From: Frank Farance
Organization: Farance Inc.
Telephone: +1 212 486 4700
Fax: +1 212 759 1605
E-mail: frank@farance.com
Date: 1995-08-04
Document Number: WG14/N456 X3J11/95-057
Subject: ANSI IISP Publication Needs For X3J11
To: Jean Baronas

---

1. COMMENTS ON MINUTES

In this section, I am providing comments on issues from the minutes of the previous meeting. The comments related to the discussion of the Portable Document Delivery Format.

1.1 Open Published Format

The following features and notes should be added to the requirements:

- Must be text-based (local text format -- use the filename extension ".txt"). This allows many of the current text processing tools to be used (e.g., POSIX.2). In a pinch, it is easier to use a text editor to change the file than to write a special purpose binary editor. Note that virtually all tools that use binary formats provide very limited editing capability. Thus, text is a must-have.

- The ''binary form'' of the file (e.g., the form that would be distributed on electronic or physical media) should be ''binary text format'' (use the filename extension ".btf"). This is identical to using ASCII characters in a binary file (see "fopen" library function in C). This works and is compatible on all systems (e.g., the "compress" program makes use of this).

- Note that the lines in the file cannot exceed 509 characters per line (excluding newline). This limitation is the C Standard limitation of text lines and is based upon the experience of the C compiler vendors and their operating systems (this includes almost every machine architecture and operating system).

- This text format can be sent directly via E-mail, assuming the document is sent in the following ''wrapper'': the first character of each line is prepended with an "X" before sending via E-mail and stripped prior to saving the E-mail -- this is a limitation of some mailers (i.e., avoid "From" being translated to ">From").

1.2 Current Commercial Availability

This section should be re-termed ''prior art''. While ''prior art'' may establish feasibility, it shouldn't necessarily be used as a baseline for a standard. We must weigh the vendors' needs against users' needs. Especially in the electronic publications area, commerical vendors have a long history of proprietary, closed formats customized to their business advantage. Unless a commercial vendor is willing to distribute source code for their proposed interface, their contribution should be treated with caution.

1.3 Platform Independence

This section should be re-termed ''conformance''. There may be several levels of conformance. There should be ways to assert and/or inquire about conformance.

1.4 Computer Storage Space Efficiency

The following should be noted with respect to compression features.

If compression is less than 25%, then compression shouldn't be used.

Remember, binary format is expanded by 33% when converting text (i.e., uuencode).

Compression is not free. Compression is computationally expensive, especially in print engines. For example,

(page 3 is missing)

```
                    # Send the document tail.
                    cat doc.tail
        )
```

Using a flexible compression technique like this allows the compression to be added or removed at will, depending on the performance requirements. Likewise, the user can add or remove internal or external compression, as his/her needs change.

1.5 Support Access/Use Restrictions

Like compression, this should be a separable feature, not hardwired in. For example, if my company has paid all the fees for unlimited usage, this shouldn't be part of the file. If I must pay a per-use fee, then they may be included in the file, but like the separate compression feature above.

1.6 Document Construction/Rearrangement Capabilities

This should not be part of the ''final form''. This will make the document format impractical. For example, would you include the original SGML source embedded in the ''final form''? If you don't you've lost the structure of the document so editing is impossible or useless. Thus, the only way to make a truly editable document is to include all the source, all the confirugation options, and, possibly, a copy of your editing tool (for several machine architectures) to get this format working right. This might take a 100K Postscript image and increase it to 20MB (with all the executables you must supply) -- at a factor of 2000 increase. Like executables for programs (think ''final form''), we don't distribute embedded source code. We shouldn't do the same for documents.

1.7 Accessibility Of Text, For Copying Or Extracting For Indexing

As above, this probably shouldn't be part of the ''final form''. These formats (extracting text and index) are different transformations of the original source, *not* extenstions of the ''final form''. In fact, the term ''final form'' shouldn't be re-termed ''display form'', since the ''final form'' isn't necessarily final. If there are any extensions, they must easily be added or removed (like compression above). In many simple cases text *can* be paired to the ''display form'', however, there are many cases when it cannot be paired. The working group should strongly consider a document being delivered as several separate files (e.g., ".txt", ".ps", ".sgm", ".htm", ".btf" (binary text format), ".asc" (ASCII), ".wav") rather than a single multipurpose file. If a single file is required, then it should be constructed with a packaging tool (e.g., TAR, CPIO, AR) rather than a combined multi-purpose format. It has been demonstrated over the past 25 years that multi-purpose formats (files, tools, etc.) are much harder to

104

integrate.

1.8 Capture Structure

As above, this probably shouldn't be part of the ''final form''. This should be delivered separately. If the ''final form'' is extended, it should be separable, like compression above. It can easily be demonstrated that delivering separable (vs. combined) features makes little difference in the design and coding of viewing tools that combine these features. However, delivering separable features greatly simplfies the design and coding of viewing tools that *don't* combine these features (e.g., print-only or listen-only ''viewers'').

1.9 Support Navigation Aids

See above. Possibly the filename extensions ".toc" (table of contents) or ".ndx" (index) could be used.

1.10 External Hyperlink Capabilities

See above.

1.11 Single Or Multiple File Documents

The use of AR, TAR, or CPIO are best suited for this purpose. Likewise, for documents that span several volumes, there are several formats (e.g., ANSI X3.27 tape labels, TAR, CPIO).

1.12 Page Fidelity

This might not be desired is all cases. For example, when viewing HTML, the line breaks vary according to the screen size. If the ''final form'' is intended to be a ''display form'', then I agree that the pages should look the same. However, if the ''final form'' is intended to have many other features, than why shouldn't it have page resizing? Like all the other features, it depends upon what the user *intends* to do with the document.

1.13 Print Capability Directly From Format

The ''display form'' might actually *be* Postscript (using the document structuring conventions). We have a form that can be displayed on many printers and displays (using ghostscript -- publicly available source code) -- we should use this.

2. DOCUMENT ARCHITECTURE USED IN ANSI X3J11 AND ISO JTC1/SC22/WG14

We use the following forms for distribution of documents in X3J11.

2.1 Display Form

This is how the document looks when viewed or printed. There are three common formats: Postscript, HP PCL, and ISO 646 (with CR-LF's as newlines). We currently use Postscript, ASCII (FTP text format), and ASCII (FTP binary, i.e., binary text format) with backspaces, CR's, CR-LF's, formfeeds, but no tabs.

2.2 Text Form

This form is useful for conveying the text of a document (e.g., full text searches). For example, the UNIX command:

```
$ deroff doc.mm >text-only
```

will convert NROFF/TROFF to text format. The notion of text is the same as FTP's notion of text here. There are no control codes. Since it is in FTP text form, the use of CR's, LF's, or CR-LF's is transparent.

2.3 Structural Form

This represents the structure of the document (e.g., chapters, paragraphs, lists, etc.). Languages such as SGML, MM/MS/ME (from NROFF/TROFF), LATEX (from TEX), and others can be used. Most of these languages can separate the style (e.g., layout features) from the structural features. For example, the following command:

```
$ troff -Tpost -mm style doc.mm >postscript
```

would apply the style to the document when converting to display form. We are using SGML (with the DOCBOOK DTD) for the C Standard. Currently, we are experimenting with the DSSSL (ISO DIS 10179.2 -- Document Style Semantics and Specification Language) for layout issues.

2.4 Conceptual Form

This represents the conceptual organization of the document. For example, hypertext is used to connect concepts. We are using HTML for our WWW site. We will have the C Standard converted to HTML by 1996-10 (HTML embedded within SGML).

3. CONCLUSIONS

3.1 Summary

Our current document flow is (the steps are summarized):

(1) The source document is in SGML.

(2) The structural form is the SGML source.

(3) The conceptual form extracts an HTML-only version from the SGML source.

(4) The text form is produced by converting SGML to MM to text (via NROFF).

(5) The display form is produced by converting SGML to MM to Postscript (via TROFF).

Currently, we are in the process of changing to TEX as the intermediate form (there is publicly available source code that can be recompiled on any machine with a C compiler).

Also, we are in the process of evaluating DSSSL (and its publicly available tools) for layout specification. Although this isn't truly necessary (the C Standard uses few formatting features), we are experimenting to share with our members (who *do* use formatting features in their papers).

Like programs (source, preprocessed source, object, executable), documents are kept in several forms (conceptual, structural, text, display). Documents, like programs, can come in forms that are easy to edit (programs: source code; documents: conceptual form) or hard to edit (executables; display form).

3.2 Recommendation

The electronic publishing working group should organize its strategy around the intended use and/or workflow of documents (e.g., a dataflow diagram) rather than file formats. Once the architecture has been refined, then the specific formats can be defined. In some cases, a format might be used for two different intentions (e.g., ASCII for text form, ASCII for display form). In several cases, there might be several possible protocols for the same form (e.g., structural form: SGML, MM; display form: ASCII, ASCII + ''vt100'' codes (X3.64), Postscript).

The architecture should establish dataflow, workflow, semantics, terminology (e.g., the four ''forms''), and naming conventions (e.g., ".ps" means Postscript file). After the architecture has been determined, then the standards needs can be identified. It is important that the architecture be distributed along with the standards needs. Without the distribution of the architecture, the use of the standards will not be clear. The architecture should be verified with prototypes or existing tools. The architecture should be widely distributed inside and outside IISP.

There are interactions with other features (e.g., compression, security, chargeback, media packaging). These features should not be directly addressed, but should be able to be incorporated either internally or externally.