ISO Document: ISO/IEC 00000:2023(E)
Document Number: Dxxxx
Date: 2023-06-02
Revises: Nnnnn
Reply to: ISO C++ Tooling Study Group
sg15@lists.isocpp.org

Working Draft, Standard for C++ Ecosystem

Note: this is an early draft. It's known to be incomplet and incorrekt, and it has lots of bad formatting.

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/IEC JTC1, Information technology, Subcommittee 22, Programming languages, their environments and system software interfaces, Working Group 21, C++.

The main changes are as follows:

— Initial release.

A list of all parts in the ISO/IEC 00000 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

1 Scope [intro.scope]

[1] This document specifies formats, processes, definitions, and so on, that facilitates the interoperation of the tools and systems that implement, and interface with, the C++ programming language.

[2] C++ is a general purpose programming language described in ISO/IEC 14882:2020 *Programming languages — C++* (hereinafter referred to as the *C++ standard*).

2 Normative references [intro.refs]

[1] 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.

Vocabulary ISO/IEC 2382, *Information technology — Vocabulary*

POSIX ISO/IEC 9945:2009, *Information technology — Portable Operating System Interface (POSIX®) Base Specifications, Issue 7*

C++ ISO/IEC 14882:2020, *Programming languages — C++*

JSON ISO/IEC 21778:2017, *Information technology — The JSON data interchange syntax*

Mathematics ISO 80000-2:2019, *Quantities and units — Part 2: Mathematics*

3 Conformance [intro.cnf]

1 A conforming implementation shall meet the following criteria for conformance to this standard:

(1.1) — An application shall support the minimum level functionality of introspection (5.8).

4 Terms and definitions [intro.defs]

1 For the purposes of this document, the terms and definitions given in ISO/IEC 2382, the terms and definitions given in ISO/IEC 14882:2020, and the following apply.

2 ISO and IEC maintain terminology databases for use in standardization at the following addresses:

(2.1) — ISO Online browsing platform: available at https://www.iso.org/obp

(2.2) — IEC Electropedia: available at https://www.electropedia.org/

3 Terms that are used only in a small portion of this document are defined where they are used and italicized where they are defined.

4.1 [defns.application]

application
a computer program that performs some desired function.

[*Note 1 to entry*: From POSIX. — *end note*]

4.2 [defns.capability]

capability
an aspect of an overall specification that defines a subset of the entire specification.

4.3 [defns.file]

file
an object that can be written to, or read from, or both.

[*Note 1 to entry*: From POSIX. — *end note*]

5 Introspection [intspct]

5.1 Preamble [intspct.pre]

1 This clause describes options, output, and formats that describe what capabilities of this standard an application supports. An application shall support the *minimum level* functionality (5.8). An application can support the *full level* functionality (5.9).

2 This clause specifies the `std:info` capability (5.6).

5.2 Overview [intspct.overview]

1 *`application`* `[--std-info[=`*declaration*`]] [--std-info-out=`*`file`*`]`

5.3 Options [intspct.options]

5.3.1 Information Option [intspct.opt.info]

1 This option shall be supported.

2 `--std-info`

Outputs the version information of the capabilities supported by the application. The option can be specified zero or one time. The application shall support the option for *minimum level* (5.8) functionality.

5.3.2 Information Output Option [intspct.opt.out]

1 This option shall be supported.

2 `--std-info-out=`*file*

The pathname of a file to output the information to. If *file* is '`-`', the standard output shall be used. The application shall support the option for *minimum level* (5.8) functionality. Not specifying this option while specifying the `--std-info` option (5.3.1) shall be equivalent to also specifying a `--std-info-out=-` option.

5.3.3 Declaration Option [intspct.opt.decl]

1 This option should be supported.

2 `--std-info=`*declaration*

Declares the required capability version of the application. The option can be specified any number of times. The application shall support the option for *full level* (5.9) functionality.

5.4 Output [intspct.output]

1 An application shall output a valid JSON text file that conforms to the introspection schema (5.5) to the file specified in the options (5.3.2).

5.5 Schema [intspct.schema]

1 An introspection JSON text file shall contain one introspection JSON object (5.5.1).

5.5.1 Introspection Object [intspct.schema.obj]

1 The *introspection object* is the root JSON object of the introspection JSON text.

2 An *introspection object* can have the following fields.

5.5.2 JSON Schema Field [intspct.schema.schema]

1 *Name*: `$schema`

2 *Type*: `string`

3 *Value*: The value shall be a reference to a JSON Schema specification.

4 *Description*: An *introspection object* can contain this field. If an *introspection object* does not contain this field the value shall be a reference to the JSON Schema corresponding to the current edition of this standard.

5.5.3 Capability Field [intspct.schema.cap]

1 *Name*: *capability-identifier* (5.6)

2 *Type*: `string` or `array`

3 *Value*: (for `string`) The value shall be a *version-number* for *minimum level* functionality. Or the value shall be a *version-range* for *full level* functionality.

4 *Value*: (for `array`) The value can be a JSON `array` for *full level* functionality. If the value is a JSON `array` the items in the `array` shall be a *version-number* or *version-range*.

5 *Description*: An *introspection object* can contain this field one or more times. When the field appears more than one time the name of the fields shall be unique within the *introspection object*.

5.6 Capabilities [intspct.cap]

capability-identifier:
name scope-designator name sub-capability-identifier$_{opt}$

sub-capability-identifier:
scope-designator name sub-capability-identifier$_{opt}$

name:
one or more of:
U+*0061* .. U+*007A* LATIN SMALL LETTER A .. Z
U+*005F* LOW LINE

scope-designator:
U+*003A* COLON

1 A *capability-identifier* is composed of two or more *scope-designator* delimited *name* parts.

2 The *name* `std` in a *capability-identifier* is reserved for capabilities defined in this standard.

3 Applications can specify vendor designated *name* parts defined outside of this standard.

5.7 Versions [intspct.vers]

1 A version shall be either a single version number (5.7.1) or a version range (5.7.2).

2 A single version number shall be equivalent to the inclusive version range spanning solely that single version number.

[*Note 1*: That is, the version number `i.j.k` is equivalent to version range `[i.j.k,i.j.k]`. — *end note*]

5.7.1 Version Number [intspct.vers.num]

version-number:
major-number minor-patch-part$_{opt}$

minor-patch-part:
U+*002E* FULL STOP *minor-number patch-part*$_{opt}$

patch-part:
U+*002E* FULL STOP *patch-number*

major-number:
decimal

minor-number:
decimal

patch-number:
decimal

decimal:
one of: U+*0030* .. U+*0039* DIGIT ONE .. NINE
zero or more of: U+*0030* .. U+*0039* DIGIT ZERO .. NINE

1 A *decimal* is a sequence of decimal digits (U+0030 .. U+0039 DIGIT ZERO .. NINE) with no superfluous leading zeros (U+0030 DIGIT ZERO).

2 A *decimal* represents a non-negative base 10 integer.

3 A version number is composed of 1, 2, or 3 decimal numbers (*decimal*) separated by a U+002E FULL STOP.

4 A version number composed of 1 decimal number is equivalent to that decimal number followed by `.0.0`.

[*Note 1*: That is, the version number `N` is equivalent to `N.0.0`. — *end note*]

5 A version number composed of 2 decimal number parts is equivalent to those decimal number parts followed by `.0`.

[*Note 2*: That is, the version number `N.M` is equivalent to `N.M.0`. — *end note*]

6 Version numbers define a total ordering where version number a with decimal parts i, j, k is ordered before version number b with decimal parts p, q, r when: i is less than p, or i is equal to p and j is less than q, or i is equal to p and j is equal to q and k is less than r.

7 Otherwise version number a is ordered before version number b when: i is greater than p, or i is equal to p and j is greater than q, or i is equal to p and j is equal to q and k is greater than r.

8 Otherwise version number a is the same as version number b.

5.7.2 Version Range [intspct.vers.range]

version-range:
version-range-min-bracket version-min-number version-range-max-part$_{opt}$ *version-range-max-bracket*

version-range-max-part:
U+002C COMMA *version-max-number*

version-min-number:
version-number

version-max-number:
version-number

version-range-min-bracket:
one of: U+005B LEFT SQUARE BRACKET U+0028 LEFT PARENTHESIS

version-range-max-bracket:
one of: U+005D RIGHT SQUARE BRACKET U+0029 RIGHT PARENTHESIS

1 A version range is composed of either one version number bracketed, or two version numbers separated by a U+002C COMMA and bracketed.

[*Example 1*: A version range with a single version number `"[1.0.0]"`. — *end example*]

[*Example 2*: A version range with a two version numbers `"[1.0.0,2.0.0]"`. — *end example*]

2 A version range a that is `[`i`,`j`]` makes i and j inclusive version range numbers, defining a Mathematics closed interval.

3 A version range a that is `(`i`,`j`)` makes i and j exclusive version range numbers, defining a Mathematics open interval.

4 A version range a that is `(`i`,`j`]` makes i an exclusive version number and j an inclusive version number, defining a Mathematics half-open interval.

5 A version range a that is `[`i`,`j`)` makes j an exclusive version number.

6 A version range with a single inclusive version number x is equivalent to the version range `[`x`,`x`]`.

7 A version range with a single exclusive version number x is invalid.

8 An exclusive version number x does not include the version number x when compared to another version number y.

9 A version range a with version numbers i and j when compared to a version range b with version number m and n will result in an empty version range when: $j < m$ or $n < i$.

10 Otherwise if i or m are inclusive version numbers and if j or n are inclusive version numbers the resulting range when a is compare to b is the inclusive version numbers "lesser of i and m" and "lesser of j and n".

11 Otherwise if i or m are inclusive version numbers and if j or n are inclusive version numbers the resulting range when a is compare to b is the inclusive version number "lesser of i and m" and the exclusive version number "lesser of j and n".

12 Otherwise if j or n are inclusive version numbers the resulting range when a is compared to b is the exclusive version number "lesser of i and m" and the inclusive version number "lesser of j and n".

13 Otherwise the resulting range when a is compared to b is the exclusive version numbers "lesser of i and m" and "lesser of j and n".

5.8 Minimum Level [intspct.min]

1 An application that supports the *minimum level* functionality indicates it by specifying a single version (**??**) as the value of the `std:info` capability (5.6).

[*Example 1*: `{ "std:info": "1.0.0" }` — *end example*]

5.9 Full Level [intspct.full]

1 An application can support the *full level* functionality as defined in this section. An application that reports supporting the *full level* functionality shall support all of the functionality in this section.

2 An application that supports the *full level* functionality indicates it by specifying a version range (5.7.2) or an array of version range items as the value of the `std:info` capability (5.6).

[*Example 1*: `{ "std:info": "[1.0.0]" }` — *end example*]

An application that responds with an array of version range items as the value of a capability field shall support the union of the range items indicated.

5.10 Introspection Information [intspct.info]

1 An application shall output an introspection schema (5.5) that contains one capability field for each capability that the application supports when given the `--std-info` option (5.3.1).

2 An application shall indicate the single version (5.7.1) or version range (5.7.2) of each capability it supports as the value of the capability field.

5.11 Introspection Declaration [intspct.dcl]

1 An application that supports the *full level* functionality when given one or more `--std-info=`*declaration* options shall conform its functionality to the indicated edition of this standard in the given *declaration version-number* for the given capability.

declaration:
capability-identifier U+003D EQUALS SIGN *version-number*

2 An application, when not given a `--std-info=`*declaration* option for a capability it supports, should conform its functionality to the most recent version of the standard it supports for that capability.

3 An application, when given a capability declaration option and the given version is outside of the version range that the application supports, should indicate an error.

Annex A (informative)
Tool Introspection JSON Schema [intsjschm]

A.1 General [intsjschm.general]

1 This Annex defines the introspection capability schema (5.5) in terms of a *JSON Schema*. A *JSON Schema* refers to the IETF RFC draft "JSON Schema: A Media Type for Describing JSON Documents" as specified in https://json-schema.org/draft/2020-12/json-schema-core.html.

2 This JSON Schema can be referenced as the `$schema` field with URI value of `"https://raw.githubusercontent.com/cplusplus/ecosystem-is/release/schema/std_info-1.0.0.json"`.

A.2 JSON Schema Specification [intsjschm.spec]

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id":
    "https://raw.githubusercontent.com/cplusplus/ecosystem-is/release/schema/std_info-1.0.0.json",
  "title": "Tool Introspection Version 1.0.0 JSON Schema",
  "$defs": {
    "VersionMin": {
      "type": "string",
      "pattern": "^[0-9]+([.][0-9]+){0,2}$"
    },
    "VersionFull": {
      "type": "string",
      "pattern": "^[[(][0-9]+([.][0-9]+){0,2}[)\\]]$"
    },
    "VersionRange": {
      "type": "string",
      "pattern": "^[[(][0-9]+([.][0-9]+){0,2},[0-9]+([.][0-9]+){0,2}[)\\]]$"
    },
    "Version": {
      "oneOf": [
        {
          "$ref": "#/$defs/VersionMin"
        },
        {
          "$ref": "#/$defs/VersionFull"
        },
        {
          "$ref": "#/$defs/VersionRange"
        }
      ]
    },
    "Versions": {
      "type": "array",
      "items": {
        "$ref": "#/$defs/Version"
      }
    },
    "VersionSpec": {
      "oneOf": [
        {
          "$ref": "#/$defs/Version"
        },
        {
          "$ref": "#/$defs/Versions"
        }
      ]
```



```
    }
  },
  "anyOf": [
    {
      "type": "object",
      "properties": {
        "$schema": {
          "description":
            "JSON Schema URI for the version of the tool introspection format.",
          "type": "string",
          "format": "uri"
        },
        "std:info": {
          "description": "The Tool Introspection format version.",
          "$ref": "#/$defs/VersionSpec"
        }
      },
      "patternProperties": {
        "^[a-z_]+(:[a-z_]+)+$": {
          "$ref": "#/$defs/VersionSpec"
        }
      },
      "additionalProperties": false
    }
  ],
  "required": [
    "std:info"
  ]
}
```


Bibliography

— ISO xxxx:YYYY, *Title*

Cross references

Each clause and subclause label is listed below along with the corresponding clause or subclause number and page number, in alphabetical order by label.

Index

Constructions whose name appears in *`monospaced italics`* are for exposition only.

