From: Frank Farance
Date: 1994-10-03
Document Number: WG14/N387 X3J11/94-072
Subject: X3J11 comments on IEEE 1238.1

This is a copy of the E-mail I send to the reflector.

From: Frank Farance, ANSI X3J11, Party of Interest
Date: 1994-07-06 22:45 ET
File: ~farance/standard/x3j11/notes/94070702.txt

Review of: IEEE P1238.1/D1.00, 1994-03, "Standard for Information Technology -- File Transfer, Access, and Management Services -- Application Program Interface [C Language Binding]"

Please respond to me via E-mail if you have any further questions.

Thank you for asking us to review your document. The following are suggestions from various technical experts in ANSI X3J11. These comments concern the applicability of ISO/ANSI C with respect to the C binding of FTAM.

@ 0 c 1
Problem:

Balloting instructions are not clear.

Action:

Provide a *correct* example (in addition to the *incorrect* examples).

@ 0 c 2
Problem:

I am not familiar with the OM headers, however, I was able to figure much of this out.

Action:

It would be useful to have a summary of the data types (e.g., OM_string, OM_enum, OM_object, etc.) as an appendix. If these have standard mappings, for example:

```
        typedef unsigned char *OM_string;
```

then these mappings would be useful for inclusion, too.

@ 5.0 c 3
Problem:

There is no state transition diagram that shows what actions are allowed at what point. For example, its not clear when "ft_delete()" can be called.

Action:

Include state transition diagram (see BSD sockets or UNIX streams as examples) or qualify each function in the API. The state transition diagram is probably easier to provide.

@ 5.0 c 4
Problem:

Programming with the API is not obvious. This is aggrevated by the lack of a description of the *program* flow (the appendix B description is a good functional description, but not good for programming).

Action:

Include a realistic example as an appendix. The UNIX "ftp" program coded using FTAM would be an excellent example. Be sure to provide all the important FTP commands (e.g., binary, ascii, get, put, mget, mput, delete, dir, ls, open, user, close).

@ 5.2 c 5
1 section 5.2 comment, page 67, line 21
Problem:

I don't know what the "OM_string" type maps into. In the prototype:

```
        FT_return_code ft_fcattributes(
                ...
                OM_string *filename ,
                ...
        );
```

It is not clear that this should be "*filename".

Action:

Shouldn't this be "filename"? If "OM_string" maps to "char *" or "char []", then this prototype is wrong. If "OM_string" maps to "char", then the prototype is right, but the name is wrong (should be "OM_char" instead -- if there is such a thing). If "OM_string" is a structure, then provide a useful example in an appendix.

@ 0 c 6
6 appendix A comment, page 118-119

Problem:

It is not clear what the example in figure A-3 is for. Furthermore, it is not used in the context of the FTAM API, so it is not clear how

538
~~537~~

this relates to FTAM. For example, what would one do with the newly created object "private_ftam_input"? What FTAM functions would we be able to call?

Action:

Provide useful example of OM functions as they interact with FTAM API.

@ 4.2 c 7
7 section 4.2 comment, page 44, line 26
Problem:

The use of the word "legal" in standards documents is used in only one context: the law. The word "legal" is used many times in this document (I'm won't itemize them all). You probably mean "valid" when you use the word "legal".

Action:

Change all uses of the word "legal" to "valid" - including identifiers.

@ 4.2 c 8
8 section 4.2 comment, page 44, line 26
Problem:

Qualifications is the only word that is abbreviated ("QUAL") in the C binding of the OM Attribute names.

Action:

Change "QUAL" to "QUALIFICATIONS".

@ 0 c 9
Problem:

I can't remember where I saw this -- the phrase was "mandatory parameter" in one the descriptions of the parameters to a function. With the C prototypes you're using *all* parameters are mandatory, an ISO/ANSI C compiler is required to diagnose too few, too many, or incompatible parameters (read: won't compile).

Action:

Either change the wording to remove "mandatory" (since it is implied) or use the varying arguments prototype, for example:

```
        int printf(const char *, ...);
```

The following is a copy of the comments I sent out on X3J11's behalf on IEEE 1238.1/D2. The D2 revision is about 20-30 ''change pages'' from the D1 revision. The comments on the D1 revision were sent to the X3J11 reflector on 1994-07-06.

-FF

---(cut here)-----------------

From: Frank Farance, ANSI X3J11, Party of Interest
Date: 1994-09-30 09:08 ET
Document Number: 940930 130821 01 FARANCE
File: ~farance/standard/x3j11/notes/94093013.txt

Review of: IEEE 1238.1/D2, 1994-08, ''Draft Standard for Information Technology -- File Transfer, Access, and Management Services -- Application Program Interface [C Language Binding]''

Please respond to me via E-mail if you have any further questions.

Thank you for asking us to review your document. The following are suggestions from various technical experts in ANSI X3J11. These comments concern the revision of the FTAM draft standard.

@ 3.3.10 c 12

Problem:

Functions called during ''interrupt servicing'' may not be safe. In C, no function calls are safe. POSIX.1 defines *some* function calls to be safe (extending the C definition). In is not clear how your document may interact with the multi-thread work that is proceeding in POSIX. The problem is not only reentrancy, but the stability of of the state of the FTAM subsystem, e.g., you'd have to use variables of type "sig_atomic_t" (defined in C standard) to be safe during interrupts.

Action:

Either specify that "ft_abandon()" and "ft_close()" are safe (not recommended and not possible in some environments) OR explain that the use of these functions in interrupt handlers may be dependent upon features outside of FTAM.

---(cut here)-----------------