ISO/IEC JTC 1 Public Comments on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Submitted by the United States

Prepared by Rex Jaeschke
rex@RexJaeschke.com
2002/04/15

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Comment number: US-C#-001

Comment category: (please indicate one) _X_ Technical __ Editorial

Summary:

§12.3, Definite Assignment, does not stand on its own.

References: (use actual document page number, not the PDF page number)

Section 12.3, page 100, lines 5–14

Detailed description:

The sentence on line 5 states “At a given location in the executable code of a function member, a variable is said to be *definitely assigned* if the compiler can prove, by static flow analysis, that the variable has been automatically initialized or has been the target of at least one assignment.”

This begs the question of what definite assignment is. Is there some particular compiler, included by reference in the specification that must be able to supply this proof? If so, where do I find that compiler? Or is every C# compiler required to be able to provide this proof for some given program? If the latter, then does a valid C# program become invalid when a new, less clever C# compiler becomes available? What kind of “static flow analysis” is required, exactly?

The next sentence (that starting on line 7) would appear to be intending to clarify, but it doesn't: “The rules of definite assignment are:

- An initially assigned variable is always considered definitely assigned.
- An initially unassigned variable is considered definitely assigned at a given location if all possible execution paths leading to that location contain at least one of the following: ...”

Unfortunately, this doesn't clear up much. The concept of a “possible execution path” would tend to suggest that a compiler for C# must be capable of solving the halting problem in order to correctly compile the language. If something less is required, it must be spelled out explicitly and with sufficient specificity that no confusion of this kind can result.

The third bullet in the section quoted above (see line 14) should at lease require that the declaration be of the variable of interest.

Proposed solution:

_____

_____

…

_____

_____

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Comment number: US-C#-002

Comment category: (please indicate one) __ Technical _X_ Editorial

Summary:

The casing of the output in an example is unexpected.

References: (use actual document page number, not the PDF page number)

Section 8.2.1, page 19, lines 17–18

Detailed description:

If the table on page 18 is correct, the Boolean values are “true” and “false”, not “True” and “False”.

Proposed solution:

Convert the ‘T’ and ‘F’ in the output to lowercase. This corresponds to the behavior of most languages and is what the user will expect. (Remember this language may become the first language that some people learn, surprises like this make learning harder.)

If this is unacceptable, explain why in the example.

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Comment number: US-C#-003

Comment category: (please indicate one) __ Technical _X_ Editorial

Summary:

A portion of the grammar in Annex A is misplaced.

References: (use actual document page number, not the PDF page number)

Section A.2

Detailed description:

The portion of the grammar referring to compilation units (§16.1, page 199, line 6) is misplaced in Annex A. Since a compilation unit is the basic component of a C# specification for a collection of objects, this should appear at the beginning of the grammar.

Proposed solution:

Move the compilation units material from A.2.5, page 347, lines 18-19 to A.2.1, page 339, line 42.

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

`ISO/IEC DIS 23270, Information technology - C# Language Specification`

Comment number: US-C#-004

Comment category: (please indicate one) __ Technical _X_ Editorial

Summary:

The term “character” is misused with respect to Unicode code units.

References: (use actual document page number, not the PDF page number)

Section: various

Detailed description:

See the proposed solution below.

Proposed solution:

1. Replace §8.2.1, page 18, lines 3–4, “The `char` type is used to represent Unicode characters. A variable of type `char` represents a single 16-bit Unicode character.”

   with

   “The `char` type is used to represent Unicode code units. A variable of type `char` represents a single 16-bit Unicode code unit.”

2. Replace §8.2.1, page 18, line 9, table entry for `string` “String type; a string is a sequence of Unicode characters”

   with

   “String type; a string is a sequence of Unicode code units”

3. Replace §8.2.1, page 18, line 9, table entry for `char` “Character type; a `char` value is a Unicode character”

   with

   “Character type; a `char` value is a Unicode code unit”

4. Replace §9.4.1, page 52, lines 31–33, “Since C# uses a 16-bit encoding of Unicode characters in characters and string values, a Unicode character in the range `U+10000` to `U+10FFFF` is represented using two Unicode surrogate characters. Unicode characters with code points above `0x10FFFF` are not supported.”

   with

   “Since C# uses a 16-bit encoding of Unicode characters in chars and strings, a Unicode code point in the range `U+10000` to `U+10FFFF` is represented using two Unicode surrogate code units. Unicode code points above `0x10FFFF` are illegal and not supported.”

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Comment number: US-C#-005

Comment category: (please indicate one) __ Technical _X_ Editorial

Summary:

Source code line breaks are overly restrictive.

References: (use actual document page number, not the PDF page number)

Section 9.1, page 49, lines 16–19

Detailed description:

Regarding line breaks, it is stated, “During such transformation, however, it is recommended that the usual line-separating character (or sequence) in the other character set be translated to the two-character sequence consisting of the Unicode carriage-return character followed by Unicode line-feed character.”

This sounds like C# will not accept anything but CRLF as a line break. This is contrary to the newline guidelines, and to XML practice, and to what the syntax appears to specify. It should accept CR, LF, NEL, and CRLF as equivalent in parsing

Proposed solution:

Omit this sentence

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Comment number: US-C#-006

Comment category: (please indicate one) __ Technical _X_ Editorial

Summary:

Issue regarding the names of several Unicode-related methods.

References: (use actual document page number, not the PDF page number)

Section D, page 390, lines 4–7

Detailed description:

Taken as a group, the names of the following methods are a little misleading:

```
public static Encoding BigEndianUnicode { get; }
public static Encoding Default { get; }
public static Encoding Unicode { get; }
public static Encoding UTF8 { get; }
```

Specifically, `UTF8` and `BigEndianUnicode` are just as much “Unicode” as is `Unicode`.

Proposed solution:

_____

_____

…

_____

_____

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Comment number: US-C#-007

Comment category: (please indicate one) __ Technical _X_ Editorial

Summary:

Issues regarding organization and style.

References: (use actual document page number, not the PDF page number)

Section various

Detailed description:

See below.

Proposed solution:

1. The document should be restructured according to ISO/IEC drafting guidelines. For example, clause titles should be flush left, not flush right.

2. Search the document for instances of "must" and "can", and replace them with "shall" and "may", respectively, as appropriate.

3. §1, Scope, should be normative not informative.

4. Since normative references and definitions are needed before one can discuss conformance, sections 1–4 should be reordered as follows:

    1 Scope
    2 Normative References
    3 Definitions
    4 Conformance

5. In §3, References, page 5, lines 2–7, replace "The following normative documents contain provisions, which, through reference in this text, constitute provisions of this ECMA Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this ECMA Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative

document referred to applies. Members of ISO and IEC maintain registers of currently valid ECMA Standards."

with the current ISO/IEC boilerplate wording "The following referenced documents are indispensable for the application 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."

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Comment number: US-C#-008

Comment category: (please indicate one) _X_ Technical __ Editorial

Summary:

Question regarding the “next line” character as a line terminator.

References: (use actual document page number, not the PDF page number)

Section 9.3.1, page 50, lines 30-44

Detailed description:

Why isn't the Next Line character (U+0085) included?

Proposed solution:

_____

_____

…

_____

_____

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

ISO/IEC DIS 23270, Information technology - C# Language Specification

Comment number: US-C#-009

Comment category: (please indicate one) __ Technical _X_ Editorial

Summary:

Lack of clarity in the section covering automatic memory management.

References: (use actual document page number, not the PDF page number)

Section 10.9, page 82, line ___

Detailed description:

This section reads like it is informative.

Proposed solution:

Rewrite this section so the assertions/provisions are clear.

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

`ISO/IEC DIS 23270, Information technology - C# Language Specification`

Comment number: US-C#-010

Comment category: (please indicate one) _X_ Technical __ Editorial

Summary:

Issues regarding operator overloading.

References: (use actual document page number, not the PDF page number)

Section 14.2.2, page 123, line ___

Detailed description:

1. `true` and `false` are listed as operators, yet they don't appear to be operators and they aren't listed in the operator precedence table.

2. The statement on lines 17-18, “Only when no applicable user-defined operator implementations exist will the predefined operator implementations be considered.” Is unclear. Just what does “considered” mean?

Proposed solution:

_____

_____

…

_____

_____

ISO/IEC JTC 1 Public Comment on ECMA Fast Track

`ISO/IEC DIS 23270, Information technology - C# Language Specification`

Comment number: US-C#-011

Comment category: (please indicate one) __ Technical _X_ Editorial

Summary:

Question about the normative nature of text regarding delegates.

References: (use actual document page number, not the PDF page number)

Section 22, page 43, lines 1–3

Detailed description:

See below.

Proposed solution:

The wording ‘An interesting and useful property of a delegate instance is that it does not know or care about the classes of the methods it encapsulates; all that matters is that those methods be compatible (§22.1) with the delegate's type. This makes delegates perfectly suited for “anonymous” invocation.’ should probably be a Note; it doesn't look like normative wording.