ETSI/MTS(01)015

European Telecommunications Standards Institute
MTS#33
24 to 26 October 2001
Sophia-Antipolis

Source:	**Steve Randall**
Title:	**IESG statement on the use of formal languages in protocol specifications**
Date:	**17 October 2001**
Document for:	**Discussion**
Agenda item:	**6.5**

Von: The IESG [mailto:iesg-secretary@ietf.org]
Gesendet: Montag, 1. Oktober 2001 15:58
Betreff: IESG statement on the use of formal languages in protocol specifications

Guidelines for the use of formal languages in IETF specifications
---

This temporary guideline has been prepared by the IESG in response to a specific case; it is intended that it be put into an internet-draft and progressed as an IETF consensus position.

At the moment, it is an indication to document authors of things the IESG is likely to think of when reviewing a document using a formal language.

General

Formal languages are useful tools for specifying parts of protocols. However, as of today, there exists no well-known language that is able to capture the full syntax and semantics of reasonably rich IETF protocols.

We therefore expect that people will continue using English to describe protocols, with formal languages as a supporting mechanism.

Pseudocode

Often, one finds people specifying parts of a protocol in "pseudocode"; constructs that look like a programming language, but are not conforming to any fixed syntax. The IESG evaluation will be on clarity of specification, just as if the specification had been in English.

  o Procedural parts of a specification may be written in a pseudo-code which is unambiguously defined in the document. This is clearly a very good way in which to express the algorithm.

  o Use of a program in any known or intuitive programming language, including pseudo-code, may be used to illustrate and support a specification which is in itself complete.

The emphasis when using pseudocode needs to be on clarity.

Formal languages

When a specification makes use of a formal language, such as C, ASN.1, SMI, ABNF, UML or MOF, there are a number of considerations:

- The use of a language requires a reference to the specification for that language. This reference is normative, and needs to fulfil the usual requirements for normative references (section 7 of RFC 2026).

- The language must be used correctly according to its specification. It must be clear whether the language is really in the language, or just pseodocoe that happens to look like that language.

- The specification needs to be verifiable. This means that it should be relatively easy to extract the code from the document and run it through a verification tool for conformance to the language syntax. Use of code fragments is problematic in this regard; the commonly used technique of collecting all code fragments in a complete form as an appendix can often be an useful way to avoid this problem,

although care must be taken to make sure the appendix is consistent with the body of the text.

- The specification needs to be complete. In particular, any modules referenced but not included in the specification are normative references. Their syntactic and semantic properties need to be known to any implementor of the specification; therefore, stability of reference is as important as for the language itself, and accessibility of the specification is a primary concern. Reference to library components that are well defined by the standards for the language under consideration, and commonly supplied together with validation tools or compilers for the language, are not a problem.

- The specification needs to be reasonably implementation independent. This means in particular that care must be taken to avoid or document dependencies on implementation idiysyncrasies, such as size of integers, character set of implementation, and so on. It is also wise to avoid areas known to create problems, such as use of (for instance) dynamic memory allocation and pointer arithmetic.

The code in the specification is NOT required to itself be a reference implementation, but it must be clear from the document what parts of the
specification are outside the code.

Note: It is not required that syntax checking tools are available before a specification using a language is first entered on the IETF standards track. However, experience shows that people are lousy at checking formal syntax, so when such tools are available, they SHOULD be used.

A specification that fails verification tools is not likely to progress.