Information Technology

SC22/WG20 N857

ISO/IEC JTC 1 N6483

Date: 2001-07-09

Replaces:

Document Type: Summary of Voting/Table of Replies

Document Title: Summary of Voting on JTC 1 N 6404, ISO/IEC DTR 14652 - Functionality for Internalization Specification Method for Cultural Conventions

Document Source: JTC 1 Secretariat

Project Number:

Document Status: Although this ballot received a majority of approval votes, there were a significant number of negative votes and comments from national bodies. Therefore, the summary of voting is sent to SC 22 for resolution of the comments and preparation of a revised DTR text.

Action ID: ACT

Due Date:

No. of Pages: 48

Secretariat, ISO/IEC JTC 1, American National Standards Institute, 25 West 43rd Street, New York, NY 10036;
Telephone: 1 212 642 4932; Facsimile: 1 212 840 2298; Email: lrajchel@ansi.org

JTC 1 Ballot System - JTC 1 N 6483

JTC 1 Reports

Votes on a Ballot

Committee :	**JTC 1 - Information Technology**
Ballot Number :	**N6404**
Ballot Title :	**ISO/IEC DTR 14652 - Functionality for Internalization Specification Method for Cultural Convenetions**
Source:	ANSI
Distribution:	JTC 1 Members

Questions for this Ballot

Does your National Body support DTR 14652 to go forward for publication?

Answers	**Votes**
Not Yet Voted	8
APPROVAL OF THE DRAFT AS PRESENTED	10
APPROVAL OF THE DRAFT WITH COMMENTS AS GIVEN ON THE ATTACHED	0
DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	8
ABSTENTION	3

Organization	**Q.1**	**Comment**
Australia	ABSTENTION	
Austria	Not Yet Voted	
Belgium	Not Yet Voted	
Brazil	Not Yet Voted	
Canada	Not Yet Voted	
China	APPROVAL OF THE DRAFT AS PRESENTED	
Czech Republic	APPROVAL OF THE DRAFT AS PRESENTED	
Denmark	APPROVAL OF THE DRAFT AS PRESENTED	
Egypt	Not Yet Voted	

Finland	DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	In our consultation with other national bodies on this, we have become aware of the forthcoming negative US comments that we find both relevant and appropriate. We believe that it would serve no useful purpose and be utmost inefficient if we were to reformulate them by ourselves, especially since we believe that it would be difficult if not impossible to try to solve the problems inherent in this DTR with pointed, individual editing instructions.
France	DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	See Attached
Germany	DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	See Attached
Hungary	Not Yet Voted	
Ireland	DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	See Attached
Italy	APPROVAL OF THE DRAFT AS PRESENTED	
Japan	APPROVAL OF THE DRAFT AS PRESENTED	
Netherlands	APPROVAL OF THE DRAFT AS PRESENTED	
New Zealand	APPROVAL OF THE DRAFT AS PRESENTED	
Norway	APPROVAL OF THE DRAFT AS PRESENTED	
Portugal	Not Yet Voted	
Republic of Korea	APPROVAL OF THE DRAFT AS PRESENTED	
Romania	APPROVAL OF THE DRAFT AS PRESENTED	
Russian Federation	Not Yet Voted	
Slovenia	DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	See Attached
South Africa	ABSTENTION	

Sweden	DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	See Attached
Switzerland	ABSTENTION	
United Kingdom	DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	See Attached
USA	DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED (Please indicate if acceptance of these reasons and appropriate changes in the text will change your vote to approval)	See Attached

France

Please note that the acceptance of these reasons and appropriate changes in the text will change your vote to approval.

1-Audience expectation

The objective of DTR 14652 set forth in line 148 :

> [..] *that are expected to be developed for a number of programming languages*

cannot be reached because the DTR 4652 is kept compatible with POSIX:1996, as stated in its "Scope" section. POSIX:1996 architecture is not fit for a general and modern specification of cultural services, and its next revision is not expected to improve on that particular matter. Conversely, keeping POSIX compatibility will doubtlessly serve POSIX audience better, so we believe the DTR shall insists it belong to the POSIX culture. Therefore, we kindly request the line 148 to specify its audience is POSIX culture, such as :

> *The descriptions are intended to be coded in text files to be used via Application Programming Interfaces, that are expected to be developed for a number of systems which comply with ISO/IEC 9945.*

2-POSIX 200X compatibility

At the same time, every reasonable step must be taken to ensure that the DTR will accommodate POSIX 200x. We kindly request this effort is asserted or documented somewhere.

3-Multiple currency

The currency multiple value is not expected to be used because the DTR will be published after the most important dual currency period. It is expected that the solutions that will be implemented using (hopefully) POSIX until 2002-02-17 (end of dual currency for the first round of "*Euroland*" countries) will be used in future currency-switching countries, in Europe or elsewhere. In short this solution arrives too late, and is not proven to be appropriate. We kindly request it is withdrawn.

4-Iswgraph

We are not sure that the DTR allows for intelligent categories, e.g. that one can handle multiple "space" characters, like C/C++ does with iswgraph. If not so, we kindly request the DTR to do so, in particular (but not only) for multiple space.

Germany

Vote: Disapproval with comments

If the comments are satisfactorily resolved, the vote will change to approval for this TR.

For the record it should be stated that Germany will not support the transformation of this TR into an international standard even if all of these comments are resolved.

Comments:
General: The draft technical report has now reached a certain stage of maturity that might possibly make it useful for guidance in certain communities. However, it still contains a considerable number of errors and shortcomings, some of a systematic nature, that make it unsuitable for acceptance.

* There are multiple errors in the membership of LC_CTYPE classes. For example, the draft introduces two new classes that are meant to be related to the ISO/IEC 10646-1 descriptions of combining characters. However, the draft has its own, somewhat peculiar interpretation of combining characters: Simply "combining" are, quite properly, "Characters to form composite graphic symbols, such as characters listed in ISO/IEC 10646:1993 annex B.1." (l. 939f). This is, what one would intuitively understand also combining3 (i. e. combining characters allowed in a level 3 implementation of 10646, i. e. all) to mean. However, combining3 is combining - "combining2", i. e. minus those combining characters in a level 2 implementation. The terminology should be adapted accordingly, e. g. combining for all combining characters (with combining3 as an equivalent) and combining2 to mean specifically those allowed in a level 2 implementation - if combining2 is indeed needed at all.

Most ideographs are not included in the <graph> class (why?). Also, the draft includes only the repertoire of 10646 as it was in 1998 and should be extended to cover at least 10646-1:2000.

*The changes to the monetary section that are incompatible with the current POSIX.2 standard (ISO/IEC 9945-2) must be removed, in particular all cases where it has previously only been allowed to insert one value, but now a semicolon-delimited list of values. This is true in particular for the definition of multiple national currencies.

a) This breaks implementations that expect the single values defined in POSIX.2.

b) It does not specify which of the currencies should be selected, unless the "valid_from" and "valid_to" keywords are meant to be such a mechanism. In this case, the mechanism would be highly unsuitable especially for the case of the Euro where the currency co-exists over a period of time (now virtually over), and the correct currency sign is selected on a case by case basis. The Java approach of multiple locales for one language and locale, differing only with respect to a certain point -

currency in this case - is far more flexible and user friendly.

* The fixed, locale-based currency-rate must be removed, as repeatedly discussed in the past (ll 2275ff and also ll 6504ff). It is an unsuitable mechanism that will not work even for those European countries in the Euro-zone.

* The changes to the LC_TIME section (section 4.7) that are incompatible with POSIX.2 must be undone. This includes the issue of "twelve or thirteen semicolon-separated" (2574f) months, whereas previously only twelve months were allowed. Implementations that expect exactly twelve entries here will break.

* The value of the timezone keyword (2663ff) in LC_TIME is difficult to see for countries that span more than one timezone. The relevant timezone is in any case present in the TIMEZONE environment variable.

* The usefulness of the LC_XLITERATE category (section 4.9) has repeatedly been questioned. As the TR freely admits, it is suitable only to "simple transliteration based on substring substitution" (2938). There is often if not usually more than one transliteration scheme from a source script to a target script even within one culture. To hardcode one of these into a locale makes little sense. Therefore, LC_XLITERATE should be removed.

Ireland

Ireland votes NO on DTR 14652. In consultation with members in the Irish IT industry, we became aware of the forthcoming negative US comments. These comments are extensive and exhaustive, and it seems clear that this project, which has been on the books for a very long time indeed, still lacks consensus and technical accuracy. We do not feel that it would be useful to publish our own litany of what is wrong with this standard; rather, in this case, we consider the US comments to state the case quite clearly.

It is not clear that this matter ought to be standardized. It seems far more appropriate for it to be formulated and published in another medium, such as an RFC or a UTR.

Slovenia

Standards and Metrology Institute of Slovenia (SMIS) as a full member of JTC1 would like to vote "against" for the document **ISO/IEC 14652** with the folowing techical comments:

GENERAL: There is no consistency with existing practice in the technical part of the document. In particular:

OBJECTION 1
Section 4.1.4.1 comment_char (lines 652-653, and affecting the FDCC-set definition)

Current text:
"The comment character defaults to the number_sign "#". All examples in this Technical Report uses "%" as the comment character, except where otherwise noted."

Problem and Action:
ISO/IEC 9945-2:1992 (POSIX.2) uses the default comment_char, and for consistency with existing practice, this document should as well.
Change the sentence "All examples..." to "All examples in this Technical Report use the default comment character." Also, revise the FDCC-set definition.

OBJECTION 2
Section 4.1.4.2 escape_char (lines 666-667, and affecting the FDCC-set definition)

Current text:
"The escape character defaults to backslash '\". All examples in this Technical Report uses "/" as the escape character, except where otherwise noted."

Problem and Action:
ISO/IEC 9945-2:1992 (POSIX.2) uses the default escape_char, and for consistency with existing practice, this document should as well. =
Change the sentence "All examples..." to "All examples in this Technical Report use the default escape character." Also, revise the FDCC-set definition.

Sweden

We find that this Technical Report of type 1 are not up-to-date with modern internationalisation techniques. Incremental changes are unlikely to result in anything sufficiently up-to-date. We therefore suggest that this project be discontinued. A new internationalisation format report could be taken up at a later date, should resources and sufficient consensus arise.

SE 1. MAJOR:
There is no character encoding declaration for a FDCC set file itself, nor any requirement to use an encoding scheme for the universal character set (e.g. UTF-8). Instead there is essentially a limitation to POSIX so-called portable characters (a subset of ASCII), otherwise the encoding is in principle undefined ("implementation defined") and that cannot be relied upon. Therefore expressing some of the things covered by 14652, like weekday names, are needlessly cumbersome, using various kinds of character references. Instead such items should be expressed directly as the strings that one wishes to have output (or parsed).

SE 2. MAJOR, LC_CTYPE:
Draft 14652 suggests to tie character properties to locales (FDCC sets). This will surely lead to inconsistencies among locales for property assignments for the same characters. Instead haracter properties should be defined on the universal character set (UCS). Together with well defined mappings between various character encodings and the UCS one can get consistent property assignments. In particular some properties may be defined only for a subset of the UCS characters in many locales, which works very badly together with programming paradigm where all character string processing is done on UCS strings, and other encodings are handled via conversion (this is the modern approach to character processing).

SE 3. MAJOR, REPERTOIREMAP:
More than 25 pages (in small print) are devoted to a so-called repertoiremap (clause 6), with non-mnemonic arcane "names" for characters. This list of names should be removed. Instead, for these names, for the few instances really needed (like invisible characters), use the code point number (in hexadecimal). But for the majority of cases use the character itself, as mentioned in the first point above.

SE 4. MAJOR:
Many of the components formats presupposes a C-like API, using format strings with % followed by a letter. Not all systems may wish to use such format strings. Further, the character classes are insufficient for many purposes, assuming an "encoding independent" paradigm (which is assumed for standard C, but cannot be used for the most modern character encodings, i.e. the UCS encoding forms, since the UCS has many features not present in most or any legacy character encoding.

SE 5. MAJOR:
The locale (FDCC set) layout structure is very much geared towards having fixed premade locales. It's not geared towards having data in one layer and user preference selections in another layer. Instead these layers are mixed up, needlessly complicating things for users that may wish to compose their own "locale", i.e. formatting preferences.

SE 6. MAJOR:
LC_COLLATE: The format and semantics are described in 14651. Only a reference to that is to be made, no conflicting (as is presently given in DTR 14652) specification can be allowed.

SE 7. MAJOR:
Charmaps: charmaps are not in any way related to 'locales' (FDCC sets), and locales should thus never specify any charmap. Any character encoding can occur for any locale (compare XML and its 'encoding' pseudo attribute). Further the "xliterate" 'category' seems to be more related to character mapping fallbacks than to real transliteration.

United Kingdom

Technical comments

The work is still premature and does not represent any industry practice. Although some useful possibilities are documented which would benefit from further development, many of these should not be considered for development within a standard, nor in a technical report which can be referred to normatively.

In addition, there has been sustained opposition to this from various industry sources who participate in ISO/IEC JTC1/SC22.

Given the lack of consensus, this item should be withdrawn from the ISO/IEC JTC1/SC22/WG20 work programme. It may be useful for this to be developed in other fora, e.g. some Linux development groups, but it should not be developed further in ISO/IEC JTC1/SC22/WG20.

There are also errors in this draft which have not been corrected to take account of previous comments in the meetings. The editorial comments below list just a few of these.

Editorial comments

There remain errors in new tables in LC_CTYPE classes: these use different descriptions and different character groups than those defined in ISO/IEC 10646-1:2000.

In the monetary and time sections, (a) definitions of multiple currencies are introduced, which conflict with implementations which anticipate only the single values defined in the POSIX.2 standard ISO/IEC 9945-2, and (b) different definitions for the number of months, and the start day of the week, are introduced.

The LC_XLITERATE section for character transliteration does not include the corrections suggested at previous meetings of ISO/IEC JTC1/SC22/WG20. The conversions proposed are somewhat idiosyncratic, and do not represent any consensus for conversion within ISO/TC46/SC2 (Conversion of Written Languages) which develops standards on transliteration, and other alternative transliteration conventions are not catered for.

United States

OBJECTION #1
Section 4.1.4.1 comment_char (lines 652-653, and affecting the FDCC-set definition)

Current text:
"The comment character defaults to the number_sign "#". All examples in this Technical Report uses "%" as the comment character, except where otherwise noted."

Problem and Action:
ISO/IEC 9945-2:1992 (POSIX.2) uses the default comment_char, and for consistency with existing practice, this document should as well. Change the sentence "All examples..." to "All examples in this Technical Report use the default comment character." Also, revise the FDCC-set definition.

OBJECTION #2
Section 4.1.4.2 escape_char (lines 666-667, and affecting the FDCC-set definition)

Current text:
"The escape character defaults to backslash "\". All examples in this Technical Report uses "/" as the escape character, except where otherwise noted."

Problem and Action:
ISO/IEC 9945-2:1992 (POSIX.2) uses the default escape_char, and for consistency with existing practice, this document should as well. Change the sentence "All examples..." to "All examples in this Technical Report use the default escape character." Also, revise the FDCC-set definition.

OBJECTION #3
Section 4.2 LC_IDENTIFICATION (lines 698-777)

Problem:
The text defines a list of properties for an FDCC-set, and states that "All keywords are mandatory unless otherwise noted." (lines 701-702) However, at lines 728-729, it states "If information required for any of the mandatory keywords above is not available, then the corresponding string is an empty string." Further, the i18n LC_IDENTIFICATION section defined at lines 748-777 contains empty strings for six `mandatory' keywords.

This is confusing. What the text is trying to say is that certain keywords must be present, as opposed to requiring that values be assigned to certain keywords. But when most people think of "mandatory", they think of it in terms of values, not keywords. Besides, what is the rationale of requiring that certain keywords be present, but NOT requiring that they include a value? If values are not required, they are not mandatory.

Action:

Make the following changes.

1. Change the sentence "All keywords are mandatory..." to "Values must be supplied for all keywords, unless otherwise noted."

2. Add the sentence "This keyword is optional." to the description of keywords email, tel, fax, language, and territory.

3. Remove the sentence at lines 728-729 ("If information required for any of the mandatory keywords...").

OBJECTION #4
Section 4.3 LC_CTYPE (lines 787-788 and 817-821 and affecting Section 4.3.2 "i18n" LC_CTYPE category)

Current wording:
"The double increment hexadecimal symbolic ellipses ("..(2)..") works like the hexadecimal symbolic ellipses, but generates only every other of the symbolic character names. As an example. <U01AC>..(2)..<U01B2> is interpreted as the symbolic character names <U01AC>, <U01AE>, <U01B0>, and <U01B2>, in that order."

Problem:
This type of symbolic ellipses allows an FDCC-set author to save a little typing for some scripts if letters for those scripts are arranged in a code set
in uppercase/lowercase pairs. Using this type of ellipses, the author can indicate a start and end point for a range, and pick up every other entry.

The problem is that this is extremely confusing, especially considering that there already are three other types of ellipses. It will be extremely easy for authors to make mistakes, and difficult to implement and maintain all these variations. The work saved by adding this type of ellipses is overshadowed by the implementation, maintenance, readability, and potential for mistakes that it adds.

Action:
Remove lines 817-821. Remove the reference to double increment hexadecimal symbolic ellipses in lines 787-788. Change the entries in Section 4.3.2 to eliminate usage of this type of ellipses.

EDITORIAL #5
Section 4.3.1 Character classification keywords (line 834)

Problem and Action:
Grammar; change existing text to "...the interpreting system provides them if missing and accepts them silently..."

OBJECTION #6
Section 4.3.1 Character classification keywords (lines 855-857)

Current wording for digit class:
"Define the characters to be classified as numeric digits. Digits corresponding to the values 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 can be specified in groups of 10 digits,..."

Problem:
The text was not quite accurate in POSIX.2, and it definitely is not accurate here. The first sentence is copied from POSIX.2, but in that standard, *only* the portable digits 0-9 could be specified. This proposal extends the definition, but only allows decimal digits. The restriction should be spelled out.

Action:
Change the first sentence to "Define the characters to be classified as decimal digits."

OBJECTION #7
Section 4.3.1 Character classification keywords (line 867)

Problem and Action:
Incorrect class name; change "digits" to "digit".

OBJECTION #8
Section 4.3.1 Character classification keywords (lines 869-878)

Current wording for "outdigit" class:
"Define the characters to be classified as numeric digits for output from an application, such as to a printer or a display or a output text file. Digits corresponding to the values <0>, <1>, <2>, <3>, <4>, <5>, <6>, <7>, <8>, and <9> can be specified, and in ascending order of the values they represent. The intended use is for all places where digits are used for output, including numeric and monetary formatting, and date and time formatting. Only one set of 10 digits may be specified. If this keyword is not specified, the digits 0 through 9 of the portable character set automatically belong to this class, with application-defined character values..."

Problem:
This keyword as defined is insufficient for its stated use. Assume someone wants to define Roman numerals for use in dates. Since only the values 0-9 can be specified, there is no way to list the Roman numerals X, XI, and XII for the 10th-12th months. Or suppose someone wants to write Chinese monetary values. There is a single character for "ten", a single character for "hundred", and so on. To express 10, you use the "ten" character; to express 20, you use the "two" character plus the "ten" character (two 10s). The outdigit keyword does not allow for the Chinese "ten" or "hundred" (and so on) characters, and so does not fulfill the intended use for "all places where digits are used for output, including numeric and monetary..."

Action:

Remove this keyword since it does not satisfy the stated need.

OBJECTION #9
Section 4.3.1 Character classification keywords (lines 902-905)

Current wording in description of "xdigit" class:
"...If this keyword is not specified, the digits <0> through <9>, the uppercase letters "A" through <F>, and the lowercase letters <a> through <f>, automatically belong to this class, with application-defined character values..."

Problem:
As written, this is different from the POSIX.2 requirement that the xdigit class must contain the portable digits 0-9 and the portable letters A-F and a-f. This only says that if the keyword is not specified, these portable characters are included, but with this text, a person could write an xdigit class that included only Hindi digits and some subset of Greek letters, and it would be legal. This is inconsistent with POSIX.2, and therefore must be changed.

Action:
Remove the clause "If this keyword is not specified," from the sentence beginning at line 902. The revised sentence will read "The digits <0> through <9>..."

Also note that "A" in the sentence should be <A>.

OBJECTION #10
Section 4.3.1 Character classification keywords (lines 929-932)

Current wording in tolower description:
"...If this keyword is specified,
the uppercase letters <A> through <Z>, and their corresponding lowercase letter, are specified. If this keyword is not specified, the mapping is the reverse mapping of the one specified for toupper."

Problem:
The description is incorrect for what happens when the keyword is specified. This is what happens if the keyword is NOT specified. However, the sentence (if fixed) still would be unnecessary because the second sentence "If this keyword is not specified, the mapping is the reverse..." implies that <A> to <Z> will be included.

Action:
Remove the sentence on lines 929-931 ("If this keyword is specified,...")

OBJECTION #11
Section 4.3.1 Character classification keywords (lines 933-946)

(and see also Section 4.3.2 "i18n" LC_CTYPE category [class "combining" and class "combining_level3; lines 1664-1694])

Current wording for "class" class:
"Define characters to be classified in the class with the name given in the first operand, which is a string. This string only contains characters of the portable character set that either has the string "LETTER" in its description, or is a digit or <hyphen-minus> or <low-line>. The following operands are characters. This keyword is optional. The keyword can only be specified once per named class. The following two names are recognized:

```
combining           Characters to form composite graphic symbols, such
                    as characters listed in ISO/IEC 10646:1993 annex B.1.
combining_level3    Characters to form composite graphic symbols, that
                    may also be represented by other characters, such as
                    characters listed in ISO/IEC 10646-1:1993 annex B.2."
```

And also current wording from the "i18n" FDCC-set definition, lines 1664-1694:
"% The "combining" class reflects ISO/IEC 10646-1 annex B.1
% That is, all combining characters (level 2+3).

```
class           "combining" /
   <U0300>..<U036F>; <U20D0>..<U20FF>; <UFE20>..<UFE2F>;/
   <U0483>..<U0486>;<U0591>..<U05A1>;<U05A3>..<U05B9>;/
   <U05BB>..<U05BD>;<U05BF>;<U05C1>;<U05C2>;<U05C4>;<U064B>..<U0652>;<U0670>;/

<U06D7>..<U06E4>;<U06E7>;<U06E8>;<U06EA>..<U06ED>;<U0901>..<U0903>;<U093C>;/

<U093E>..<U094D>;<U0951>..<U0954>;<U0962>;<U0963>;<U0981>..<U0983>;<U09BC>;/
   ...
   <U0F97>;<U0F99>..<U0FAD>;<U0FB1>..<U0FB7>;<U0FB9>;<U302A>..<U302F>;/
   <U3099>;<U309A>;<UFB1E>
```

%
% The "combining_level3" class reflects ISO/IEC 10646-1 annex B.2
% That is, combining characters of level 3.

```
class           "combining_level3"; /
   <U0300>..<U036F>;<U20D0>..<U20FF>;<U1100>..<U11FF>;<UFE20>..<UFE2F>;/
   <U0483>..<U0486>;<U0591>..<U05A1>;<U05A3>..<U05AE>;<U05C4>;/
   <U05AF>;<U093C>;<U0953>;<U0954>;<U09BC>;<U09D7>;<U0A3C>;/
   <U0A70>;<U0A71>;<U0ABC>;<U0B3C>;<U0B56>;<U0B57>;<U0BD7>;<U0C55>;<U0C56>;/
   <U0CD5>;<U0CD6>;<U0D57>;<U0F39>;<U302A>..<U302F>;<U3099>;<U309A>"
```

Problem:
I've quoted a lot of original text here, because this is a confusing problem. I could not understand from the description what the classes were supposed to be for, so I looked at the i18n FDCC-set example.

It turns out the description and definition of the two combining classes is exactly backward. ISO 10646 defines three levels:

Level 1 -- most restrictive; shall not contain any characters listed in Annex B.1
Level 2 -- less restrictive; shall not contain any characters listed in Annex B.2
Level 3 -- least restrictive; can contain any coded character.

The members listed of the classes in the FDCC-set, however, do not

match the definitions. What is called combining_level3 is the group of characters that canNOT appear in a Level 1 or 2 implementation. What is called "combining", and described as being "all combining characters (level 2 + 3)", actually is the list of characters that canNOT appear in a Level 1 implementation.

Action:
These classes do not exist in other standards and are so ill-defined that it is impossible to say what characters are supposed to be defined in which class. Remove lines 933-946 and lines 1664-1694 from the draft.

OBJECTION #12
Section 4.3.1 Character classification keywords (lines 947-955)

Current wording in width description:
"Define the column width of characters, for example for use of the C function wcwidth(). The operands are first a list for characters, possibly using various ellipses, and semicolon separated, then a <colon>, and then the width of these characters given as an unsigned positive integer. Such width-lists separated by <semicolon> may be given for the various widths. The default value of width of characters in class "cntrl" and class "combining" is 0, else the default value of width is 1. A width for a character may be overridden by a WIDTH specification in a charmap. This keyword is optional."

Problem:
This description is very confusing. What does it mean that a "...width for a character may be overridden by a WIDTH specification in a charmap"? Does that mean if it's one thing in the charmap and another in the FDCC-set, the charmap wins? Why should width specifications be in two places?

Also, this class is quite different from other LC_CTYPE classes. For other classes, one lists which characters are in that class, or a one-to-one mapping between uppercase and lowercase. This is different; you list a group of characters, and then define what value their width is. Each character in this class can have a different value, as opposed to other classes where it simply is a Boolean function -- if you're listed, you're in.

This class is confusingly-defined, and seems out-of-place in the Boolean-oriented LC_CTYPE section.

Action:
Remove lines 947-955.

OBJECTION #13
Section 4.3.1 Character classification keywords (lines 956-973)

Problem:
The map keyword is poorly described. According to Annex A, it is supposed to provide the functionality associated with the C library function towctrans(), but that's not clear from the text here ("Define the

mapping of characters." What?).

Action:
Either remove this keyword, or rewrite the description to make it clearer that this is designed to allow mapping of one type of characters to another, related type. For example, you might want to map hiragana to katakana. Or Hindi digits to portable digits. Etc.

OBJECTION #14
Section 4.3.1 Character classification keywords (lines 975-1002)

Problem:
The mapping table of character class combinations duplicates information in POSIX.2 without adding any new data about classes included in this document.

Action:
Either remove the table completely, since the information already is available in another standard, or update it to include combination information about classes added for this document.

OBJECTION #15
Section 4.3.2 "i18n" LC_CTYPE category

Problem:
The membership of classes is inconsistent and confusing. With a few exceptions, it should match the classifications in the Unicode standard, where the classes/properties are comparable. Right now, class memberships are similar, but not identical to, comparable Unicode classes. For example:

* the digit class includes a large group of digits that Unicode also identifies as being decimal, but is missing these groups:

Myanmar (U1040..U1049)
Ethiopic (U1369..U1371)
Khmer (U17E0..U17E9)
Mongolian (U1810..U1819)
Fullwidth (UFF10..UFF19)

Why should these be omitted, when the others are included?

* the space class includes many of those that Unicode identifies as being space, but is missing:

U00A0 -- No-Break Space
U2007 -- Figure Space
U202F -- Narrow No-Break Space

Note that this class also has several control characters, like <tab> and <carriage-return>, that Unicode does not consider part of the space class. However there is much existing practice on POSIX-based systems for including those controls, so it is understandable why they are here.

* the punct class includes some, but not all, characters that Unicode identifies as being punctuation. For example:

   + it includes U2030..U2046, which are in the Unicode general punctuation block, but omits

      U2048 -- Question Exclamation Mark
      U2049 -- Exclamation Question Mark
      U204A -- Tironian Sign Et
      U204B -- Reversed Pilcrow Sign

      These also are in the general punctuation block.

   + it includes the currency symbols in the range U20A0..U20AA, but omits these other currency symbols in the same block:

      U20AB -- Dong Sign
      U20AC -- Euro Sign
      U20AD -- Kip Sign
      U20AE -- Tugrik Sign
      U20AF -- Drachma Sign

   + unlike Unicode 3.0, it includes most of the "Letterlike Symbols" from the range U2100..U213A in the punct class. This includes characters like U210B (Script Capital H), U2115 (Double-Struck Capital N), etc., but omits those that happen to have the word "LETTER" in their name; e.g.,

      U210C -- Black-Letter Capital H
      U2111 -- Black-Letter Capital I

      This range also omits U2139 (Information Source), and U213A (Rotated Capital Q), which are also in this Letterlike Symbols block.

      It's not clear why any in this range are included in punct, but the particular subset of characters listed is even more confusing.

There are many more differences between this i18n FDCC-set and Unicode, but the point is that the differences exist. This document should use the Unicode values where they exist instead of inventing another group of classifications that differ in dozens of small ways.

Action:
Revise the membership of all classes to match the lists Unicode provides, where they exist. HOWEVER, in the few cases where the common practice in POSIX systems differs from Unicode (for example, including some control characters in the space class), retain that existing practice for members of the portable character set.

Note, too, that 14652 defines some classes for which there are no matching Unicode properties. Obviously, in these cases, the i18n FDCC-set cannot match Unicode.

Section 4.4 LC_COLLATE

This is a placeholder for the content of Section 4.4 (LC_COLLATE).
See TECHNICAL #61, TECHNICAL #62 and TECHNICAL #63 later in this document.

OBJECTION #16
Section 4.5 LC_MONETARY (entire section)

Problem:
This section includes multiple keywords that were defined in POSIX.2, but it changes their definitions in such a way that existing applications would be invalid. This is incorrect. The changes allow the rules for multiple currencies to be specified in existing keywords, but in POSIX.2, only rules for single currencies can be defined.

While the need to handle multiple currencies is real, the method defined here is significantly different than what has been done when other LC_ categories have had to be extended. When expanding LC_TIME to allow for multiple calendars, new keywords were added (era, era_year, etc.), rather than simply tacking new entries on to the end of existing keywords.

Consider the previously existing LC_MONETARY keyword currency_symbol. It is defined in POSIX.2 as "The string that shall be used as the local currency symbol," while here it is defined as "One or more strings separated by semicolons that are used as the local currency symbol." (lines 2293-2294). Assume I'm defining French currency and the euro. I might have something like this:

```
currency_symbol   "<F>";"<euro>"
```

However, the description of this category no longer is correct -- these are not strings "that are used as the local currency symbol". That implies the two strings are synonyms for each other. The reality is that these are strings that represent different currencies used for this locale. They should not be glommed together in one keyword. It would be more accurate to separate these (and all other keywords that in this draft can take multiple values) into something like

```
currency_symbol         "<F>";
alt_currency_symbol     "<euro>";
```

As defined in this draft, it is not clear how application programs parse or use these values. Existing implementations request *the* currency symbol and use it to format values. What would happen to a previously conforming application if it requested the (single) currency_symbol value, but an array of strings was returned? Lines 6509-6510 of the rationale state:
"Also the same application call can be made to be valid for countries with a single currency and countries with dual currencies." That's only true if the application is expecting one *or more* values. Existing applications expect exactly one value for most of these keywords.

Now, suppose an application is rewritten to allow for multiple currency symbols. Now what? What rules does it use to decide which currency_symbol value it should use to format a monetary quantity? If the section were designed so that the existing definitions had not changed, but alt_* keywords were added when needed, an application could request currency_symbol when formatting national currency values, and alt_currency_symbol when formatting euros (or another alternate currency).

Also, *because* this section allows multiple currencies to be specified, there is an implied tie between keywords. If currency_symbol includes French francs and euros (in that order), frac_digits, ps_cs_precedes, etc., must also specify the rules for francs and euros in the SAME order. The valid_from keyword attempts to explain this dependency, but the wording is very confusing and not restricted to that keyword.

Moving to other keywords, there are a new set of int_* keywords. Under POSIX.2, there were only two such keywords -- int_currency_symbol and int_frac_digits. They were for formatting monetary values using the international currency strings (e.g., "USD " rather than "$" for the U.S. dollar; "DEM " rather than "DM" for the German mark; etc.). Under POSIX.2, quantities that used the international currency string and those that used the local currency symbol used the same values for keywords such as p_cs_precedes, p_sep_by_space, etc. Annex A says these have been added to accommodate "differences between local and international formats." For example?

At the end of this section, the "i18n" FDCC-set does nothing to illuminate the many new keywords and revised definitions of existing keywords. Since attempted support for multiple currencies is the reason for the many changes and additions to this section (as compared to POSIX.2), an example in this section that illustrates how multiple currencies might actually be specified must be provided. There is an example in the rationale section, but the information needs to be available here.

See below for additional comments on specific keywords.

Action:
Restore the original definitions of keywords that exist in the LC_MONETARY section of POSIX.2. Add new keywords for defining alternate currencies. Remove the additional int_* keywords, unless a concrete rationale with examples of real differences between local and international formats, is provided. Add an example that shows how to specify multiple currencies.

EDITORIAL #17
Section 4.5 LC_MONETARY (lines 2250-2252)

Current wording:
"...Keywords that are not provided, string values set to the empty string "", or integer keywords set to -1, are used to indicate that the value is unspecified, and then no default is implied."

Problem:
This wording is unclear.

Action:
To follow POSIX.2 more closely, revise the sentence as follows: "Keywords that are not provided, string values set to the empty string (""), or integer keywords set to -1 shall be used to indicate that the value is not available. No defaults are implied."

OBJECTION #18
Section 4.5 LC_MONETARY (lines 2258-2268)

Current wording of valid_from keyword:
"One or more strings separated by semicolons, representing a Gregorian date in the form "YYYYMMDD" according to ISO 8601, specifying the beginning date (inclusive from the beginning of day local time) of the validity of a currency. The position of the string in the list corresponds to the position of operands in other keywords in the LC_MONETARY category. The currencies should be ordered in terms of validity dates, and for each validity period with the currency that the amounts are stored in first. If not specified, it is taken to be an implementation-defined beginning of time. This keyword is optional."

Problem:
This wording is unclear and confusing. I think *part* of what this is trying to say is:

"One or more strings, separated by semicolons, of Gregorian dates in the form "YYYYMMDD" that specify the date on which a currency became or becomes valid. Dates are inclusive from the beginning of the day local time....If not specified, the value of this keyword is an implementation-defined beginning of time. This keyword is optional."

The earlier overall objection to this section notes that information about dependencies on the order of values is not restricted to this keyword. Thus, the sentence "The position of the string..." should not appear in this description.

There is no reason to mention ISO 8601 here; specifying the YYYYMMDD order is sufficient.

The sentence "The currencies should be ordered in terms of validity dates..." is unclear; I have no idea what it means.

Action:
Revise the text as recommended, rewrite the sentence about "The currencies should be ordered...", and add an example to show how this might be defined.

OBJECTION #19
Section 4.5 LC_MONETARY (lines 2269-2274)

Current wording of the valid_to keyword:
"One or more strings separated by semicolons, representing a Gregorian date in the form "YYYYMMDD" according to ISO 8601, specifying the end date (inclusive to the end of day local time) of the validity of a currency. If not specified, it is taken to be an implementation-defined end of time. This keyword is optional."

Problem:
The current wording is unclear, and the default value is inappropriate since not all systems define an end of time.

Action:
Rewrite as follows:

"One or more strings, separated by semicolons, of Gregorian dates in the form "YYYYMMDD" that specify the last day on which a currency was or will be valid. Dates are inclusive to the end of the day local time. This keyword is optional."

OBJECTION #20
Section 4.5 LC_MONETARY (lines 2275-2292)

Current wording:
"one or more pairs of integers separated by a <semicolon> specifying the fixed conversion rate between the current currency (determined by the parameter number) and the first currency that is valid, determined by a date provided by the application. If the currency is not the first valid currency for the period in question, the first integer is for multiplying the first valid currency, and the second for dividing this result to get the amount in the current currency. The currency to be the current currency is selected by the application from the date applicable and the currency number (first, second, third etc valid currency at that date); and whether domestic or international formatting is used is also determined by the application. Each pair of integers are separated by a <slash>. The default value is "1/100". This keyword is optional..."

Problem:
The description of the conversion_rate keyword is incomprehensible. However, an example in the rationale section shows this definition for Deutsch marks and euros:

```
conversion_rate         1;              195/100
```

From this, it appears that the first value for conversion rate should be 1, because it is the "primary" currency, and the value for the second currency should be the true conversion value. However, this example does not match the current keyword text. Note, for example, that an entry is supposed to be "one or more *pairs* of integers", but that the first value in the example is a single integer.

It also is not clear that conversion rates should be in locales, since they often change over time. The fact that euro conversion rates are fixed in relation to certain national currencies is a specific instance of currency rules, but is not applicable around the world.

Action:
Remove this keyword. The description is not clear, the example does not match the description, and it addresses a euro-specific feature, as opposed to being generally applicable to multiple currencies and locales. Further, since previous recommendations are to define different currencies in separate keywords, it would not be consistent to continue defining rules for multiple currencies in conversion_rate.

OBJECTION #21
Section 4.5 LC_MONETARY (multiple keyword entries)

Current wording at end of all non-optional keywords:
"This keyword is specified, unless the "copy" keyword is used."

Problem:
The multiple appearances of this sentence all are unnecessary. The description of the "copy" keyword states that if it "...is specified, no other keyword is specified." Thus, it is redundant to spell out the restriction about the "copy" keyword at the end of the other keywords. It also is inconsistent. Keyword descriptions in other sections of this draft do not include the redundant sentence.

Action:
Remove the sentence at lines 2317-2318, 2320-2321, 2323-2324, 2328-2329, 2333-2334, 2339-2340, 2344-2345, 2350-2351, 2367, 2384, 2392-2393, and 2397-2398.

OBJECTION #22
Section 4.7 LC_TIME (lines 2540-2543 and 2547-2551)

Current wording in abday and day keyword descriptions:
"... The first string is the [abbreviated|full] name of the day corresponding to the first day of the week (default Sunday), the second the [abbreviated|full] name of the day corresponding to the second day of the week (default Monday), and so on."

Problem:
This wording implies that the first day of the week is locale-specific, and that the %a and %A descriptors may produce the locale-equivalent of "Sunday" if Sunday is defined as the first day of the week, *or* the locale-equivalent of "Monday" if Monday is defined as the first day of the week, etc. This differs from the existing POSIX.2 definition and the descriptions in ISO C for the keywords and the meaning of the format descriptors. In the other standards, abday, day, %a, and %A all are defined in terms of a week that begins on Sunday.

Of course, many locales use a week that begins on Monday, and it is

understandable that some want to support this within abday, day, and the format descriptors. But this is an incompatible change with existing practice that will break existing implementations. Further, support for Monday-first locales already exists with the %u, %V, and %W format descriptors.

Action:
Revise the text at 2540-2543 as follows: "The first string is the abbreviated name of the day corresponding to Sunday, the second the abbreviated name of the day corresponding to Monday, and so on."

Revise the text at 2547-2551 as follows: ""The first string is the full name of the day corresponding to Sunday, the second the full name of the day corresponding to Monday, and so on."

OBJECTION #23
Section 4.7 LC_TIME (lines 2552-2567)

Current wording for week keyword:
"Is used to define the number of days in a week, and which weekday is the first weekday (the first weekday has the value 1), and which week is to be considered the first in a year. The first operand is an integer specifying the number of days in the week. The second operand is an integer specifying the Gregorian date in the format YYYYMMDD, and it specifies a day that is a first weekday (all other first weekdays may then be calculated by adding or subtracting a whole multiple of the number of days in the week as specified with the first operand). The third operand is an integer specifying the weekday number to be contained in the first week of the year. The third operand may also be understood as the number of days required in a week for it to be considered the first week of the year. If the keyword is not specified the values are taken as 7, 19971130 (a Sunday), and 7 (Saturday), respectively. ISO 8601 conforming applications should use the values 7, 19971201 (a Monday), and 4 (Thursday), respectively. This keyword is optional."

Problems:
There are multiple problems with this description.

1. There is no need to define the number of days in a week, because the seven-day week is common to all major calendars.

2. The description says this keyword defines "...which weekday is the first weekday (the first weekday has the value 1)" which is confusing but probably is supposed to define which day of the week is considered the first (for example, Sunday is the first day of the week in some cultures, while Monday is in others). Assuming this interpretation is correct, the second operand here is ill-defined to meet this requirement. It requires picking a random date that falls on the first day of the week for this FDCC-set. In this example, November 30, 1997 falls on a Sunday, so it is the value used for locales that have a Sunday-first rule. Implementors then are required to calculate ALL other first weekdays (before and after) from the randomly chosen date. This is hogwash.

3. The description further says the keyword defines "...which week is to be considered the first in a year." It is more accurately defined later in the description as "the number of days required in a week for it to be considered the first in a year." The first definition is unclear and should be changed.

Action:
Remove the operand for the number of days in a week. Remove the operand that defines a date of a (random) first weekday. Change the description of the keyword to be defining "the number of days required in a week for it to be considered the first in a year."

OBJECTION #24
Section 4.7 LC_TIME (lines 2569 and 2574)

Problem:
The descriptions of the abmon and mon keywords say they consist of "twelve or thirteen" month names. POSIX.2 and ISO C only support twelve-month calendars, and existing implementations will break if this is changed.

Action:
Change the descriptions of the keywords to say the operands consist of twelve month names, not "twelve or thirteen."

OBJECTION! #25
Section 4.7 LC_TIME (timezone section; lines 2663-2757)

Problem:
It is completely inappropriate to specify timezone information in a FDCC-set. The draft says this is for specifying cultural conventions, but timezones can cross national boundaries and many time zones can exist within a single country. For countries like the U.S., Canada, Russia, Australia, and others that span many time zones, there is no way to determine which time zone to include in an FDCC-set, or, if multiple zones are included, how to figure out which one to use in what area.

As the draft notes, the TZ (timezone) environment variable already exists for specifying time zone information. It absolutely does not belong within a locale or FDCC-set.

Action.
Remove lines 2663-2757.

EDITORIAL #26
Section 4.7 LC_TIME (line 2767)

Problem:
Table 3 is called "Escape sequences for the date field", but all

other text calls these values "field descriptors".

Action:
Change "Escape sequences" to "Field descriptors".

OBJECTION #27
Section 4.7 LC_TIME (line 2780)

Current wording for the %F descriptor:
"The date in the format YYYY-MM-DD (ISO 8601 format)."

Problem:
Multiple other places in this draft describe "ISO 8601" format as YYYYMMDD.

Action:
Make all references to ISO 8601 consistent.

OBJECTION #28
Section 4.7 LC_TIME (lines 2781-2782)

Current wording for the %g and %G descriptors, respectively:
"Week-based year within century, as a decimal number (00-99).
Week-based year with century, as a decimal number (for example 1997)."

Problem:
There is no explanation of how a "week-based year" differs from any other year. The existing %y and %Y descriptors specify the year within a century, and the year with century, so there is no need for these new descriptors.

Action:
Remove lines 2781-2782.

OBJECTION #29
Section 4.7 LC_TIME (line 2787)

Current wording for the %m descriptor:
"Month, as a decimal number (01-13)."

Problem:
As described previously, existing implementations support a 12-month calendar.

Action:
Change the text as follows: "Month, as a decimal number (01-12)."

OBJECTION #30
Section 4.7 LC_TIME (lines 2819-2826)

Current wording:
"NOTE: %g, %G and %V give values according to the ISO 8601 week-based year. In this system, weeks begin on a Monday and week 1 of the year is the week that includes 4th January, which is also the week that includes the first Thursday of the year, and is also the first week that contains at least four days in the year. . . If the 29th, 30th or 31st January is a Monday, it and any following days are part of week 1 of the following year. Thus, for Tuesday 30th December 1997, %G is replaced by 1998 and %V is replaced by 1."

Problem:
The month name in one example is wrong. The sentence should read "...If the 29th, 30th, or 31st of December is a Monday,..."

Also, since an earlier objection recommends removing %g and %G, this text should remove references to the descriptors, too.

Action:
Revise the text as indicated.

EDITORIAL #31
Section 4.8 LC_MESSAGES (lines 2931-2932)

Current wording:
"Note: This uses regular expression syntax with brackets ([]) to for example specify the both <+> and <1> is allowed as an affirmative answer."

Problem:
Grammatically incorrect sentence that doesn't say what it means to say. Inconsistent use of symbolic names. Also, since the definitions of yesexpr and noexpr say they are "extended regular expression[s]", it is not necessary to repeat that in the note.

Action:
Rewrite the text as follows: "For yesexpr, this specifies that either <plus-sign> or <one> is considered an affirmative answer. For noexpr, the supported negative responses are defined as <hyphen-minus> or <zero>."

OBJECTION #32
Section 4.9 LC_XLITERATE (lines 2934-3047)

Problem:
The ability to transliterate characters from one writing system and/or language to another is something users might think of as a "wow, cool" bit of functionality. However, this is an extremely complex problem. The keywords and syntax defined in this section are completely inadequate to handle this problem, so this section should be removed from the document.

Consider the example provided and the way it is described to work. Of course, the example is not intended to be a complete functioning transliteration section, but it raises enough questions to point out how inadequate this proposal is.

Current wording:
[begin]
"4.9.3 Example of use of transliteration

```
    LC_XLITERATE
    include "de_DE";"de_repmap"
    default_missing <?>
    translit_ignore <U3200>..<UFAFF>
    <ae>       <a:>;<e*>;"<a><e>";"<e>"
    <s>        <s*>;<s=>
    "<K><O>"   <KO>
    END LC_XLITERATE
```

...
The "include" keyword specifies that the FDCC-set "de_DE" is copied and that the repertoiremap "de_repmap" is used to define the symbolic character names in the FDCC-set "de_DE".
...
The first transliteration statement defines a number of transliterations for the LATIN LETTER AE, including into LATIN LETTER A WITH DIAERESIS, GREEK LETTER EPSILON, the two Latin letters A and E, and finally the LATIN LETTER E.
The second transliteration statement defines transliteration of the LATIN LETTER S into GREEK LETTER SIGMA, and CYRILLIC LETTER ES.
The third transliteration statement transliterates the two Latin letters K and O into the Japanese Hiragana character KO."
[end]

Start with the "include" keyword. The example shows including the de_DE FDCC-set, and according to the keyword description this is "the name of the FDCC-set...to transliterate from." So the plan here is to transliterate from German into something else. But what part or parts of a FDCC-set are supposed to be included here? The entire FDCC-set, with all sections as defined in 14652? If so, what purpose does it serve to include LC_CTYPE, LC_COLLATE, etc., sections here? If not, what exactly from the de_DE set is supposed to be included? There is no information in the LC_XLITERATE section to explain this.

Under what circumstances might users define in a locale (FDCC-set) that they want to transliterate from German? Suppose this excerpt appears in a Japanese FDCC-set. It's easy to imagine users wanting to transliterate from Japanese to a number of other writing systems and/or languages. But under this design, a finite set of transliterations would have to be hard-coded into each FDCC-set, seriously limiting users' choices.

This is an operation that should be like iconv -- users specify, independent of current configurations, what they want to convert from, and what they want to convert to. Hard-coding a set of instructions is unnecessarily restrictive.

The include keyword also specifies the "de_repmap". The keyword definition says the repertoiremap is "to be used for the definition of the transliteration statements." What does that mean? That it defines the list of symbolic characters the German FDCC-set includes? If so, now what? If we continue assuming this is a Japanese FDCC-set, what

if there is conflict between the symbolic names in the two sets?

Now consider the sample transliteration statements themselves.

```
    <ae>       <a:>;<e*>;"<a><e>";"<e>"
    <s>        <s*>;<s=>
    "<K><O>"   <KO>
```

The first converts from <ae> to any of four different possible characters. (It's curious why some are in quotes, but others are not.) According to precedence rules, if the first possibility exists in the target set, that is how <ae> is transliterated; if not, the next one is tried, and so on. Now, it's hard to imagine a circumstance under which an <ae> would be present but not the first-possibility <a-diaeresis>, but assume it isn't. The second choice listed here is a Greek <epsilon>. Thus, according to this, I'm transliterating Latin characters into other Latin characters, but if they're not available, I'll choose Greek next. Under what circumstances might someone choose such a transliteration?

Of course, the fourth possibility listed is completely superfluous, because all FDCC-sets are required to support the portable characters. Both <a> and <e> from the third choice are in the portable set, so one would never get to the fourth choice.

This second example line converts from a Latin <s> to either a Greek <sigma> or a Cyrillic <es>. This seems to assume that you know what you're converting from, but have no idea what you want to convert to, and so are allowing any potential match. How, then, does a user prevent getting a result that mixes Latin, Greek, Cyrillic, and who-knows-what-else? Once again continuing the Japanese example, many Japanese encodings include Greek and Cyrillic characters.

The third example shows Latin <K> and <O> being converted to Hiragana <KO>. What if the source language/writing system can pronounce a string in multiple ways? For example, consider English "through", "bough", "rough". This syntax seems to assume a one-to-one mapping between substrings and a target phonetic character.

Action:
The keywords and syntax in this section are inadequate to handle transliteration. Remove lines 2934-3047.

EDITORIAL #33
Section 4.10 LC_NAME (lines 3072-3077, 3094)

Problem:
Inconsistent wording; what are called "field descriptors" elsewhere in this document are called "escape sequences" here.

Action:
Change all instances of "escape sequences" to "field descriptors."

OBJECTION #34
Section 4.10 LC_NAME (line 3080-3081)

Current wording for %g and %G:
"First given name.
First given initial."

Problem:
The descriptions are European and North American-centric in assuming a position of a given name. Perhaps the description should be "Primary given name?"

Also, what qualifies as an "initial"? Any single character? Any single-byte character? Any single Latin character? Some explanation must be provided.

Action:
Remove assumptions about name position. Add information somewhere in the section about what qualifies as an "initial."

OBJECTION #35
Section 4.10 LC_NAME (line 3082)

Current wording for %l:
"First given name with Latin letters."

Problem:
What is the rationale for having a descriptor of *first given names*, and only first given names, to be transcribed into Latin letters??

Action:
Remove this descriptor and line 3082.

OBJECTION #36
Section 4.10 LC_NAME (line 3084-3085)

Current wording for the %m and %M descriptors:
"Middle names.
Middle initials."

Problem:
The descriptions are European and North American-centric in assuming that additional given names are "middle" names. Also, while other field descriptors here take a single value, these are described such that they could contain multiple names/initials. Thus, it appears multiples would be treated as a unit. For example, if someone has three given names -- Mary Laura Grace -- it appears the value of %m would be "Laura Grace" rather than "Laura" and "Grace". But most people treat each name as a separate entity. It makes more sense to have a single name in each format descriptor, and to use multiple %m's, if needed.

See previous objection about the definition of "initial".

Action:
Change the descriptions to "Additional given name" and "Initial for additional given name."

OBJECTION #37
Section 4.10 LC_NAME (line 3086)

Problem:
The format descriptor %p is described as "Profession." What does this mean and why does it appear in something that is described as being for "addressing a person; e.g., in a postal address or in a letter"? What kinds of values are expected here? Software engineer? Human Resources representative? Journalist? Garbage collector? Truck driver? Training coordinator? All or some of these? How might these be used within a postal address? Within a letter?

Action:
If there is a legitimate need for this field, add that information. Otherwise, remove the descriptor.

OBJECTION #38
Section 4.10 LC_NAME (line 3100-3103)

Current wording:

```
  % This is the ISO/IEC TR 14652 "i18n" definition for
  % the LC_NAME category.
  name_fmt    "<U0025><U0070><U0025><U0074><U0025><U0067><U0025><U0074>/
  <U0025><U006D><U0025><U0074><U0025><U0066>"
```

Problem:
Since few people have ASCII values memorized, add a comment that explains this name_fmt specifies %p%t%g%t%m%t%f, which is Profession, First (Primary?) Name, Middle (Additional?) Name, Family Name.

However, remove %p (Profession)...

Action:
Make the recommended changes.

OBJECTION #39
Section 4.11 LC_ADDRESS (lines 3108-3110 and 3125-3137)

Current wording:
"The LC_ADDRESS category defines formats to be used in specifying a location like a person's living or office, for use in a postal address or in a letter, and other items related to geography...."

Problem:
First, there is the wording problem of the phrase "...specifying a location like a person's living or office,..." What is a person's living? This

probably should be "...specifying a location like a person's home or office,..."

Second, given this description of the LC_ADDRESS section, why are there four keywords for identifying natural language? While there is justification for a locale or cultural file to include natural language information, it is out-of-place in the LC_ADDRESS section. The natural language information does not "define formats for use in specifying a location...or other items related to geography."

Action:
Reword the sentence at lines 3108-3110. Remove lines 3125-3137.

EDITORIAL #40
Section 4.11 LC_ADDRESS (entire section)

Problem:
This section uses the term "escape sequences" for what it called "field descriptors" elsewhere in the draft. "Field descriptor" is the term POSIX.2 uses, and this draft should consistently use it as well.

Action:
Change all occurrences of "escape sequence" in this section to "field descriptor" to be consistent with most earlier sections.

OBJECTION #41
Section 4.11 LC_ADDRESS (lines 3115-3120)

Current wording for postal_fmt keyword:
"Define the appropriate representation of a postal address such as street and city. The proper formatting of a person's name and title is done with the "name_fmt" keyword of the LC_NAME category. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain escape sequences defined below."

Problem:
Most postal addresses include the name of the addressee, but from this description and from the listed field descriptors, name formatting is not described here. That seems to mean users should specify name_fmt information LC_NAME and address-information-without-names in LC_ADDRESS. The two cannot be mixed because each uses the same descriptors to mean different things -- e.g., %f means family name in LC_NAME, but firm name in LC_ADDRESS; %S means salutation in LC_NAME, but state, province, or prefecture in LC_ADDRESS.

How are the two values from separate sections tied together without causing collisions?

Action:
Explain in this section how to add an addressee's name to an address.

OBJECTION #42
Section 4.11 LC_ADDRESS (lines 3123-3124)

Current wording of country_post keyword:
"The operand is a string with the abbreviation of the country, used for postal addresses, for example by CEPT-MAILCODE."

Problem:
What is CEPT-MAILCODE? Is it the only abbreviation allowed, or are other? If others are allowed, how does a user identify the abbreviation in use?

Action:
Either explain what CEPT-MAILCODE is, or remove the reference to it. If it is retained, explain either how to identify the abbreviation system in use, or that there is no way to identify which abbreviation system is being used.

OBJECTION #43
Section 4.11 LC_ADDRESS (lines 3145-3163)

Current wording for selected field descriptors:
"%a C/O address.
...
%h House number or designation.
%N If any graphical characters have been specified then an end of line is made.
%t If the preceding escape sequence resulted in an empty string, then the empty string, else a <space>.
%r Room number, door designation.
%e Floor number.
%C Country designation, from the <country_post> keyword.
%l Local township
...
%c Country."

Problems:
First, it is not clear which descriptors, if any, are restricted to holding numbers only. Usually, a description with the word "number" in it would be assumed to be numeric only, but addresses that have a floor number in them tend to be something like "2nd floor" rather than a simple number, and a house number may include other characters along with numbers. If any of these are restricted to numeric values, that should be spelled out.

Second, some descriptions are inadequate. Specifically:

%a -- what is a C/O address? In English, this is "in care of," and it identifies a person, not an address. And earlier objections note that people's names can't be included in LC_ADDRESS because of the overlap between LC_NAME and LC_ADDRESS field descriptors. So what is intended for this field?

%N -- it would be clearer to say "Insert an end-of-line if the previous

descriptor's value was not an empty string; otherwise ignore."

%t -- what does this mean? Suppose the preceding descriptor was %f,
and there was no value for it. This says do nothing. What purpose
does that serve?

%r -- can this include all characters or just numeric?

%l -- How does this differ from %T?

%c -- Is this value taken from the country_name keyword? If so, that should
be listed here.

Action:
Make the recommended changes or add more information to explain the
intention of a given field descriptor.

OBJECTION #44
Section 4.11 LC_ADDRESS (lines 3174-3184)

Current wording:
" LC_ADDRESS
 % This is the ISO/IEC TR 14652 "i18n" definition for
 % the LC_ADDRESS category.
 %
 postal_fmt   "<U0025><U0061><U0025><U004E><U0025><U0066><U0025><U004E>/
 <U0025><U0064><U0025><U004E><U0025><U0062><U0025><U004E><U0025><U0073>/
 <U0020><U0025><U0068><U0020><U0025><U0065><U0020><U0025><U0072><U0025>/
 <U004E><U0025><U0043><U002D><U0025><U007A><U0020><U0025><U0054><U0025>/
 <U004E><U0025><U0063><U0025><U004E>"
 END LC_ADDRESS"

Problem:
Once again, most of us don't have ASCII memorized, so there should be a
comment that explains what has been defined for this keyword. Currently,
it is:
"%a%N%f%N%d%N%b%N%s<space>%h<space>%e<space>%r%N\
%C<hyphen-minus>%z<space>%T%N%c%N"

Even this is very cryptic, so here is more information with all "%N" values
converted to <newline>, and all <hyphen-minus> and <space> characters
indicated:

c/o <newline>
firm name <newline>
department name <newline>
building name <newline>
street or block name <space> house number <space> floor number <space> room
number <newline>
country_post value <hyphen_minus> zip/postal code <space> town/city <newline>
country <newline>

Here's an example of a fictional address using this format:

```
c/o
General Electric
Consumer Products Division
Building 52
Lightbulb Road 110 2 57
USA-44555 Chicago
United States of America
```

Given the confusion about the %a (c/o address) descriptor, the sample value here is simply a place-holder. This also assumes that house, floor, and room number values must be numeric only, thought that may be incorrect.

While it certainly is true that addresses are culture-specific, and no one format will satisfy all, the "i18n" value here matches the postal_fmt value in the sample Danish FDCC-set later in this draft. It appears, then, that this format matches Danish conventions.

It's not clear the listed order is appropriate for an international standard. For example, using the field names defined here, in the U.S. the order generally is:

```
<addressee>             //not defined in LC_ADDRESS
<department>
<firm>
<building>              //uncommon
<house number> <street name>
<floor number> <door number>  //uncommon
<town or city>
<state or province> <zip/postal code>
<country>
```

The fact that the existing postal_fmt lists country in two different ways, does not include a value for state/province, and puts the town/city after country and zip/postal code makes this unsuitable for U.S. addresses. Of course, the goal is not to define U.S. addresses, but it's not clear whether the value listed is appropriate for a significant number of users from other countries.

Action:
Research whether the listed postal_fmt value is appropriate for a significant percentage of the world community. If not, revise the value.

Regardless of whether postal_fmt changes, add a comment explaining what the value is (all the descriptors plus an explanation of them).

EDITORIAL #45
Section 4.12 LC_TELEPHONE (entire section)

Problem:
This section uses the term "escape sequences" for what it called "field descriptors" elsewhere in the draft. "Field descriptor" is the term POSIX.2 uses, and this draft should consistently use it as well.

Action:
Change all occurrences of "escape sequence" in this section to "field descriptor" to be consistent with most earlier sections.

OBJECTION #46
Section 4.12 LC_TELEPHONE (lines 3212-3216)

Current list of field descriptors:
"%a area code without prefix (prefix is often <0>).
%A area code including prefix (prefix is often <0>).
%l local number.
%c country code
%C alternative carrier service code used for dialing abroad"

Problem:
These field descriptors are ambiguously described, and it's not clear they are adequate for specifying telephone numbers. Specific problems include:

* When the field descriptors contain numeric values, are those values restricted to the portable digits, or can they contain other decimal digits? Either way, this information needs to be included.

* What is the "prefix" that %a and %A mention? There is no description of it.

* Is %l restricted to numeric content only, or can it contain characters users commonly use to make local phone numbers more readable? For example, if the local number is 4561234, could %l contain only "4561234", or could it contain "456-1234"? If it could contain the latter, how does one define where format characters should be included? (Formatting conventions are culture-specific.) If it can only contain numbers, this is inadequate, because local phone numbers almost always include some non-numeric characters to improve readability.

* There needs to be more information about what the "alternative carrier service code" is. It's not clear whether it is useful, since there's nothing to explain what it is.

* What about extensions? Some local phone numbers have extensions to them (e.g., 434-1212 x97), but no extension field is provided here.

Action:
Add information about prefix and alternative carrier service codes. Add a field descriptor for extensions. Add information about numeric restrictions, or lack thereof. Add information about formatting local numbers.

OBJECTION #47
Section 4.12 LC_TELEPHONE (lines 3221-3227)

Current wording:

```
" LC_TELEPHONE
 % This is the ISO/IEC TR 14652 "i18n" definition for
 % the LC_TELEPHONE category.
 %
 tel_int_fmt    "<U002B><U0025><U0063><U0020><U002B><U0061><U0020><U002B>/
 <U006C>"
 END LC_TELEPHONE"
```

Problem:
As before, most people have not memorized ASCII, so there needs to be a comment that explains what this represents. A comment might, in fact, have helped bring to light that this format contains two errors. It currently is defined as:

```
+%c<space>+a<space>+l
```

Thus, two field descriptors are missing the required leading "%" signs. To match what the author presumably intended, the actual definition should be:

```
+%c<space>+%a<space>+%l
```

and lines 3225-3226 would be:

```
 tel_int_fmt    "<U002B><U0025><U0063><U0020><U002B><U0025><U0061>/
<U0020><U0025><U002B><U006C>"
```

However, consider the output of a telephone number using this format. It could be:

```
+1 +212 +5551212  //assumes %l cannot contain formatting characters
+44 +91 +12-34-56 //assumes %l can contain formatting characters
```

Many telephone numbers in "international format" use the <plus_sign> to designate the country code, but we are not aware of any that use the <plus_sign> before the area code and local number.

Action:
Remove the <plus_sign> before %a and %l in the format. Add the <percent_sign> characters before the "a" and "l" format descriptors. Add a comment explaining what tel_int_fmt designates.

OBJECTION #48
Section 5. CHARMAP (lines 3232-3233)

Current wording:
"A character set description may exist for each coded character set supported by an application. This text is referred elsewhere in this Technical Report as a charmap."

Problem:
This does not make sense. Applications should not support specific coded character sets; implementations like OSes and desktops usually provide such support. Also "This text is referred elsewhere..." is incoherent.

Action:
Reword the paragraph as follows:
"A character set description file may exist for each coded character set supported by the implementation. This file is referred to elsewhere in this Technical Report as a charmap."

OBJECTION #49
Section 5. CHARMAP (lines 3267-3276 and other affected lines throughout)

Current wording for <escape_char> and <comment_char>, respectively:
"The escape character used to indicate that the characters following is interpreted in a special way, as defined later in this subclause. This defaults to backslash (\). The character slash (/) is used in all the following text and examples, unless otherwise noted.

The character that when placed in column 1 of a charmap line, is used to indicate that the line is ignored. The default character is the number sign (#). The character percent-sign (%) is used in all the following text and examples, unless otherwise noted."

Problem:
This document should use the default <escape_char> and <comment_char>, rather than the characters chosen here. Using the defaults aligns this document with POSIX.2.

Action:
Reword the sentences as follows:

//for <escape_char>
"This defaults to backslash (\), which is the character used in all following text and examples, unless otherwise noted."

//for <comment_char>
"This defaults to the number sign (#), which is the character used in all following text and examples, unless otherwise noted."

Also, change the examples throughout this document to match the usage described here.

OBJECTION #50
Section 5. CHARMAP (lines 3283-3310)

Problem:
The <escseq2022>, <addset>, and <include> keywords are designed to allow charmap writers to specify ISO 2022 escape sequences. With more of the world's internationalization implementations moving to ISO 10646 and Unicode, it is not necessary to add increasingly-obsolete ISO 2022 syntax to the charmap.

Note also that the existing description of <addset> refers to the

<escseq> keyword, not <escseq2022>.

Action:
Delete these keywords, and the example at lines 3406-3478. Also note that the example variously calls a particular code set <eastern7bit> and <shift7bit>, and that the example also uses <escseq> as a keyword, even though the actual keyword is <escseq2022>. Of course, all this should be removed.

EDITORIAL #51
Section 5. Charmap (line 3374)

Current wording:
"...(hexadecimal constants is recommended)."

Problem:
Grammar.

Action:
Rewrite as: "...(hexadecimal constants are recommended)."

OBJECTION #52
Section 5. Width subsection (lines 3481-3511)

Problem:
Both the FDCC-set description and the charmap have keywords for defining width. It is incorrect to have them both places; it will only lead to confusion.

Also note, with a mixture of amusement and fatigue, that the width keyword is currently defined as taking an "unassigned positive integer" (line 3487).

Action:
Keep width information in only one place. This seems a bit more appropriate than the FDCC-set, but if it is retained there (over Objection #11), it must be removed here.

OBJECTION #53
Section 6. REPERTOIREMAP (entire section)

Problem:
There are multiple problems with this section. They include:

* The naming conventions chosen for symbolic characters. The cited justification is the "many POSIX charmaps registered with ISO/IEC 15897" and "use on the Internet". However, ISO/IEC 15897 only defines the information that can be contained in its registry of cultural elements, not the naming conventions to be used. The author of this draft has submitted multiple charmaps to be registered under ISO/IEC 15897 and has used the naming conventions he cites here. In essence, he is

endorsing himself when he points to those charmaps and their naming conventions.

Note, too, that the POSIX charmaps have been offered as international standards since the early 1990s, but they have only been used when organizations take free software (e.g., Linux). There is little evidence of them actually being used, and ample evidence that industry leaders who ship charmaps and locales are NOT using these naming conventions.

The naming conventions are unnecessarily obscure and should not be the ones used for a repertoiremap.

* The repertoire of the repertoiremap is curiously incomplete. It cites ISO/IEC 10646, but contains only a subset of characters in that standard. If the repertoiremap exists, it should contain the entire repertoire of characters in ISO/IEC 10646.

It's difficult to determine exactly what characters are and are not in the repertoiremap because they are reordered relative to their ISO/IEC 10646 code points. The repertoire may most closely match Unicode R2.0, but *without* any of the thousands of CJK Unified Ideographs.

* At lines 3642-3667, the repertoiremap includes "weight" characters (e.g., <a8>, <b8>, <c8>, etc.) that are supposed to indicate the position of the last a, the last b, the last c, and so on. While potentially handy for those who use the Latin script, it's questionable why Latin-specific weights should be in a repertoiremap. Further, these weights are equated to ISO/IEC 10646 codepoints:

```
<a8>    <U0252> Weight indicating the position of the last a
<b8>    <U0182> Weight indicating the position of the last b
<c8>    <U0255> Weight indicating the position of the last c
<d8>    <U018D> Weight indicating the position of the last d
<e8>    <U0264> Weight indicating the position of the last e
<f8>    <U0191> Weight indicating the position of the last f
...
```

However, these code points already are assigned to other characters. <U0252> is LATIN SMALL LETTER TURNED ALPHA; <U0182> is LATIN CAPITAL LETTER B WITH TOPBAR; <U0255> is LATIN SMALL LETTER C WITH CURL; and so on. This is an obvious conflict with ISO/IEC 10646.

* As noted, the order of the repertoiremap does not match ISO/IEC 10646. It should. There is nothing to be gained by changing the order, and a lot of easy look-up ability to be lost.

Action:
Since the ISO/IEC 10646 identifier names are being used elsewhere in the document, it is not clear that a repertoiremap is needed at all. If it continues to exist, the justification for symbolic names is faulty and should be removed. More mnemonic symbolic names should be substituted. The repertoiremap should include the full ISO/IEC 10646 repertoire. The weights must be removed to avoid conflict with ISO/IEC 10646. Entries must be in the same order as they appear in ISO/IEC 10646.

OBJECTION #54
Annexes

Problem:
This is a placeholder objection for the content of the annexes. They have not been reviewed at this time because there are so many objections to the main sections. Assuming those objections are processed appropriately, the annexes will have to change in multiple ways to accommodate the many changes.

Readers should NOT assume the annexes are considered correct and complete. For some specific comments on the annexes please see EDITORIAL #70, TECHNICAL #71, TECHNICAL #72, TECHNICAL #73, TECHNICAL #74, and EDITORIAL #75

Additional comments

EDITORIAL #55
Problem:
The Document type on the title page is specified as "International standard"

Action:
This must be corrected to show the document type as a Technical Report, since this is a DTR and not a DIS.

TECHNICAL #56
Section 4.2 LC_IDENTIFICATION (lines 730-731)

Current Text:
"If required information is not present in ISO 639 or ISO 3166, the relevant Maintenance Authority should be approached to get the needed item registered."

Problem:
This does not solve the problem of what to specify as the value for the language and/or territory keywords is missing.

Action:
Specify some appropriate default value to be used in those cases ("unknown", "unspecified", "unregistered", or some other short abbreviated form, if the intent is that these strings be limited to two characters). Separate out the suggestion regarding approaching the relevant Maintenance Authority to register new entity strings, since that is a distinct recommendation from what needs to be specified as the values of these fields in a given LC_IDENTIFICATION category.

TECHNICAL #57
Section 4.2 LC_IDENTIFICATION (lines 713-714)

Current Text:
"Natural language to which the FDCC-set applies, as
specified in ISO 639."

Problem and Action:
This does not make it clear whether both 2-letter codes
from 639-1 and 3-letter codes from 639-2 are intended here.
If that is the intent, it should be explicitly noted, so
that implementers do not make parsing assumptions that lead
to errors.

TECHNICAL #58
Section 4.3.2 "i18n" LC_CTYPE category, upper, lower
(lines 1030-1034, 1073-1074)

Problem:
The definition of upper unaccountably omits 01A6.
The upper definition also unaccountably includes the titlecase
digraphs 01C5, 01C8, 01CB, and 01F2. The lower definition also
unaccountably includes the title case digraphs 01C5, 01C8, and
01CB, but *not* 01F2.

Action:
Add 01A6 to upper and make the treatment of the titlecase digraphs
consistent, at least, or better omit them from the lists.

TECHNICAL #59
Section 4.3.2 "i18n" LC_CTYPE category, toupper (line 1402)

Problem:
The case pair (<U0280>,<U01A6>) is omitted.

Action:
Fix it.

TECHNICAL #60
Section 4.3.2 "i18n" LC_CTYPE category, class (lines 1666, 1689)

Problem:
The use of terminal semicolon is inconsistent on these two lines.

Action:
If, contrary to U.S. OBJECTION #11, these sections are retained,
then at least make the syntax of the entries consistent.

TECHNICAL #61
Sections 4.4.12..4.4.13 LC_COLLATE section reordering

(lines 2157..2205)

Problem:
This section reordering syntax is inconsistent with the similar but distinct syntax defined in ISO 14651. It is also not defined in the BNF in C.2 (p. 107 ff). The phrase "and <sort-rule>s not to be changed may be given by empty specifications" is completely unclear.

Action:
Resolve this inconsistency to match ISO 14651, or explain how an inconsistent syntax for section reordering can coexist with tailorings that follow the syntax defined in 14651. Fix the BNF in C.2 to define whatever syntax is the outcome of this resolution.

TECHNICAL #62
Section 4.4.12.2 Example of section reordering (lines 2187..2200)

Problem:
This example makes use of section-symbols for reordering sections, but does not show how such symbols would be present in the table to be targets for the reorder-section-after keyword. The "i18n" LC_COLLATE category defined in section 4.4.14, derived by copying the ISO 14651 table, contains no such section-symbols. As it stands the example is incoherent.

Action:
Expand the example until it is an accurate example that would show how section-symbols could be inserted and then reordered.

TECHNICAL #63
Section 4.4.10.1 Example of "reorder-after" (lines 2130..2138)
and Section 6 Repertoiremap (lines 3550-3554, lines 3642..3667).

Current text:
"... <y8> is used to indicate the last entry of the <y> letters.
... <z8> is used to indicate the last entry of the <z> letters."

Problem:
The problem here is that the concept of "the <y> letters" and "the <z> letters" is nowhere defined. It is not defined in 14651, which LC_COLLATE explicitly depends on, but is only mentioned in the Repertoiremap section of 14652, where the symbols used here are incorrectly defined:

"<y8> <U01B3> Weight indicating the position of the last y
<z8> <U0293> Weight indicating the position of the last z"

The other entries in the repertoire map are characters in the repertoire, but these instances as "weights indicating

the position of the last z", and so on. They might be *used*
as symbolic weights in a collation table, but in the
repertoiremap they are simply symbolic equivalences for
some other characters, most of which don't otherwise appear
in the repertoiremap. If the repertoiremap is to be taken
seriously, they are simply names then for particular UCS
characters, rather than placeholders for last positions for
some otherwise undefined set of "a"s, "b"s, and so on. And
if they are intended to be treated as fixed targets for
reorder statements, they can be invalidated by future additions
of characters to the UCS. Furthermore, the entire concept of
"the last y", "the last z", and so on is relative to particular
collation tailorings.

Action:
Remove these entries or correctly define them. Fix the
example.

EDITORIAL #64
Section 4.7 LC_TIME (lines 2539, 2546)

Problem:
Casing inconsistency for "Gregorian".

Action:
Fix these and search the document for other instances, if any.

EDITORIAL #65
Section 5.1 Character Set Description Text (lines 3443, 3505)

Problem:
The examples are still using nonexistent UCS values.

Action:
There is no excuse, even in examples, to make use of
unassigned code positions when perfectly good, illustrative
characters are available. Fix <U0244> to an actual character.
Fix <U3200>..<UFAFF> to an actual range of CJK characters.

EDITORIAL #66
Section 6 Repertoiremap (line 59)

Problem:
Number agreement "symbolic names ... are predefined and refers..."

Action:
"refers" --> "refer"

TECHNICAL #67
Section 6 Repertoiremap (lines 3531-3532)

Current Text:
"Characters not in ISO/IEC 10646 may be referenced by the
symbolic character names <P00000000>..<PF8FFFFFFF>.

Problem:
The rationale for the choice of the endpoint of this artificial
name range, which would imply a 36-bit numerical name space
(minus a few values) is not apparent. Is there one too many
"F"'s in that end range?

Action:
Provide a rationale for the endpoint or fix it to something
more self-explanatory.

TECHNICAL #68
Section 6 Repertoiremap (line 3523..3526)

Current Text:
"The repertoire mapping is defined by specifying ... and
optionally the long ISO/IEC 10646 character name in the
following syntax:

 "%s %s %s\n",<symbolic-name>,<10646-short-identifier>,<comments>"

Problem:
The textual description says only that the third field may
optionally be the long 10646 character name, but that is
inconsistent both with the formal description, which implies
that the third field may be any arbitrary comments, and
the actual repertoire map, which includes the <a8>..<z8>
entries that do not have 10646 character names in the corresponding
locations.

Action:
Change the formal description to indicate that it can contain
only the ISO/IEC 10646 character name. Change the actual
repertoiremap to remove entries that do not have 10646 names.

TECHNICAL #69
Section 7.3 Charmap conformance (lines 5893..5894)

Current Text:
"A charmap description is conforming to the Technical Report
if it meets the requirements in clause 5."

Problem:
No actual complete example of a conforming charmap description
is given in clause 5, so it is rather difficult to determine
what it would take for a charmap to be conforming. The U.S.
has already asked (see OBJECTION #50) for the removal of the
additional apparatus of <escseq2022>, <addset>, and <include>,

which were added only for the increasingly irrelevant definition of ISO 2022-type charmaps. If these objectionable additions are removed, in what way does the definition of a charmap description in 14652 differ from POSIX.2, and why should any separate conformance specification be given for it? If the difference lies in the addition of a WIDTH section, then why is that not specified at line 5973, item 18, where the enhancement of the charmap specification is summarized?

Action:
Remove the ISO 2022 support from Clause 5. If no significant difference then remains from the POSIX.2 definition of a charmap description, then remove the redundant Clause 5 itself and the superfluous conformance clause 7.3. If a significant difference does remain, then properly document that in Annex A and in Clause 5 itself.

EDITORIAL #70
Section B.1.7 LC_MESSAGES rationale (line 6575)

Current Text:
"The ISO internationalization working group"

Problem and Action:
This self-reference to SC22/WG20 is still unclear, and should be clarified.

TECHNICAL #71
Section B.1.7 LC_MESSAGES rationale (lines 6572..6578)

Problem:
The rationale provided incorrectly describes the category itself. LC_MESSAGES claims it "defines the format and values for affirmative and negative responses". But the rationale claims that it "is described in clause 4 as affecting the language used by utilities for their output." Furthermore the rationale makes an exorbitant claim for the development of an interface "that would allow applications ... to access messages from various message catalogs, tailored to a user's LC_MESSAGES value" that does not match the 6 meager lines on this topic currently available in WD5 of 15435 (cf. WG20 N794, p. 14).

Action:
Correct the rationale to accurately reflect the definition of LC_MESSAGES in the DTR itself and provide an accurate rationale for its definition and use.

TECHNICAL #72
Section C.2 Grammar for FDCC-sets (line 6726)

Current Text:

global_statement = ... 'comment_char' SP char_symbol EOL

Problem:
Because of the definition of EOL (line 6949) as including an optional comment, this definition of a comment line is tangled and inconsistent.

Action:
Fix the problem.

TECHNICAL #73
Section C.2 Grammar for FDCC-sets (line 6790)

Current Text:
order_statements = order_start collation_order order_end ;

Problem:
This definition doesn't allow for symbol weights occurring in the table before the order_start.

Action:
Fix the problem.

TECHNICAL #74
Section C.2 Grammar for FDCC-sets (line 6795)

Current Text:
order_opt = order_opt [comma opt_word] ;

Problem:
This is a recursive definition.

Action:
Fix the problem.

EDITORIAL #75
Annex D

Problem:
The list of issues after the first paragraph and ahead of the D1 section are in fact the comments provided by Japan. The items 1, 2, and 3 unnecessarily
duplicate the items 1, 2, and 3 under D.1.

Action:
Remove items 1, 2, 3 in the "Issues List", retitle the "Issues list" to "Outstanding Issues Raised by National Bodies", and reword the remaining first and last paragraph of the section ahead of D.1. Correct the grammar in the first paragraph: "it is decided to be" --> "it has been decided that it will be". Other minor wording fixes can be worked out in committee. In the U.S. comments in Section D.2, fix the "--" double dashes to em dashes and the "*are*" to italics.

========== end of U.S. comments ==========