L2/99-286

US comments to the PDTR ballot on TR 14652

Documents: SC22 N2955 (SC22/WG20 N690, L2/99-209)

September 24, 1999

1. General Comments

The US is in favor of the change of status of this document from a draft standard to a draft technical report, which seems appropriate under ISO directives, given the disagreements within the committee regarding how to proceed and the lack of consensus to make the document an International Standard. However, the conversion from a draft standard to a draft technical report does not seem to have been completed. Two issues stand out:

1. There are a number of places in the document where it refers to itself as a "standard". These must, of course, all be corrected. Places we noted were: p. iv, lines 77, and 110 (two times) and p. 3, line 268. The entire document should be searched to guarantee that no other instances occur.

2. While it is not unheard of for a Technical Report to contain a conformance clause, and makes a certain amount of sense, given the history of this document, it is still unusual. Furthermore, the language throughout the Technical Report is expressed in terms that express conformance: "...shall be used as...", "...shall contain...", and so on. The more appropriate rhetorical structure for a Technical Report is simply to describe and specify things, rather than strive to sound as if it is a standard even when it isn't. To pick a typical example from page 7 of the document:

> "The FDCC-set definition text shall contain one or more FDCC-set category source definitions, and shall not contain more than one definition for the same FDCC-set category."

That is a prescriptive specification, tied to the conformance clause. The descriptive reformulation would be:

> "The FDCC-set definition contains one or more FDCC-set category source definitions, and does not contain more than one definition for the same FDCC-set category."

We would prefer that the text be rewritten to remove its prescriptive character. But failing that, it would make sense to at least add a prominent paragraph to the Forward that describes the origin of the document, explains why it is written *like* a standard, but is not actually a standard. What is there now is just directives boiler plate, but does not explain why the text is still written in its prescriptive manner and why it still has a conformance clause.

2. Technical Comments

2.1. p. v, line 126

"A standard set of values for all the categories has been defined covering the repertoire of ISO/IEC 10646-1."

This may not be correct, depending on what is meant by "covering the repertoire". LC_COLLATE, by reference to FCD3 14651, does indeed provide full coverage for 10646-1 (plus Amendments 1-7), plus two characters from Amendment 18. LC_CTYPE also intends to cover the same repertoire, although it is difficult to determine whether it is actually complete. In any case, PDTR 14652 is insufficiently precise in defining the repertoire to be covered, since the "repertoire of ISO/IEC 10646-1" is constantly changing.

It would be helpful to refer to specific versions of the Unicode Standard to define repertoire, rather than 10646-1 plus amendments, since that would be more precise (and mechanically checkable, since a fixed, machine-readable data file exists for each Unicode version). For example, 10646-1 plus Amendments 1-7 (or 1-9, for that matter, since Amendments 8 and 9 did not add characters), corresponds to Unicode 2.0.0. But the addition of the euro sign and object replacement character can be specified exactly as Unicode 2.1.9, but does not correspond to any particular configuration of 10646-1:1993 plus amendments.

2.2. p. 1, line 169 in Normative References

This lists 10646-1:1993 as the normative reference. We think this is a mistake. Given the timing of this Technical Report, it is clear that the reference should be to 10646-1:2000, the second edition. Then the specification of the particular repertoire of that standard to be covered should be by reference to a particular version (or versions) of the Unicode Standard, or by reference to the numbered collections of the standard (e.g. collection BMP-AMD.7 is explicitly provided in 10646-1:2000 to specify the repertoire that matches Unicode 2.0.0.).

2.3. p. 3, line 270, section 3.2.1 Notation for defining syntax.

This entire syntax is unhelpful and could easily be dispensed with if the thrust of this document were not to maintain upward compatibility with POSIX specifications. In a descriptive specification of cultural conventions, it is far better to simply specify a tagged format of some sort. This would allow dispensing with the superfluous C-style printf specifiers and "\n" line terminators, etc. Such a format would be more useful to other users of the cultural conventions, and if properly designed could be easily converted to XML, where it could be transmitted and verified in standard ways.

Furthermore, use of the printf argument specifiers in this meta-syntax conflicts with the definition of other similar format specifiers for date, time, and other format specifications within particular FDCC-set categories. This is most confusing in a Technical Report. The TR should describe the specification for cultural conventions in an implementation neutral but clear way -- and then the POSIX++ implementation is free to make use of printf-style specifiers for its actual implementation on UNIX platforms.

2.4. p. 4, line 304, section 3.2.3 Portable character set.

This is completely unnecessary for the specification of cultural conventions. A neutral specification would simply make use of the UCS identifications of characters. The requirement to specify the Portable character set is a POSIX artifact that should itself be restricted to the POSIX specifications.

2.5. p. 7, line 491ff

The definition of category body is not consistent with the allowance for comment lines at any point.

2.6. p. 9, line 585

"Concatenated constants can include a mix
of the above character representations."

This is just a silly idea. At the very least this should be deprecated.

2.7. p. 11, line 677ff,

Section 4.2 LC_IDENTIFICATION
"All keywords are mandatory unless otherwise noted, and
the operands are strings."

This is a *good* example of how the specification should work. This should be carried through for all the other FDCC-set categories, so that the printf-style format specifications can be dropped.

2.8. p. 11, line 717

"i18n:1999" ==> "i18n:2000"

2.9. p. 12, line 738,

the "i18n" LC_IDENTIFICATION category.

"ISO" does not match the spec for territory, which required this to be a 2-letter 3166 code.

2.10. p. 12, line 740ff.

Update all 1999's to 2000's.

2.11. p. 13, line 795ff.

The "double increment hexadecimal symbolic ellipses" are a clever, but still goofy convention. They make the tables for LC_CTYPE less mind-numbing, but a better strategy is to define all such property categories by reference to a specified level of the UnicodeData.txt database, and then to define only the deltas from that to satisfy the committee on certain categories that might differ in usage from that specified in UnicodeData.txt. This would be far, far easier to verify and to implement than what is currently provided.

2.12. p. 14, line 874ff, xdigit

It is an incredibly bad idea to allow hex digits to have script variants. xdigit should always refer to 0..9, a..f, A..F, period, full stop. Anything else is inviting implementation disasters.

2.13. p. 15, line 926 map

The "map" keyword is an unnecessary extension of the concept of an FDCC-set. The FDCC-set is not the place that all possible relations between characters are defined. The concept of an FDCC-set should be restricted to the specification of *cultural* practices for formatting quantities, names, addresses, and such, which are clearly relevant to localization of software. Mixing it all up with an architecturally unsound approach to the specification of character encodings and character semantics is a very, very, bad idea. This should just be removed.

2.14. p. 16, line 976ff,

section 4.3.2 Character string transliteration

It is no more acceptable to have a bad transliteration specification tucked away inside the LC_CTYPE category than it was to have it specified as a separate category. In fact, it is even less coherent than before, when treated as a part of the LC_CTYPE specifications of cultural conventions. This section should be removed, as it has nothing to do with the specification of cultural conventions.

By the way, if this is included in the Technical Report, it is quite likely that it will be widely ignored by those implementing transliteration (outside the POSIX community at least), and will just make the committee look silly.

2.15. p. 18, line 1091.

<U3200>..<UFAFF> is not the correct
range for ideographic characters in UCS, in whatever
version.

2.16. p. 19, line 1108ff "i18n" LC_CTYPE category.

Once again, for the Technical Report to try to define all this
is an incredible waste of time and error-prone. Now that
14652 is a Technical Report and not an International Standard,
the allergy it has previously shown towards referring to
the industry standard implementation of these properties
should be shrugged off in favor of a more useful and
accurate approach.

We have not tried to locate *every* error in these tables, but
some examples will show the problem.

For the "upper" category: 01A6 should be added; 01C5, 01C8,
01CB, 01F2 should be removed (those are titlecase, not uppercase);
all the IPA extensions should be removed (the IPA small-caps
letters are notionally lowercase, and should not be included
in the "upper" category); 03D2..03D4 should be added;
the range <U03E3>..(2)..<U03EF> has the wrong start point --
it should be <U03E2>, ... and so on.

On p. 21, line 1315,
the CJK unified ideographs range starts
at the wrong point. It should be 4E00, not 4E01.

On p. 22, line 1369,
the cntrl range is <U007F>..<U009F>,
not <U0077>..<U009F>.

For the "punct" category on p. 22, this departs very strongly
from the UnicodeData definition of punctuation. "punct" here
includes currency signs and miscellaneous symbols as well
as true punctuation. This should be justified or corrected.

2.17. LC_CTYPE category (continued)

By the way, the consistent use of UCS designations to refer
to characters in the "i18n" LC_CTYPE specification puts the
lie to the usefulness and requirement for 14652 to define
and make use of the bizarre repertoiremap of section 6 of
14652 (p. 62 ff). Consistent use of ASCII characters as
themselves, a judicious use of a few other symbolic names
for the few other characters explicitly referred to in
ways where that would help (e.g. for examples for
collation, etc.) and use of UCS symbolic names for everything
else would be far, far preferable for this Technical Report,
to the way it currently stands.

2.18. p. 28, line 1848

"This ordering is used by regular expressions...
also as the collation weight to be used in sorting."

The intent of this sentence is unclear. What is the
effect on pattern matching if collation weights *are*
explicitly specified? This item must be clarified.

2.19. p. 29, line 1888

"This value is elsewhere referred [to] as
the COLL_WEIGHT_MAX limit." (also on p. 33, line 2081)

Where is this elsewhere? Either specify the elsewhere, or
drop this non sequitur.

2.20. p. 31, line 1964

"lower than the coded character set value"

Since this is talking about a *symbolic* ellipsis here,
should this be lower in the sequence of symbolic names,
rather than lower in the *coded character set value* ?

2.21. p. 31 ff., general

All this specification of *how* to assign weights and deal
with the IGNORE's, and so on should be left to 14651, and
acquired in 14652 by reference. This discussion here is
just inviting inconsistencies in implementation. The appropriate
scope for 14652 is to specify the additions to the LC_COLLATE
syntax over and above 14651, i.e. specification of limits
on numbers of levels, introduction of the symbol-equivalence
keyword, and use of the copy keyword.

2.22. p. 32, line 2014.

"A <comment_character> occurring where
the delimiter ";" may occur, terminates the collating
statement."

This is a *good* example of the way the entire text of the TR
should be written. This is descriptive, and does not resort
to the prescriptive "shall" when describing a format.

2.23. p. 34, line 2121 Note.

This claim is not generally true. It depends on what type of
decomposition is done. This claim is for canonical decomposition.
Clarify the text if this note is to remain.

2.24. p. 35, line 2203 F and B, and B and P

The way these "and"s are used here, this claim is very difficult
to parse and understand. This should be broken out to explicit
single statements about what terms are mutually exclusive.

2.25. p. 36, line 2226 ".

..and shall be present in the source FDCC-set
copied via the "copy" keyword."

This is not required by 14651, and should not be, because it
restricts tailorings unnecessarily. The point is, that a
<collating-symbol> has to be defined before it is referred
to by a reorder-after statement, but there is no reason why
that definition has to come from a particular FDCC-set copied
via the "copy" keyword. If this is POSIX-specific stuff, once
again, the implementation details should be elsewhere, and not
in the neutral specification of FDCC-sets for cultural
conventions.

2.26. p. 37, line 2293, section 4.4.12.1 section reordering statements

This syntax extends the syntax of 14651 for this statement type.
That should be explicitly pointed out and explained, since it
is not going to be the expectation of those using the TR.

2.27. p. 38 ff., "i18n" LC_COLLATE category

The long list of collating symbols is not needed here. This
completely duplicates the list present in the 14651 common
tailorable template, but with the introduction of just
four collating-symbols: <BLANK>, <CAPITAL-SMALL>, <SMALL-CAPITAL>,
and <BOTH>. The introduction of <BLANK> is not necessary.
That is the result of an error in the 14651 table for two
entries, where "<BLANK>" should be corrected to "<BASE>".
The other three are simply unexplained additions by the
editor. And even *if* they were needed for some further
purpose, the correct specification for 14652 is simply to
specify the 3 additional collating-symbols, along with
any desired symbol-equivalences, rather than duplicating
the rest of the list in 14652.

2.28. p. 53, line 3277 ff. "i18n" LC_MESSAGES

Why is "[+1]" and "[-0]" suggested as the default
definition for yes and no in the "i18n" LC_MESSAGES category?
"[1]" and "[0]" might make some sense, though "y" and "n"
would be better, given the status of English as the
international language. But at the very least, the
minus sign on the zero makes no sense and should be
removed.

2.29. p. 53, line 3237, Section 4.9 LC_PAPER

Paper size conventions certainly are cultural conventions,
but there is a mismatch between what the requirements should
be for specifying these cultural conventions and what is
presented here.

For the U.S., for example, paper size conventions are

"letter" (8-1/2"x11") and "legal" (8-1/2"x14"), whereas
for the U.K., normal printing paper is A4 (210mm x 297mm).
In addition to the metric series defined in DIN 66008,
there may be other local traditional sizes not defined in
terms of the metric system, as for example, traditional
sizes used in book publishing.

A specification for cultural conventions should allow the
expression of all this, and not attempt to reduce everything
to one height and one width expressed in millimeters (which
won't even be accurate for U.S. paper sizes, by the way).

This FDCC-set category seems to be, instead,
another implementation-driven category that could be used
transiently to convey some "locale-based" information to
a printing process through some API. Effectively, the
FDCC-set is being conceived of a one gigantic C struct
for storing anything an application might ever conceivably
need for cultural adaptability for access to an API, rather than
as a "specification for cultural conventions", which is
nominally what this Technical Report is about.

2.30. p. 54, line 3262, Section LC_NAME

This entire specification is very weak and basically useless.
The list of keywords provided is Western-centric.

This is a good example of a category that should be
reorganized to *first* provide a discursive listing of
name formatting and salutation conventions in different
cultures, before trying to invent a syntax to make it
machine-readable. Otherwise this is "standards-fishing"--
promulgating a standard syntax in the hope that it may
be sufficient and implementable, but in the absence of
evidence of use or enough data to demonstrate its appropriateness
to the field of application.

2.31. p. 54, line 3298 %m Middle names

It is unclear whether this is intended to be one string
associated with potentially multiple names. If so, this
should be clarified. Also, what is the relationship between
the potentially multiple names of %m and the apparent
single initial letter of %M "Middle initial"?

2.32. p. 54, line 3200 %p Profession.

This is also unclear as to intent. Does this, combined
with the "i18n" LC_NAME category value for name_fmt
imply that "Lawyer John C. Smith" and
"Assistant Manager at McDonald's James T. Peabody" are
valid formatted names?

2.33. p. 54, line 3274 name_gen

The example given for the use of Japanese "-sama" as a salutation is incorrect. While it is true that "-sama" is not gender-specific, it is not appropriate "for all persons", and certainly not in all contexts. Furthermore, it is an honorific, and not a salutation, since it cannot be used independently of a name.

2.34. p. 55, line 3305

"The va[lu]e may be stored in the database with the person information."

What database? Once again, this specification is referring to some unclear context outside the scope of the document, with the implication that this is designed for some particular implementation, rather than standing independently as a specification. Either delete the reference to "the database" or make it clear in this document what is intended by it.

2.35. p. 55, line 3318 name_fmt specification

This is essentially just a mask definition for a format. It would be better stated explicitly as a mask, rather than as a printf style format string. And there is nothing gained by use of the symbolic name "<p>", rather than just "p"; this just clutters the specification and makes it hard to read.

The same comment applies to the format masks for LC_ADDRESS pos.al.fmt (p. 56, line 3400), and the LC_TELEPHONE tel_int_fmt (p. 57, line 3443).

2.36. p. 55, section 4.11 LC_ADDRESS

There is no justification for why the LC_ADDRESS category, which should be focussed on specification of cultural specifications for addresses, is cluttered up with a bunch of keywords related to ISO 3166 country codes, motor vehicle country codes, ISO 2108 ISBN codes, and ISO 639 language codes. Once again this looks like a case of tossing in the kitchen sink, rather than designing the category appropriate to its usage. All of these superfluous keywords should be removed from the category. If an FDCC-set needs to specify any of this information, it should be elsewhere, and not just tossed into LC_ADDRESS for no apparent reason.

The Rationale for this category in Annex B does not discuss this, either.

2.37. p. 57, section 5. CHARMAP

As we have indicated before, specification of a CHARMAP is completely out of scope for cultural conventions. This is just a bad design that continues the way POSIX deals with implementation of character set encodings. There is no good reason for the 14652 Technical Report

on the specification method for cultural conventions to
be continuing and expanding on this bad design. It
should be removed. (See further comments on the Annex B
Rationale below.)

2.38. p. 58, line 3502, <escseq>

Even with the example now provided on p. 61, this
explanation for <escseq> is still almost incomprehensible.
The same applies to the discussion of operands for
the <include> keyword on p. 59, which also refer to
"the g-set or c-set to be defined" and the "range
of characters in the referenced charmap". This is
another reason why this entire section should be dropped
from the TR.

2.39. p. 60, line 3593 ff.,

"The encoding part shall be expressed as..."

Even if this is specified for upward compatibility with
existing POSIX practice, it is ridiculous in this day
and age to be encourage the use of octal or decimal
in the specification of character encoding values.
Hexadecimal is clearly the radix of choice, and should
be encouraged by the technical report. The others should
be deprecated.

2.40. p. 61, line 3625 ff.,

"Example of using ISO 2022 techniques"

These examples do now provide enough information to be
able to make a guess at how the CHARMAP keyword "<escseq>"
is intended to be used. But the examples also illustrate
the reason why this entire CHARMAP section should be
removed from the Technical Report. This is an implementation
format for extension of the POSIX architecture for
character set encoding definitions, rather than any
information useful for the specification of cultural
conventions. Anyone wanting to understand the structure
of these character encodings would be far better off going
to the easily available (and far more complete and accurate)
book by Ken Lunde on CJKV Information Processing. And as
for cultural conventions, this section of CHARMAP (as well
as the subsequent one on REPERTOIREMAP) just get in the
way and confuse the issues.

2.41. p. 61, lines 3661, 3662.

If the CHARMAP section and examples stay in, at the very
least, the example characters used should be legal,
assigned characters. The comment says "the character
codes are only examples", but in fact <U0365> and <U0744>
are not valid, assigned characters in the version of
10646-1 normatively referred to by the Technical Report.

At the least, make the effort to use assigned characters.

2.42. p. 62, Section 6 REPERTOIREMAP

Once again, we object to this ridiculous REPERTOIREMAP, which does not belong in 14652. It should be removed.

The justification on p. 63 keeps getting longer, but is no more convincing than before. The *concept* of a repertoiremap is prior art, but the wholesale extension of that prior art to include thousands of the editor's whole cloth inventions can hardly be characterized as prior art. It is simply an example of dogged persistence.

At most, perhaps 100 or so of these symbols beyond the range of Latin-1 characters have any usefulness. The rest are just the result of a mediocre concept extended at least two standard deviations beyond the range of reasonableness. Why would anyone want to use the symbol "<W*;?J>" instead of "<U1FAF>" to refer to U+1FAF GREEK CAPITAL LETTER OMEGA WITH DASIA AND PERISPOMENI AND PROSGEGRAMMENI? Or "<_./>//>" instead of "<U25E2>" for U+25E2 BLACK LOWER RIGHT TRIANGLE?

The silliness of this REPERTOIREMAP, which takes up nearly one quarter of a 114 page Technical Report, is illustrated by its incompleteness. After inventing 2318 of these symbols, the editor apparently ran out of gas, and didn't extend them to cover Georgian, Thai, Lao, or any of the Indic scripts. He included symbols for compatibility Arabic positional shape characters, but not symbols for compatibility fullwidth ASCII or halfwidth katakana characters. Why? And it is completely unclear how this mechanism would be extended, except by more arbitrary and nearly random string assignments, to cover the repertoire of Unicode 3.0 (with 1165 Yi syllables and 630 Canadian Aboriginal Syllabics syllables, just to name the most problematical).

2.43. p. 64, line 3823 - 3848

The introduction of these "<a8>" .. "<z8>" symbols here is problematical. First of all, these are intended for use with LC_COLLATE for tailoring of the table from 14651, but that is not explained here or anywhere.

Second, the particular assignments of these symbol weights to particular Unicode characters is invalid in general. There is nothing that ensures that U+0252 is going to be the "last A" in the common tailorable template table. This could be affected either by a revision of that table or the addition of new characters to 10646 that get incorporated into the table. It is inadvisable in any case to tie a hack for tailoring Latin characters to particular Unicode values. The proper way to do this is to introduce "<a8>", etc. as collating-symbols in LC_COLLATE and then introduce the appropriate tailoring with reorder-after statements based on a particular version of the 14651 table.

2.44. p. 92 Annex B (informative) Rationale

The entire Technical Report now is informative. Restructuring the document as a Technical Report should move most of this material into the main body of the text. There is no rhetorical reason why it should be separated off in an annex, since it provides explanatory material that is often needed at the point where concepts are introduced.

In particular, the LC_MONETARY Rationale (B.1.4) is a better start toward what the Technical Report *should* contain about monetary formatting than the definition of the LC_MONETARY category itself.

2.45. p. 92, line 6183

"an ISO/IEC 10646 system that has
defined 16-bit bytes may..."

There is no reason for the document to be obtuse about this. No Unicode (or 10646) system implements 16-bit characters by defining 16-bit bytes. "Byte" in the industry has an unshakeable meaning of an 8-bit quantity. It is only character standards diehards who keep insisting that "byte" is variable width and refers to the width in bits that a character is encoded in. The entire industry has chosen the other route, and measures data in bytes, which is a fixed-size quantity. The days when bytes did differ in size on different machine architecture are long gone.

2.46. p. 93, Section B.1.3 LC_COLLATE Rationale

Most of this discussion is out of scope. It is arguing about 14651, rather than providing the particulars for the rationale for the LC_COLLATE category.

2.47. p. 94, line 6286

"The syntax for the LC_COLLATE
category source is the result of a cooperative effort between representatives for many countries and organizations working with international issues, such as UniForum, X/Open, and ISO, ..."

If this rationale is to contain all the general discussion about collation weighting and contents relevant to 14651, rather than just the discussion of the specific keywords for the LC_COLLATE category in the FDCC-set, then why is not the Unicode Consortium on that list? Either add it or remove all the out-of-scope discussion of 14651-related issues from this rationale.

2.48. p. 95, line 6315

"It is estimated that the Technical
Report covers the requirements for all European languages,
and no particular problems are anticipated for Cyrillic
or Middle Eastern scripts."

First of all, it is not PDTR 14652 that covers these
requirements, but FCD3 14651 that does. PDTR 14652 simply
provides the metasyntactic shell to incorporate the
14651 framework inside the LC_COLLATE category.

Secondly, 14651 *does* cover Cyrillic, Arabic, and Hebrew,
as well as the rest of the scripts included in Unicode 2.0,
so there is no reason to make this imprecise statement
about anticipations.

2.49. p. 98, Section B.1.3.3

Sample FDCC-set specification for Danish.

line 6485: The symbol "<SPECIAL>" is not defined.

In general, the introduction of the particular symbols
needed for this Danish collation specification should be
done *here*, and not in the "i18n" LC_COLLATE category
definition on p. 38 and in the "i18nrep" REPERTOIREMAP
on p. 64.

2.50. p. 103, line 6753.

"It is expected that National Standards
Bodies will provide specifications."

This seems a rather forlorn hope, given a bad format
metasyntax in the first place.

And why insert a plaintive cry for other NB's to do the
work, instead of just digging into the IBM Green Book
discussion of Date Format as a starting point?

2.51. p. 103, line 6762.

"The internationalization working
group is developing an interface..."

Who? What internationalization working group, affiliated
with whom? This is not made clear in the document.

2.52. p. 104, Section B.2 Character Set Rationale

This rationale is completely unconvincing as a rationale
for including the CHARMAP mechanism in the Technical
Report 14652.

In particular, the claim that the "charmap was introduced to resolve problems with the portability of, especially, FDCC-set sources" should have been addressed simply by using 10646 (i.e. Unicode) as the *reference* character set for the Technical Report. This is the obvious solution, as taken by the HTML and XML standards. This approach allows for a source document to be represented in any character encoding, but it is interpreted *as if* it were converted to the reference character set, i.e., the UCS. In fact, most implementations will *actually* convert the source document to Unicode, to simplify their parsing/lexing engines. The same approach should now be taken towards all such specifications, including that of 14652, now that the UCS is a reality. Continuing to fob off this old model of "source portability" from POSIX on new standards and technical reports does a disservice to those who are trying to understand and implement them. Instead of assisting in source portability, it really only succeeds again and again in unnecessarily importing all the complexities of legacy character encodings into standards and technical reports that have nothing to do with the details of character encoding.

2.53. p. 105, Section B.3 Repertoiremap Rationale

"The repertoiremap was introduced to make FDCC-sets independent of the availability of charmaps."

Once again, this entire argument would be obviated by making the UCS the reference character set for the document. All the complexity of CHARMAP and REPERTOIREMAP would be pushed off to where it belongs, in POSIX implementations of the technical report, rather than in the technical report itself.

Technical Comments re Section 4.5 LC_MONETARY

We consider the specification in section 4.5 to illustrate the nature of the technical problems endemic to the TR's entire approach to the specification of cultural conventions. The metasyntax provided for LC_MONETARY is designed to be an extension of existing POSIX implementations of currency formatting. However, the extensions are:

A. Over-elaborate in terms of particular parameter specifications.

B. Insufficiently precise to be well-defined or to avoid undecidable cases of conflicting parameter specifications.

C. Result in specifications of cultural conventions for monetary formatting that are incomprehensible to the human reader, but rather are designed to facilitate a programmers task of parsing out parameters and setting a number of boolean settings in a localedef implementation.

D. Have a "data normalization" problem forced by insisting that a single FDCC-set must have *two* monetary formats specifiable--

   one for the "local" currency and one for the "international"
   currency. This is the Unix euro hack. It seriously complicates
   specification of cultural conventions by pushing the euro
   problem onto the procrustean bed of the single locale. This
   is another sign of implementation considerations driving a
   complex and unclear specification, rather than considerations
   of clear and simple exposition driving the specification.

We consider the third problem to be particularly egregious, since it means that registered FDCC-set's will be effectively incomprehensible in the registry and be unmaintainable by visual inspection. This is likely to result in duplicate, overlapping, and/or mistaken registrations, where the formatting intent of the human trying to produce these FDCC-set definitions may not match the behavior of the implementations using them.

The problems can be illustrated by examining the "i18n" FDCC-set definition proposed for LC_MONETARY, and then by giving another example where the mind flows freely, posing an outlandish case to see what the metasyntax allows (and thereby what it *requires* of implementations).

```
LC_MONETARY
% This is the 14652 i18n fdcc-set definition for
% the LC_MONETARY category.
%
int_curr_symbol     ""
currency_symbol     ""
mon_decimal_point   "<,>"
mon_thousands_sep   ""
mon_grouping        -1
positive_sign       ""
negative_sign       ""
int_frac_digits     -1
frac_digits         -1
p_cs_precedes       -1
p_sep_by_space      -1
n_cs_precedes       -1
n_sep_by_space      -1
p_sign_posn         -1
n_sign_posn         -1
%
END LC_MONETARY
```

O.k., but what does this mean? It apparently is an attempt to provide a vanilla default that effectively does nothing except specify that "," is the decimal separator. But rather than being conceived in terms of a visual international currency format that would make any sense whatsoever, it is conceived in terms of the implementation of localedef, with values to match variable initializations in that program (null strings, unused values undefined). But if you think in terms of actual formatting recommendations, this implies that the currency number 1000 would format as:

```
   "1000"
```

(or possibly "1000,0" or "1000,00" or ..., since frac_digits in not specified!)

But the negative value for a currency number -1000 would *also* format as:

```
   "1000"
```

(or possibly "1000,0" or "1000,00" or ...)

We don't think it was the intention of the editor of 14652 to recommend that positive and negative currency values be formatted exactly the same, but that is the implication of this LC_MONETARY definition. So an unreasonable default has crept in, simply because thinking in terms of program parameter initialization for setting values in a cryptic metasyntax does not lend itself to specification of real formats that would make sense as defaults or recommendations.

By the way, there is a definite technical problem which *must* be addressed to even make the specification comprehensible. The meaning of -1 as an argument value for int_frac_digits, frac_digits, p_cs_precedes, etc. through n_sep_by_space, is not defined in the TR. So as it stands, the LC_MONETARY definition is anomalous. It doesn't mean anything to have a negative one number of fractional digits to the right of a decimal separator, for example.

Now for the overpermissiveness and imprecision of the metasyntax suggested. Let's consider the case of Lower Slobovia. Lower Slobovia used the tugrik as its local currency until the end of 1998, but had an unplanned currency reform starting earlier this year. Things being rather chaotic in Lower Slobovia, the currency reform didn't complete until 1999-09-16, when Lower Slobovia officially declared the slobovik as its currency and established the conversion rate of 1 slobovik for 9 tugrik, suggested by the court astrologer because the King of Slobovia has 9 daughters. Now since Lower Slobovia derives most of its income by rather shady money-laundering, they have adopted the USD as their international money formatting convention. This tends to obscure the conversions back and forth from dollars to sloboviks (or tugriks). Lower Slobovia has chosen to register the following LC_MONETARY specification to reflect their local cultural conventions:

```
LC_MONETARY
% This is the official Lower Slobovian fdcc-set definition for
% the LC_MONETARY category.
%
valid_from          ;"19990916"
valid_to            "19981231";
conversion_rate     1;9
int_curr_symbol     "USD$";"USD!"
currency_symbol     "<tugrik>";<slobovik>"
% For <tugrik> substitute U+20AE.
% For <slobovik> substitute U+2445 (chosen to honor the
%    King of Lower Slobovia's sartorial style)
mon_decimal_point   "?"
mon_thousands_sep   "6"
% The use of "6" as a thousands separator was dictated by
% the king's astrologer for its felicitous sound, but has
% proven quite profitable when foreign computers unfamiliar
% with our conventions misinterpret it as a digit when
% parsing out tugriks and sloboviks.
mon_grouping        1;2;3;2;1
positive_sign       "-"
```

```
negative_sign       "+"
int_frac_digits     6;1
frac_digits         4;9
p_cs_precedes       0
p_sep_by_space      2
n_cs_precedes       0;1
n_sep_by_space      0;1
int_p_cs_precedes   1;0
int_p_sep_by_space  2;0
int_n_cs_precedes   0;1
int_n_sep_by_space  0;2
p_sign_posn         0;1
n_sign_posn         2;4
int_p_sign_posn     1;3
int_n_sign_posn     3
%
END LC_MONETARY
```

The implication for valid_from and valid_to is that the tugrik was valid from "the beginning of time" to 19981231, and that the slobovik is valid from 19990916 to "the end of time". (The metasyntax for valid_from and valid_to in fuzzy on this point, so this is just a stab at what it might mean to make use of those implicit infinities when defining more than own currency in the LC_MONETARY specification.) What an implementation will do when faced with a currency formatting for a date in between is unclear -- but perhaps that is o.k., since Lower Slobovia was in a state of financial chaos at the time, anyway.

O.k., now let's see what happens to the currency values when we use this definition to describe a money-laundering operation that had a credit of a little over 2 billion tugrik last year (2,000,600,609 to be exact) and then had to enter a negative value in the books for the same amount today. When you work it all out, the fully formatted currency strings come out to be:

Positive amount in 1998:

```
Local format:     "(260600660066069?0000 <tugrik>)"

Internat format:  "- USD$ 260600660066069?0"
```

Negative amount today:

```
Local format:     "<slobovik>+ 2622628869566?566666666"

Internat format:  "2622628869566?6+!USD"
```

Is this absurd? Well, of course. But it is *allowed* by the current specification. That means, in principle, that implementations of this specification have to be able to produce garbage like this without choking on various bizarre combinations of parameters. And legions of programmers are going to be scratching their heads over such things as what to do with the fourth character of the int_curr_symbol which "shall be the character used to separate the international currency symbol from the monetary quantity" when int_p_cs_precedes specifies that the currency sign goes to the *right* of the monetary quantity and int_p_sep_by_space specifies that a *space* is used for separation. Hmm.

To avoid this kind of nonsense, a usable specification should be constraining the allowable values for parameters and not allowing any possible combination in the forlorn hope of not offending the Lower Slobovians when they finally come to register their cultural conventions and find that their system wasn't accounted for by some constraint on allowed values.

It would be far, far better to specify in detail the *actual* cultural conventions for currency formatting, with a clear method for *describing* those conventions in detail. The IBM Green Book (August 1994, pp. 11 - 19) does so already in great detail in a far more useful format for implementers. That actual number of combinations in use is far, far less than an unconstrained metasyntax such as that of section 4.5 of PDTR 15642 allows. A better approach would be to simply define a hierarchy of:

A. Unsigned positive numeric formats.

B. Positive and negative signed numeric formats.

C. Currency sign placements in A. and B.

D. A list of known local currency signs with their country association and relation to official international banking signs.

From that, any competent software designer can create a mechanism for formatting and parsing currency strings that is adaptive to local cultural conventions. And it can be related to the TR's specification of values for A, B, C, and D above, so that one implementer of A:2;B:3;C:1;D:Pts can relate that to another implementation of the same format.

But PDTR 14652 is attempting something quite different -- it is attempting to *standardize* the Son-of-POSIX specification for LC_MONETARY to support portable implementations of localdef for Linux.

It is our opinion that the two goals--clear and effective exposition of cultural conventions versus extension of POSIX mechanisms for portable implementations on UNIX systems-- are effectively at odds, and that the technical quality of PDTR 14652 is suffering from mixing of incompatible goals.

Minor Technical Comments re Section 4.5 LC_MONETARY

p. 41, line 2616 and line 2621.
The definition of "the beginning of time" and "the end of time" is not clear here. As a cultural convention, these should just be UNSPECIFIED. It is then up to implementations to decide what to do about that. A reasonable option, of course, is to equate UNSPECIFIED time with either the first possible or last possible "time" value in a machine implementation of time, but that is platform-specific in terms of actual dates that get associated with those values.

p. 41, line 2622ff.

The convention for using two integers
in a specification of conversion rate also seems implementation-driven,
rather than designed for clarity. If a conversion rate is set
at 1.86301 (not at all unusual as a possibility), why should not
a cultural specification be able to use "1.86301" to express that,
rather than 186301;100000 ? This has to do with avoiding
float parsing complications on Unix rather than with clarity
of specification.

3. Editorial Comments

1. p. iv, line 84 cultural ==> culturally

2. p. iv, line 104 become ==> becomes

3. p. iv, line 116 backwards ==> upward (cf. usage on p. 1, line 137)

4. p. iv, line 117 particulary ==> particularly

5. p. 2, line 221 Change final quotation mark to "."

6. p. 4, line 290 represent ==> represents

7. p. 10, line 652 indicate ==> indicates

8. p. 20, line 1167 does ==> do

9. p. 29, line 1883 reorder-sections-after ==> reorder-section-after
   line 1884 reorder-sections-end ==> reorder-section-end

   (This mistake is made many times on subsequent pages. The entire document should be searched and all instances fixed. See, for example, p. 33, lines 2075, 2089; p. 37 multiple times, etc.)

10. p. 29, line 1888 referred as ==> referred to as

11. p. 30, line 1925 replace-after ==> reorder-after
    line 1934 (same mistake -- search entire document)

12. p. 30, line 1941 " specified by its place." Add "in the list of collating statements." to the end of the sentence.

13. p. 31, line 1962 ellipsises ==> ellipses

14. p. 33, line 2078 & line 2084 col_weight_max ==> coll_weight_max

15. p. 33, line 2097 with the ==> within the

    (Same error on p. 34, lines 2136 and 2157)

16. p. 34, line 2155 collating-symbol-2 ==> collating-symbol-1

17. p. 36, line 2236 on ==> in

18. p. 55, line 3305 vaule ==> value

19. p. 56, line 3391 start ==> starting

20. p. 58, line 3512 added the ==> added to the

21. p. 90, line 6091 done ==> made

22. p. 90, line 6129 introduce ==> introduced

23. p. 91, line 6132 elipsises ==> ellipses

Editorial Comments re Section 4.5 LC_MONETARY

p. 41, line 2601 "is taken" ==> "is implied"

p. 43, line 2680 For consistency, "int_curr_symbol" should be
in double quotes.

end of document