be separated by an unencoded ampersand '&'. This '&' separator is not part of an individual **qualifier**.
- A **key** and **value** shall be separated by the unencoded equal sign '=' character. This '=' separator is not part of the **key** or **value**.
- A **value** shall not be an empty string: a **key=value** pair with an empty **value** is the same as if no **key=value** pair exists for this **key**.
- For each **key=value** pair:
 - The **key** shall be composed only of lowercase ASCII letters and numbers, period '.', dash '-' and underscore '_'.
 - A **key** shall start with an ASCII letter.
 - A **key** shall not be percent-encoded.
 - Each **key** shall be unique among all the keys of the **qualifiers** component.
 - A **value** may contain any Unicode character and all characters shall be encoded as described in the [Character encoding](#) clause.

5.6.7 Subpath

- The **subpath** string is prefixed by a '#' separator when not empty.
- The '#' is not part of the **subpath**.
- The **subpath** contains zero or more segments, separated by slash '/'.
- Leading and trailing slashes '/' are not significant and should be stripped in the canonical form.
- Each **subpath** segment shall be a percent-encoded string.
- When percent-decoded, a segment:
 - shall not contain any slash '/' characters
 - shall not be empty
 - shall not be any of '..' or ''
 - may contain any Unicode character other than '/' unless the package's **type** definition provides otherwise
- The **subpath** shall be interpreted as relative to the root of the package.

6 Package-URL Type Definition Schema

The PURL Type Definition JSON Schema is the reference data model that is used to define PURL types in a structured way. Each PURL type is specified in a JSON document that matches this schema. These JSON documents are then used to generate PURL type documentation and to support PURL libraries and tools so that they can more easily parse, build, and validate PURLs by type in a consistent and standardized manner across programming languages and technology stacks.

Location: /

Type: Object

Schema to specify a Package-URL (PURL) type as a structured definition.

Table 2 — Properties for the root object

Property	Type	Requirement	Description
type	String	Required	The type string for this Package-URL type.
type_name	String	Required	The name for this PURL type.
description	String	Required	The description of this PURL type.
repository	Object	Required	The package repository usage for this PURL type.
namespace_definition	Array	Required	Definition of the namespace component for this PURL type. The PURL namespace component shall be required, optional or prohibited for a specific PURL type definition.
name_definition	Array	Required	Definition of the name component for this PURL type. The PURL name component is required for all PURL type definitions.
version_definition	Array	Optional	Definition of the version component for this PURL type. The PURL version component is optional for a specific PURL type definition.
qualifiers_definition	Array	Optional	Definition of the qualifiers specific to this PURL type. The PURL qualifiers component is optional for a specific PURL type, but a qualifiers key or keys may be required for a specific PURL type.
subpath_definition	Array	Optional	The definition for the subpath for this PURL type. The PURL subpath component is optional for a specific PURL type definition.
examples	Array	Required	Example of valid, canonical PURLs for this package type.
note	String	Optional	Note about this PURL type.
reference_urls	Array	Optional	Optional list of informational reference URLs about this PURL type.

6.1 PURL type

Location: /type

Property: type (Required)

Type: String

Pattern Constraint: `^[a-z][a-z0-9-\.]+$`

The type string for this Package-URL type.

Example 1 (Informative)

maven

Example 2 (Informative)

npm

Example 3 (Informative)

pypi

6.2 Type name

Location: /type_name
Property: type_name (Required)
Type: String

The name for this PURL type.

Example 1 (Informative)
 Apache Maven

Example 2 (Informative)
 Python Package

6.3 Description

Location: /description
Property: description (Required)
Type: String

The description of this PURL type.

6.4 Repository

Location: /repository
Property: repository (Required)
Type: Object

The package repository usage for this PURL type.

Table 3 — Properties for the repository object

Property	Type	Requirement	Description
use_repository	Boolean	Required	true if this PURL type uses a public package repository.
default_repository_url	String	Optional	The default public repository URL for this PURL type.
note	String	Optional	Extra note text.

6.4.1 Use repository

Location: /repository/use_repository
Property: use_repository (Required)
Type: Boolean

true if this PURL type uses a public package repository.

6.4.2 Default repository URL

Location: /repository/default_repository_url
Property: default_repository_url (Optional)
Type: String
Format: URI as specified in [RFC 3986](#)

The default public repository URL for this PURL type.

6.4.3 Note

Location: /repository/note

Property: note (Optional)

Type: String

Extra note text.

6.5 Namespace definition

Location: /namespace_definition

Property: namespace_definition (Required)

Type: Object

Definition of the namespace component for this PURL type. The PURL namespace component shall be required, optional or prohibited for a specific PURL type definition.

Table 4 — Properties for the namespace_definition object

Property	Type	Requirement	Description
requirement	Array	Required	States that the PURL namespace component is optional, required or prohibited for a PURL type.
permitted_characters	String	Optional	A regular expression (ECMA-262 dialect) defining the 'Permitted characters' for this component of this Package-URL type. If provided, this shall be a subset of the 'Permitted characters' defined in the PURL specification.
case_sensitive	Boolean	Optional	true if this PURL component is case sensitive. If false , the canonical form shall be lowercased.
normalization_rules	Array	Optional	List of rules to normalize this component for this PURL type. These are plain text, unstructured rules as some require programming and cannot be enforced only with a schema. Tools are expected to apply these rules programmatically.
native_name	String	Optional	The native name of this PURL component in the package ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupId', and 'scope' for the 'npm' PURL type.
note	String	Optional	Extra note text.

6.5.1 Namespace requirement

Location: /namespace_definition/requirement

Property: requirement (Required)

Type: String

States that the PURL namespace component is optional, required or prohibited for a PURL type.

Shall be one of:

1. Component optional requirement
2. Component required requirement
3. Component prohibited requirement

6.5.2 Component optional requirement

Type: String

Constant: optional

States that this PURL component is optional for a PURL type.

6.5.3 Component required requirement

Type: String

Constant: required

States that this PURL component is required for a PURL type.

6.5.4 Component prohibited requirement

Type: String

Constant: prohibited

States that this PURL component is prohibited for a PURL type.

6.5.5 Permitted characters in this PURL component

Location: /namespace_definition/permitted_characters

Property: permitted_characters (Optional)

Type: String

Format: A regular expression dialect defined by [ECMA-262](#)

A regular expression ([ECMA-262](#) dialect) defining the 'Permitted characters' for this component of this Package-URL type. If provided, this shall be a subset of the 'Permitted characters' defined in the PURL specification.

6.5.6 Case sensitive

Location: /namespace_definition/case_sensitive

Property: case_sensitive (Optional)

Type: Boolean

Default Value: `true`

`true` if this PURL component is case sensitive. If `false`, the canonical form shall be lowercased.

6.5.7 Normalization rules

Location: /namespace_definition/normalization_rules

Property: normalization_rules (Optional)

Type: array (of String)

List of rules to normalize this component for this PURL type. These are plain text, unstructured rules as some require programming and cannot be enforced only with a schema. Tools are expected to apply these rules programmatically. Each item of this array shall be a string.

All items shall be unique.

6.5.8 Native name

Location: /namespace_definition/native_name

Property: native_name (Optional)

Type: String

The native name of this PURL component in the package ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupId', and 'scope' for the 'npm' PURL type.

6.5.9 Note

Location: /namespace_definition/note

Property: note (Optional)

Type: String

Extra note text.

6.6 Name definition

Location: /name_definition

Property: name_definition (Required)

Type: Object

Definition of the name component for this PURL type. The PURL name component is required for all PURL type definitions.

Table 5 — Properties for the name_definition object

Property	Type	Requirement	Description
requirement	Array	Required	States that the PURL name component is always required.
permitted_characters	String	Optional	A regular expression (ECMA-262 dialect) defining the 'Permitted characters' for this component of this Package-URL type. If provided, this shall be a subset of the 'Permitted characters' defined in the PURL specification.
case_sensitive	Boolean	Optional	true if this PURL component is case sensitive. If false , the canonical form shall be lowercased.
normalization_rules	Array	Optional	List of rules to normalize this component for this PURL type. These are plain text, unstructured rules as some require programming and cannot be enforced only with a schema. Tools are expected to apply these rules programmatically.
native_name	String	Optional	The native name of this PURL component in the package ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupId', and 'scope' for the 'npm' PURL type.
note	String	Optional	Extra note text.

6.6.1 Name component requirement

Location: /name_definition/requirement

Property: requirement (Required)

Type: String

States that the PURL name component is always required.

Shall be one of:

1. Component required requirement

6.6.2 Component required requirement

Type: String

Constant: required

States that this PURL component is required for a PURL type.

6.6.3 Permitted characters in this PURL component

Location: /name_definition/permitted_characters

Property: permitted_characters (Optional)

Type: String

Format: A regular expression dialect defined by [ECMA-262](#)

A regular expression ([ECMA-262](#) dialect) defining the 'Permitted characters' for this component of this Package-URL type. If provided, this shall be a subset of the 'Permitted characters' defined in the PURL specification.

6.6.4 Case sensitive

Location: /name_definition/case_sensitive

Property: case_sensitive (Optional)

Type: Boolean

Default Value: **true**

true if this PURL component is case sensitive. If **false**, the canonical form shall be lowercased.

6.6.5 Normalization rules

Location: /name_definition/normalization_rules

Property: normalization_rules (Optional)

Type: array (of String)

List of rules to normalize this component for this PURL type. These are plain text, unstructured rules as some require programming and cannot be enforced only with a schema. Tools are expected to apply these rules programmatically. Each item of this array shall be a string.

All items shall be unique.

6.6.6 Native name

Location: /name_definition/native_name

Property: native_name (Optional)

Type: String

The native name of this PURL component in the package ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupId', and 'scope' for the 'npm' PURL type.

6.6.7 Note

Location: /name_definition/note

Property: note (Optional)

Type: String

Extra note text.

6.7 Version definition

Location: /version_definition

Property: version_definition (Optional)

Type: Object

Definition of the version component for this PURL type. The PURL version component is optional for a specific PURL type definition.

Table 6 — Properties for the version_definition object

Property	Type	Requirement	Description
requirement	Array	Required	States that the PURL version is optional.
permitted_characters	String	Optional	A regular expression (ECMA-262 dialect) defining the 'Permitted characters' for this component of this Package-URL type. If provided, this shall be a subset of the 'Permitted characters' defined in the PURL specification.
case_sensitive	Boolean	Optional	true if this PURL component is case sensitive. If false , the canonical form shall be lowercased.
normalization_rules	Array	Optional	List of rules to normalize this component for this PURL type. These are plain text, unstructured rules as some require programming and cannot be enforced only with a schema. Tools are expected to apply these rules programmatically.
native_name	String	Optional	The native name of this PURL component in the package ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupId', and 'scope' for the 'npm' PURL type.
note	String	Optional	Extra note text.

6.7.1 Version requirement

Location: /version_definition/requirement

Property: requirement (Required)

Type: String

States that the PURL version is optional.

Shall be one of:

1. Component optional requirement

6.7.2 Component optional requirement

Type: String

Constant: optional

States that this PURL component is optional for a PURL type.

6.7.3 Permitted characters in this PURL component

Location: /version_definition/permitted_characters

Property: permitted_characters (Optional)

Type: String

Format: A regular expression dialect defined by [ECMA-262](#)

A regular expression ([ECMA-262](#) dialect) defining the 'Permitted characters' for this component of this Package-URL type. If provided, this shall be a subset of the 'Permitted characters' defined in the PURL specification.

6.7.4 Case sensitive

Location: /version_definition/case_sensitive

Property: case_sensitive (Optional)

Type: Boolean

Default Value: `true`

`true` if this PURL component is case sensitive. If `false`, the canonical form shall be lowercased.

6.7.5 Normalization rules

Location: /version_definition/normalization_rules

Property: normalization_rules (Optional)

Type: array (of String)

List of rules to normalize this component for this PURL type. These are plain text, unstructured rules as some require programming and cannot be enforced only with a schema. Tools are expected to apply these rules programmatically. Each item of this array shall be a string.

All items shall be unique.

6.7.6 Native name

Location: /version_definition/native_name

Property: native_name (Optional)

Type: String

The native name of this PURL component in the package ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupId', and 'scope' for the 'npm' PURL type.

6.7.7 Note

Location: /version_definition/note

Property: note (Optional)

Type: String

Extra note text.

6.8 Qualifiers definition

Location: /qualifiers_definition

Property: qualifiers_definition (Optional)

Type: Array

Definition of the qualifiers specific to this PURL type. The PURL qualifiers component is optional for a specific PURL type, but a qualifiers key or keys may be required for a specific PURL type. Each item of this array shall be a Qualifiers definition object.

6.8.1 Qualifiers definition

Location: /qualifiers_definition/[]

Type: Object

The definition of a qualifier specific to this PURL type.

Table 7 — Properties for the qualifiers_definition object

Property	Type	Requirement	Description
key	String	Required	The key for the qualifier.
requirement	Array	Optional	States that a PURL qualifier key is optional or required for a PURL type.
description	String	Required	The description of this qualifier.
default_value	String	Optional	The optional default value of this qualifier if not provided.
native_name	String	Optional	The equivalent native name for this qualifier key.

6.8.1.1 Qualifier key

Location: /qualifiers_definition/[]/key

Type: String

The key for the qualifier.

6.8.1.2 Qualifier key requirement

Location: /qualifiers_definition/[]/requirement

Type: String

States that a PURL qualifier key is optional or required for a PURL type.

Shall be one of:

1. Component optional requirement
2. Component required requirement

6.8.1.3 Component optional requirement

Type: String

Constant: optional

States that this PURL component is optional for a PURL type.

6.8.1.4 Component required requirement

Type: String

Constant: required

States that this PURL component is required for a PURL type.

6.8.1.5 Description

Location: /qualifiers_definition/[]/description

Type: String

The description of this qualifier.

6.8.1.6 Default value

Location: /qualifiers_definition/[]/default_value

Type: String

The optional default value of this qualifier if not provided.

6.8.1.7 Native name

Location: /qualifiers_definition/[]/native_name

Type: String

The equivalent native name for this qualifier key.

All items shall be unique.

6.9 Subpath definition

Location: /subpath_definition

Property: subpath_definition (Optional)

Type: Object

The definition for the subpath for this PURL type. The PURL subpath component is optional for a specific PURL type definition.

Table 8 — Properties for the subpath_definition object

Property	Type	Requirement	Description
requirement	Array	Required	States that the PURL subpath is optional.
permitted_characters	String	Optional	A regular expression (ECMA-262 dialect) defining the 'Permitted characters' for this component of this Package-URL type. If provided, this shall be a subset of the 'Permitted characters' defined in the PURL specification.
case_sensitive	Boolean	Optional	true if this PURL component is case sensitive. If false , the canonical form shall be lowercased.
normalization_rules	Array	Optional	List of rules to normalize this component for this PURL type. These are plain text, unstructured rules as some require programming and cannot be enforced only with a schema. Tools are expected to apply these rules programmatically.
native_name	String	Optional	The native name of this PURL component in the package ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupId', and 'scope' for the 'npm' PURL type.
note	String	Optional	Extra note text.

6.9.1 Subpath requirement

Location: /subpath_definition/requirement

Property: requirement (Required)

Type: String

States that the PURL subpath is optional.

Shall be one of:

1. Component optional requirement

6.9.2 Component optional requirement

Type: String

Constant: optional

States that this PURL component is optional for a PURL type.

6.9.3 Permitted characters in this PURL component

Location: /subpath_definition/permitted_characters

Property: permitted_characters (Optional)

Type: String

Format: A regular expression dialect defined by [ECMA-262](#)

A regular expression ([ECMA-262](#) dialect) defining the 'Permitted characters' for this component of this Package-URL type. If provided, this shall be a subset of the 'Permitted characters' defined in the PURL specification.

6.9.4 Case sensitive

Location: /subpath_definition/case_sensitive

Property: case_sensitive (Optional)

Type: Boolean

Default Value: **true**

true if this PURL component is case sensitive. If **false**, the canonical form shall be lowercased.

6.9.5 Normalization rules

Location: /subpath_definition/normalization_rules

Property: normalization_rules (Optional)

Type: array (of String)

List of rules to normalize this component for this PURL type. These are plain text, unstructured rules as some require programming and cannot be enforced only with a schema. Tools are expected to apply these rules programmatically. Each item of this array shall be a string.

All items shall be unique.

6.9.6 Native name

Location: /subpath_definition/native_name

Property: native_name (Optional)

Type: String

The native name of this PURL component in the package ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupid', and 'scope' for the 'npm' PURL type.

6.9.7 Note

Location: /subpath_definition/note

Property: note (Optional)

Type: String

Extra note text.

6.10 PURL examples

Location: /examples

Property: examples (Required)

Type: array (of String)

Pattern Constraint: `^pkg:[a-z][a-z0-9-\.]*/.*$`

Example of valid, canonical PURLs for this package type. Each item of this array shall be a string.

All items shall be unique.

6.11 Note

Location: /note

Property: note (Optional)

Type: String

Note about this PURL type.

6.12 Reference URLs

Location: /reference_urls

Property: reference_urls (Optional)

Type: array (of String)

Format: URI as specified in [RFC 3986](#)

Optional list of informational reference URLs about this PURL type. Each item of this array shall be a string.

All items shall be unique.

Annex A (normative)

PURL Type Definition

This Annex provides a copy of the current Package-URL Type Definition Schema. The format is JSON Schema version [draft-07](#).

The schema shown below is available in electronic form at: <https://github.com/package-url/purl-spec/blob/main/schemas/purl-type-definition.schema.json>

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://packageurl.org/schemas/purl-type-definition.schema-1.0.json",
  "title": "Package-URL Type Definition",
  "description": "Schema to specify a Package-URL (PURL) type as a structured
definition.",
  "type": "object",
  "additionalProperties": false,
  "definitions": {
    "optional_requirement": {
      "title": "Component optional requirement",
      "description": "States that this PURL component is optional for a PURL
type.",
      "type": "string",
      "const": "optional"
    },
    "required_requirement": {
      "title": "Component required requirement",
      "description": "States that this PURL component is required for a PURL
type.",
      "type": "string",
      "const": "required"
    },
    "prohibited_requirement": {
      "title": "Component prohibited requirement",
      "description": "States that this PURL component is prohibited for a PURL
type.",
      "type": "string",
      "const": "prohibited"
    },
    "purl_component_definition": {
      "title": "PURL component definition",
      "description": "PURL component definition properties that apply to most PURL
components",
      "type": "object",
      "properties": {
        "permitted_characters": {
          "title": "Permitted characters in this PURL component",
          "description": "A regular expression (ECMA-262 dialect) defining the
'Permitted characters' for this component of this Package-URL type. If provided,
this shall be a subset of the 'Permitted characters' defined in the PURL
specification.",
          "type": "string",
          "format": "regex"
        }
      },
      "case_sensitive": {
```

```

        "title": "Case sensitive",
        "description": "true if this PURL component is case sensitive. If false,
the canonical form shall be lowercased.",
        "type": "boolean",
        "default": true
    },
    "normalization_rules": {
        "title": "Normalization rules",
        "description": "List of rules to normalize this component for this PURL
type. These are plain text, unstructured rules as some require programming and
cannot be enforced only with a schema. Tools are expected to apply these rules
programmatically.",
        "type": "array",
        "uniqueItems": true,
        "items": {
            "type": "string"
        }
    },
    "native_name": {
        "title": "Native name",
        "description": "The native name of this PURL component in the package
ecosystem. For instance, the 'namespace' for the 'maven' type is 'groupId', and
'scope' for the 'npm' PURL type.",
        "type": "string"
    },
    "note": {
        "title": "Note",
        "description": "Extra note text.",
        "type": "string"
    }
}
},
"required": [
    "$id",
    "type",
    "type_name",
    "description",
    "repository",
    "namespace_definition",
    "name_definition",
    "examples"
],
"properties": {
    "$schema": {
        "title": "JSON schema",
        "description": "Contains the URL of the JSON schema for Package-URL type
definition.",
        "const": "https://packageurl.org/schemas/purl-type-
definition.schema-1.0.json",
        "format": "uri"
    },
    "$id": {
        "title": "PURL type definition id",
        "description": "The unique identifier URI for this PURL type definition.",
        "type": "string",
        "pattern": "^https:\\/\\/\\/packageurl\\.org/types/[a-z0-9-]+-
definition\\.json$"
    },
    "type": {
        "title": "PURL type",

```

```

    "description": "The type string for this Package-URL type.",
    "type": "string",
    "pattern": "^[a-z][a-z0-9-\\.]+$",
    "examples": [
      "maven",
      "npm",
      "pypi"
    ]
  },
  "type_name": {
    "title": "Type name",
    "description": "The name for this PURL type.",
    "type": "string",
    "examples": [
      "Apache Maven",
      "Python Package"
    ]
  },
  "description": {
    "title": "Description",
    "description": "The description of this PURL type.",
    "type": "string"
  },
  "repository": {
    "title": "Repository",
    "description": "The package repository usage for this PURL type.",
    "type": "object",
    "additionalProperties": false,
    "required": [
      "use_repository"
    ]
  },
  "properties": {
    "use_repository": {
      "title": "Use repository",
      "description": "true if this PURL type uses a public package
repository.",
      "type": "boolean",
      "default": false
    },
    "default_repository_url": {
      "title": "Default repository URL",
      "description": "The default public repository URL for this PURL type",
      "type": "string",
      "format": "uri"
    },
    "note": {
      "title": "Note",
      "description": "Extra note text.",
      "type": "string"
    }
  }
},
"namespace_definition": {
  "title": "Namespace definition",
  "description": "Definition of the namespace component for this PURL type.
The PURL namespace component shall be required, optional or prohibited for a
specific PURL type definition.",
  "type": "object",
  "required": [
    "requirement"
  ],

```

```

    "properties": {
      "requirement": {
        "title": "Namespace requirement",
        "description": "States that the PURL namespace component is optional,
required or prohibited for a PURL type.",
        "type": "string",
        "oneOf": [
          {
            "$ref": "#/definitions/optional_requirement"
          },
          {
            "$ref": "#/definitions/required_requirement"
          },
          {
            "$ref": "#/definitions/prohibited_requirement"
          }
        ]
      }
    },
    "allOf": [
      {
        "$ref": "#/definitions/purl_component_definition"
      }
    ]
  },
  "name_definition": {
    "title": "Name definition",
    "description": "Definition of the name component for this PURL type. The
PURL name component is required for all PURL type definitions.",
    "type": "object",
    "required": [
      "requirement"
    ],
    "properties": {
      "requirement": {
        "title": "Name component requirement",
        "description": "States that the PURL name component is always
required.",
        "type": "string",
        "oneOf": [
          {
            "$ref": "#/definitions/required_requirement"
          }
        ]
      }
    }
  },
  "allOf": [
    {
      "$ref": "#/definitions/purl_component_definition"
    }
  ]
},
"version_definition": {
  "title": "Version definition",
  "description": "Definition of the version component for this PURL type. The
PURL version component is optional for a specific PURL type definition.",
  "type": "object",
  "required": [
    "requirement"
  ],
  "properties": {

```

```

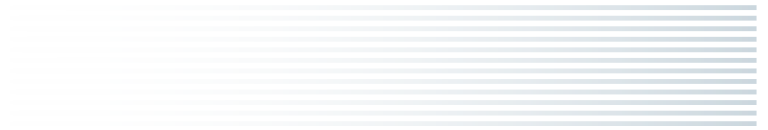
    "requirement": {
      "title": "Version requirement",
      "description": "States that the PURL version is optional.",
      "type": "string",
      "oneOf": [
        {
          "$ref": "#/definitions/optional_requirement"
        }
      ]
    },
    "allOf": [
      {
        "$ref": "#/definitions/purl_component_definition"
      }
    ]
  },
  "qualifiers_definition": {
    "title": "Qualifiers definition",
    "description": "Definition of the qualifiers specific to this PURL type. The PURL qualifiers component is optional for a specific PURL type, but a qualifiers key or keys may be required for a specific PURL type.",
    "type": "array",
    "additionalItems": false,
    "uniqueItems": true,
    "items": {
      "title": "Qualifiers definition",
      "description": "The definition of a qualifier specific to this PURL type.",
      "type": "object",
      "additionalProperties": false,
      "required": [
        "key",
        "description"
      ],
      "properties": {
        "key": {
          "title": "Qualifier key",
          "description": "The key for the qualifier.",
          "type": "string"
        },
        "requirement": {
          "title": "Qualifier key requirement",
          "description": "States that a PURL qualifier key is optional or required for a PURL type.",
          "type": "string",
          "oneOf": [
            {
              "$ref": "#/definitions/optional_requirement"
            },
            {
              "$ref": "#/definitions/required_requirement"
            }
          ]
        }
      }
    },
    "description": {
      "title": "Description",
      "description": "The description of this qualifier.",
      "type": "string"
    },
    "default_value": {

```

```

        "title": "Default value",
        "description": "The optional default value of this qualifier if not
provided.",
        "type": "string"
    },
    {
        "native_name": {
            "title": "Native name",
            "description": "The equivalent native name for this qualifier key.",
            "type": "string"
        }
    }
},
"subpath_definition": {
    "title": "Subpath definition",
    "description": "The definition for the subpath for this PURL type. The PURL
subpath component is optional for a specific PURL type definition.",
    "type": "object",
    "required": [
        "requirement"
    ],
    "properties": {
        "requirement": {
            "title": "Subpath requirement",
            "description": "States that the PURL subpath is optional.",
            "type": "string",
            "oneOf": [
                {
                    "$ref": "#/definitions/optional_requirement"
                }
            ]
        }
    }
},
"allof": [
    {
        "$ref": "#/definitions/purl_component_definition"
    }
]
},
"examples": {
    "title": "PURL examples",
    "description": "Example of valid, canonical PURLs for this package type.",
    "type": "array",
    "uniqueItems": true,
    "minItems": 1,
    "items": {
        "type": "string",
        "pattern": "^pkg:[a-z][a-z0-9-\\.]+/.*$"
    }
},
"note": {
    "title": "Note",
    "description": "Note about this PURL type.",
    "type": "string"
},
"reference_urls": {
    "title": "Reference URLs",
    "description": "Optional list of informational reference URLs about this
PURL type.",
    "type": "array",
    "uniqueItems": true,

```

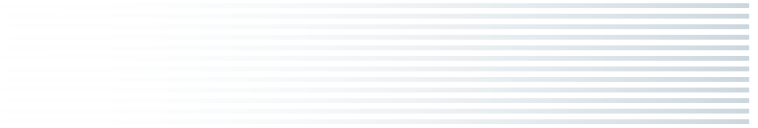


```
"items": {  
  "type": "string",  
  "format": "uri"  
}  
}  
}
```



Bibliography

- ECMA-262, *ECMAScript® language specification*, 16th Edition. Annex A.8: Grammar Summary for Regular Expressions
<https://262.ecma-international.org/16.0/index.html#sec-regular-expressions>
- ECMA-404, *The JSON data interchange syntax*
<https://www.ecma-international.org/publications-and-standards/standards/ecma-404/>
- INCITS 4-1986 [R2022], *Information Systems - Coded Character Sets - 7-Bit Standard Code for Information Interchange (7-Bit ASCII)*
<https://webstore.ansi.org/standards/incits/incits1986r2022>
- IETF Draft, March 2018, *JSON Schema Validation: A Vocabulary for Structural Validation of JSON (Draft-07)*
<https://datatracker.ietf.org/doc/html/draft-handrews-json-schema-validation-01>



Colophon

This Standard is authored on [GitHub](#) in a plaintext source format called [Ecm Markup](#). Ecm Markup is an HTML and Markdown dialect that provides a framework and toolset for authoring ECMA specifications in plaintext and processing the specification into a full-featured HTML rendering that follows the editorial conventions for this document. Ecm Markup builds on and integrates a number of other formats and technologies including [Grammarkdown](#) for defining syntax and [Ecm Markupdown](#) for authoring algorithm steps. PDF renderings of this Standard are produced using a print stylesheet which takes advantage of the CSS Paged Media specification and is converted using [PrinceXML](#).

We extend our gratitude to TC39 for their exceptional work in developing Ecm Markup, which has greatly facilitated TC54's successful adoption of this tool for the preparation and maintenance of our technical specifications.



Software License

Ecma International
Rue du Rhone 114
CH-1204 Geneva
Tel: +41 22 849 6000
Fax: +41 22 849 6001
Web: <https://ecma-international.org/>

All Software contained in this document ("Software") is protected by copyright and is being made available under the "BSD License", included below. This Software may be subject to third party rights (rights from parties other than Ecma International), including patent rights, and no licenses under such third party rights are granted under this license even if the third party concerned is a member of Ecma International. SEE THE ECMA CODE OF CONDUCT IN PATENT MATTERS AVAILABLE AT <https://ecma-international.org/memento/codeofconduct.htm> FOR INFORMATION REGARDING THE LICENSING OF PATENT CLAIMS THAT ARE REQUIRED TO IMPLEMENT ECMA INTERNATIONAL STANDARDS.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the authors nor Ecma International may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE ECMA INTERNATIONAL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ECMA INTERNATIONAL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

