Information technology —
Specification method for cultural conventions

Technologies de l’information —
Méthode de modélisation des conventions culturelles
ISO/IEC TR 14652:2002(E)

This page left for ISO/IEC copyright notices.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOREWORD</td>
<td>iv</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>v</td>
</tr>
<tr>
<td>1 SCOPE</td>
<td>1</td>
</tr>
<tr>
<td>2 NORMATIVE REFERENCES</td>
<td>1</td>
</tr>
<tr>
<td>3 TERMS, DEFINITIONS AND NOTATIONS</td>
<td>2</td>
</tr>
<tr>
<td>4 FDCC-set</td>
<td>6</td>
</tr>
<tr>
<td>4.1 FDCC-set description</td>
<td>7</td>
</tr>
<tr>
<td>4.2 LC_IDENTIFICATION</td>
<td>11</td>
</tr>
<tr>
<td>4.3 LC_CTYPE (controversial)</td>
<td>12</td>
</tr>
<tr>
<td>4.4 LC_COLLATE</td>
<td>26</td>
</tr>
<tr>
<td>4.5 LC_MONETARY (controversial)</td>
<td>37</td>
</tr>
<tr>
<td>4.6 LC_NUMERIC</td>
<td>41</td>
</tr>
<tr>
<td>4.7 LC_TIME (controversial)</td>
<td>42</td>
</tr>
<tr>
<td>4.8 LCMESSAGES</td>
<td>49</td>
</tr>
<tr>
<td>4.9 LC_XLITERATE (controversial)</td>
<td>49</td>
</tr>
<tr>
<td>4.10 LC_NAME</td>
<td>51</td>
</tr>
<tr>
<td>4.11 LC_ADDRESS</td>
<td>53</td>
</tr>
<tr>
<td>4.12 LC_TELEPHONE</td>
<td>55</td>
</tr>
<tr>
<td>5 CHARMAP</td>
<td>56</td>
</tr>
<tr>
<td>6 REPERTOIREMAP (controversial)</td>
<td>61</td>
</tr>
<tr>
<td>Annex A (informative) DIFFERENCES FROM POSIX</td>
<td>89</td>
</tr>
<tr>
<td>Annex B (informative) RATIONALE</td>
<td>91</td>
</tr>
<tr>
<td>Annex C (informative) BNF GRAMMAR</td>
<td>104</td>
</tr>
<tr>
<td>Annex D (informative) OUTSTANDING ISSUES</td>
<td>109</td>
</tr>
<tr>
<td>Annex E (informative) INDEX</td>
<td>113</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>117</td>
</tr>
</tbody>
</table>
Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The main task of a technical committee is to prepare International Standards but in exceptional circumstances, the publication of a Technical Report of one of the following types may be proposed:

- type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

- type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

- type 3, when a technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Report of type 3 do not necessarily have to be reviewed until the date they provide are considered to be no longer valid or useful.

ISO/IEC TR 14652 is a Technical Report type 1, and it was prepared by Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee 22, Programming languages, their environments and system software interfaces.

This Technical Report is a TR type 1 as it failed the FCD.2 ballot in 1999 as recorded in ISO/IEC JTC 1/SC22 N2937, where a number of member bodies requested that it be published as a TR type 1, and the WG20 agreed to that. Concerns from a number of member bodies with the current text are recorded in Annex D.
Introduction

This Technical Report defines a general mechanism to specify cultural conventions, and it defines formats for a number of specific cultural conventions in the areas of character classification and conversion, sorting, number formatting, monetary formatting, date formatting, message display, addressing of persons, postal address formatting, and telephone number handling.

There are a number of benefits coming from this Technical Report:

**Rigid specification**
Using this Technical Report, a user can rigidly specify a number of the cultural conventions that apply to the information technology environment of the user.

**Cultural adaptability**
If an application has been designed and built in a culturally neutral manner, the application may use the specifications as data to its APIs, and thus the same application may accommodate different users in a culturally acceptable way to each of the users, without change of the binary application.

**Productivity**
This Technical Report specifies those cultural conventions and how to specify data for them. With that data an application developer is relieved from getting the different information to support all the cultural environments for the expected customers of the product. The application developer is thus ensured of culturally correct behaviour as specified by the customer, and possibly more markets may be reached as customers may have the possibility to provide the data themselves for markets that were not targeted.

**Uniform behaviour**
When a number of applications share one cultural specification, which may be supplied from the user or provided by the application or operating system, their behaviour for cultural adaptation becomes uniform.

The specification format is independent of platforms and specific encoding, and targeted to be usable from a wide range of programming languages.

A number of cultural conventions, such as spelling, hyphenation rules and terminology, are not specifiable with this Technical Report, but it provides mechanisms to define new categories and also new keywords within existing categories. An internationalized application may take advantage of information provided with the FDCC-set (such as the language) to provide further internationalized services to the user.

This Technical Report defines a format compatible with the one used in the International string ordering standard, ISO/IEC 14651. This Technical Report is upward compatible with the ISO/IEC 9945-2:1993 POSIX shell and utilities standard, particularly its clauses 2.4 and 2.5. The major extensions from that text are listed in annex A. This Technical Report has enhanced functionality in a number of areas such as ISO/IEC 10646 support, more
classification of characters, transliteration, dual (multi) currency support, enhanced date and time formatting, personal name writing, postal address formatting, telephone number handling, and management of categories. There is enhanced support for character sets including ISO/IEC 2022 handling and an enhanced method to separate the specification of cultural conventions from an actual encoding via a description of the character repertoire employed. A standard set of values for all the categories has been defined covering the repertoire of ISO/IEC 10646-1, as referenced in the normative references clause.

The Technical Report was originally scheduled for adoption as an International Standard, but a number of members of ISO and IEC found the specification problematical. It was then decided to convert the specification into a Technical Report type I. Annex D lists a number of issues that some members of ISO and IEC have with the specification.

The following clauses are thus marked as controversial:
4.3 LC_CTYPE
4.5 LC_MONETARY
4.7 LC_TIME
4.9 LC_XLITERATE
6 REPERTOIREMAP
Information technology —
Specification method for cultural conventions

1 SCOPE

This Technical Report specifies a description format for the specification of cultural
conventions, a description format for character sets, and a description format for binding
character names to ISO/IEC 10646, plus a set of default values for some of these items.

The specification is upward compatible with POSIX locale specifications - a locale
conformant to POSIX specifications will also be conformant to the specifications in this
Technical Report, while the reverse condition will not hold. 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. An
alignment effort has been undertaken for this specification to be aligned with the revision

2 NORMATIVE REFERENCES

The following normative documents contain provisions which, through reference in this
text, constitute provisions of this Technical Report. For dated references, subsequent
amendments to, or revisions of, any of these publications do not apply. However, parties to
agreements based on this Technical Report are encouraged to investigate the possibility of
applying the most recent editions of the normative documents indicated below. For undated
references, the latest edition of the normative document referred to applies. Members of
ISO and IEC maintain registers of currently valid Technical Reports.

ISO 639 (all parts), Codes for the representation of names of languages.

ISO/IEC 2022, Information technology - Character code structure and extension
techniques.

ISO 3166 (all parts), Codes for the representation of names of countries and their
subdivisions.

ISO 4217, Codes for the representation of currencies and funds.

ISO 8601, Data elements and interchange formats - Information interchange -
Representation of dates and times.

ISO/IEC 9945:200x (to be published), Information technology - Portable Operating System
Interface (POSIX).

(POSIX) - Part 2: Shell and Utilities.

ISO/IEC 10646-1:1993, Information technology - Universal Multiple-Octet Coded
Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane, including Cor.1
and AMD 1-9 plus AMD 18. From AMD 18 only the characters U20AC EURO SIGN and
UFFFFC OBJECT REPLACEMENT CHARACTER are accounted for in this TR.

comparing character strings and description of a default tailorable ordering.
3 TERMS, DEFINITIONS AND NOTATIONS

3.1 Terms and definitions

For the purposes of this Technical Report, the terms and definitions given in the following apply.

3.1.1 Bytes and characters

3.1.1.1 byte:
An individually addressable unit of data storage that is equal to or larger than an octet, used to store a character or a portion of a character.

A byte is composed of a contiguous sequence of bits, the number of which is implementation defined. The least significant bit is called the low-order bit; the most significant bit is called the high-order bit.

3.1.1.2 character:
A member of a set of elements used for the organization, control or representation of data.

3.1.1.3 coded character:
A sequence of one or more bytes representing a single character.

3.1.1.4 text file:
A file that contains characters organized into one or more lines.

3.1.2 cultural and other major concepts

3.1.2.1 cultural convention:
A data item for information technology that may vary dependent on language, territory, or other cultural habits.

3.1.2.2 FDCC
A Formal Definition of a Cultural Convention, that is a cultural convention put into a formal definition scheme.

3.1.2.3 FDCC-set:
A Set of Formal Definitions of Cultural Conventions (FDCC’s). The definition of the subset of a user’s information technology environment that depends on language and cultural conventions.

Note: the FDCC-set is a superset of the "locale" term in C and POSIX.
3.1.2.4 charmap:
A definition of a mapping between symbolic character names and character codes, plus related information.

3.1.2.5 repertoiremap:
A definition of a mapping between symbolic character names and characters for the repertoire of characters used in a FDCC-set, further described in clause 6.

3.1.3 FDCC categories related

3.1.3.1 character class:
A named set of characters sharing an attribute associated with the name of the class.

3.1.3.2 collation:
The logical ordering of strings according to defined precedence rules.

3.1.3.3 collating element:
The smallest entity used to determine logical ordering.

See collating sequence. A collating element consists of either a single character, or two or more characters collating as a single entity. The LC_COLLATE category in the associated FDCC-set determines the set of collating elements.

3.1.3.4 multicharacter collating element:
A sequence of two or more characters that collate as an entity.

For example, in some languages two characters are sorted as one letter, as in the case for Danish and Norwegian "aa".

3.1.3.5 collating sequence:
The relative order of collating elements as determined by the setting of the LC_COLLATE category in the applied FDCC-set.

3.1.3.6 equivalence class:
A set of collating elements with the same primary collation weight.

Elements in an equivalence class are typically elements that naturally group together, such as all accented letters based on the same letter.

The collation order of elements within an equivalence class is determined by the weights assigned on any subsequent levels after the primary weight.
3.2 Notations

The following notations and common conventions for specifications apply to this Technical Report:

3.2.1 Notation for defining syntax

In this Technical Report, the description of an individual record in a FDCC-set is done using the syntax notation given in the following.

The syntax notation looks as follows:

"<format>" ,[<arg1>,<arg2>,...,<argn>]

The <format> is given in a format string enclosed in double quotes, followed by a number of parameters, separated by commas. It is similar to the format specification defined in clause 2.12 in the ISO/IEC 9945-2:1993 standard and the format specification used in C language printf() function. The format of each parameter is given by an escape sequence as follows:

- %s specifies a string
- %d specifies a decimal integer
- %c specifies a character
- %o specifies an octal integer
- %x specifies a hexadecimal integer

A " " (an empty character position) in the syntax string represents one or more <blank> characters.

All other characters in the format string represent themselves, except:

- %  specifies a single %
- \n specifies an end-of-line

The notation "..." is used to specify that repetition of the previous specification is optional, and this is done in both the format string and in the parameter list.

3.2.3 Portable character set

A set of symbolic names for characters in Table 1, which is called the portable character set, is used in character description text of this specification. The first eight entries in Table 1 are defined in ISO/IEC 6429 and the rest is defined in ISO/IEC 9945-2 with some definitions from ISO/IEC 10646-1.

Table 1: Portable character set

<table>
<thead>
<tr>
<th>Symbolic name</th>
<th>Glyph</th>
<th>UCS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;NUL&gt;</td>
<td>&lt;U0000&gt;</td>
<td>NULL (NUL)</td>
<td></td>
</tr>
<tr>
<td>&lt;alert&gt;</td>
<td>&lt;U0007&gt;</td>
<td>BELL (BEL)</td>
<td></td>
</tr>
<tr>
<td>&lt;backspace&gt;</td>
<td>&lt;U0008&gt;</td>
<td>BACKSPACE (BS)</td>
<td></td>
</tr>
<tr>
<td>&lt;tab&gt;</td>
<td>&lt;U0009&gt;</td>
<td>CHARACTER TABULATION (HT)</td>
<td></td>
</tr>
<tr>
<td>&lt;carriage-return&gt;</td>
<td>&lt;U000D&gt;</td>
<td>CARRIAGE RETURN (CR)</td>
<td></td>
</tr>
<tr>
<td>&lt;newline&gt;</td>
<td>&lt;U000A&gt;</td>
<td>LINE FEED (LF)</td>
<td></td>
</tr>
<tr>
<td>&lt;vertical-tab&gt;</td>
<td>&lt;U000B&gt;</td>
<td>LINE TABULATION (VT)</td>
<td></td>
</tr>
</tbody>
</table>
This Technical Report may use other symbolic character names than the above in examples, to illustrate the use of the range of symbols allowed by the syntax specified in 4.1.1.

4 FDCC-set

A FDCC-set is the definition of the subset of a user’s information technology environment that depends on language and cultural conventions. It is made up from one or more categories. Each category is identified by its name and controls specific aspects of the behaviour of components of the system. This Technical Report defines the following categories:

- **LC_IDENTIFICATION** Versions and status of categories
- **LC_CTYPE** Character classification, case conversion and code transformation.
- **LC_COLLATE** Collation order.
- **LC_TIME** Date and time formats.
- **LC_NUMERIC** Numeric, non-monetary formatting.
- **LC_MONETARY** Monetary formatting.
- **LC_MESSAGES** Formats of informative and diagnostic messages and interactive responses.
- **LC_XLITERATE** Character transliteration.
- **LC_NAME** Format of writing personal names.
- **LC_ADDRESS** Format of postal addresses.
- **LC_TELEPHONE** Format for telephone numbers, and other telephone information.

Note: In future editions of this Technical Report further categories may be added.
Other category names beginning with the 3 characters "LC_" are reserved for future standardization, except for category names beginning with the five characters "LC_X_" which is not used for future addition of categories specified in this Technical Report. An application may thus use category names beginning with the five characters "LC_X_" for application defined categories to avoid clashes with future standardized categories.

This Technical Report also defines an FDCC-set named "i18n" with values for some of the above categories in order to simplify FDCC-set descriptions for a number of cultures. The contents of "i18n" categories should not necessarily be considered as the most commonly accepted values, while in many cases it could be the recommended values. The complete "i18n" FDCC-set is defined as the sum of the "i18n" categories specified in the clauses below.

4.1 FDCC-set description

FDCC-sets are described with the syntax presented in this subclause. For the purposes of this Technical Report, the text is referred to as the FDCC-set definition text or FDCC-set source text.

The **FDCC-set definition text** contains one or more FDCC-set category source definitions, and does not contain more than one definition for the same FDCC-set category. If the text contains source definitions for more than one category, application-defined categories, if present, appears after the categories defined by this clause. A category source definition contains either the definition of a category or a copy directive. In the event that some of the information for a FDCC-set category, as specified in this Technical Report, is missing from the FDCC-set source definition, the behaviour of that category, if it is referenced, is unspecified. A FDCC-set category is the normal way of specifying a single FDCC.

There are no **naming conventions** for FDCC-sets specified in this Technical Report, but clause 6.8 in ISO/IEC 15897:1999 specifies naming rules for POSIX locales, charmaps and repertoiremaps, that may also be applied to FDCC-sets, charmaps and repertoiremaps specified according to this Technical Report.

A **category source definition** consists of a category header, a category body, and a category trailer. A category header consists of the character string naming of the category, beginning with the characters "LC_.". The category trailer consists of the string "END", followed by one or more "blank"s and the string used in the corresponding category header.

The **category body** consists of one or more lines of text. Each line is one of the following:

- a line containing an identifier, optionally followed by one or more operands. Identifiers are either keywords, identifying a particular FDCC, or collating elements, or section symbols,
- one of transliteration statements defined in 4.3.

In addition to the keywords defined in this Technical Report, the source can contain application-defined keywords. Each **keyword** within a category has a unique name (i.e., two categories can have a commonly-named keyword); no keyword starts with the characters "LC_.". Identifiers are separated from the operands by one or more "blank"s.

**Operands** are characters, collating elements, section symbols, or strings of characters.
Strings are enclosed in double-quotes. Literal double-quotes within strings are preceded by the <escape character>, described below. When a keyword is followed by more than one operand, the operands are separated by semicolons; "blank"s are allowed before and/or after a semicolon.

4.1.1 Character representation

Individual characters, characters in strings, and collating elements are represented using symbolic names, UCS notation or characters themselves, or as octal, hexadecimal, or decimal constants as defined below. When constant notation is used, the resultant FDCC-set definitions need not be portable between systems.

(0) The left angle bracket (<) is a reserved symbol, denoting the start of a symbolic name; when used to represent itself outside a symbolic name it is preceded by the escape character.

(1) A character can be represented via a symbolic name, enclosed within angle brackets (< and >). The symbolic name, including the angle brackets, exactly matches a symbolic name defined in a charmap or a repertoiremap to be used, and is replaced by a character value determined from the value associated with the symbolic name in the charmap or a value associated via a repertoiremap. Repertoiremaps have predefined symbolic names for UCS characters, see clause 6. A FDCC-set may also use the UCS notation of clause 6 to represent characters, without a repertoiremap being defined for the FDCC-set. Use of the escape character or a right angle bracket within a symbolic name is invalid unless the character is preceded by the escape character.

Example: `<c>;<c-cedilla> "<M><a><y>

The items (2), (3), (4) and (5) are deprecated and are retained for compatibility with the POSIX standard. FDCC-sets should be specified in a coded character set independent way, using symbolic names. To make actual use of the FDCC-set, it is used together with charmaps and/or repertoiremaps, so that the symbolic character names can be resolved into the actual character encoding used.

(2) A character can be represented by the character itself, in which case the value of the character is application-defined. Within a string, the double-quote character, the escape character, and the right angle bracket character are escaped (preceded by the escape character) to be interpreted as the character itself. Outside strings, the characters

, ; < > escape_char

are escaped by the escape character to be interpreted as the character itself.

Example: c ä "May"
A character can be represented as an octal constant. An octal constant is specified as the escape character followed by two or more octal digits. Each constant represents a byte value.

Example: \143; \347; "\115"

A character can be represented as a hexadecimal constant. A hexadecimal constant is specified as the escape character followed by an x followed by two or more hexadecimal digits. Each constant represents a byte value.

Example: \x63; \xe7;

A character can be represented as a decimal constant. A decimal constant is specified as the escape character followed by a d followed by two or more decimal digits. Each constant represents a byte value.

Example: \d99; \d231;

Multibyte characters can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character. Concatenated constants can include a mix of the above character representations.

Example: \143xe7; "\115xe7d171"

Only characters existing in the character set for which the FDCC-set definition is created are specified, whether using symbolic names, the characters themselves, or octal, decimal, or hexadecimal constants. If a charmap is present, only characters defined in the charmap can be specified using octal, decimal, or hexadecimal constants. Symbolic names not present in the charmap can be specified and are ignored, as specified under item (1) above.

Note: The <character> symbolic character notation is recommended for use of specifying all characters in a FDCC-set, to facilitate portability of the FDCC-sets, as the coded character set of the application of the FDCC-set may be different from the coded character set of the FDCC-set source. This is also recommended for format effectors in strings, such as in LC_DATE or LC_ADDRESS, where the format effectors are allowed to be stored together with the rest of the string, in a binary string with a different encoding from that of the source FDCC-set.

4.1.2 Continuation of lines

A line in a specification can be continued by placing an escape character as the last visible graphic character on the line; this continuation character is discarded from the input. The line is continued to the next non-comment line.

4.1.3 Names for copy keyword

In most of the categories a "copy" keyword is allowed. The name specified with this copy keyword is one of:

- "i18n" which indicate the "i18n" FDCC-set defined in this specification,
- the name of a FDCC-set or POSIX locale registered by the process defined in ISO/IEC 15897,
- any other name which may be recognized in some local context - not being recommended as an international specification.

4.1.4 Pre-category statements

In a FDCC-set the following statements can precede category specifications, and they apply to all categories in the specified FDCC-set.

4.1.4.1 comment_char

The following line in a FDCC-set modifies the comment character. It has the following syntax, starting in column 1:

"comment_char %c\n", <comment_character>

The comment character defaults to the number-sign (#). All examples in this Technical Report use "%" as the <comment_character>, except where otherwise noted. Blank lines and lines containing the <comment_character> in the first position are ignored. In collating statements a <comment_character> occurring where the delimiter ";" may occur, terminates the collating statement.

4.1.4.2 escape_char

The following line in a FDCC-set modifies the escape character to be used in the text. It has the following syntax, starting in column 1:

"escape_char %c\n", <escape_character>

The escape character is used for representing characters in 4.1.1 and for continuing lines. The escape character defaults to backslash "/". All examples in this Technical Report uses "/" as the escape character, except where otherwise noted.

4.1.4.3 repertoiremap

The following line in a FDCC-set specifies the name of a repertoiremap used to define the symbolic character names in the FDCC-set. There may be at most one "repertoiremap" line. It has the following syntax, starting in column 1:

"repertoiremap %s\n", <repertoiremap>

The name is one of:
- "i18nrep" which indicates the "i18nrep" repertoiremap defined in this specification,
- the name of a <repertoiremap> registered by the process defined in ISO/IEC 15897,
- any other name which may be recognized in some local context - not being recommended as an international specification.

4.1.4.4 charmap

The following line in a FDCC-set specifies the name of a charmap which may be used with the FDCC-set. It has the following syntax, starting in column 1:
"charmap %s\n",<charmap>

This keyword gives a hint on which charmaps a FDCC-set is meant to be supported by. There may be more than one charmap specification useful with a FDCC-set. It is an application’s responsibility to decide what charmap specification is to be used with that application.

The name is one of:
- the name of a <charmap> registered by the process defined in ISO/IEC 15897,
- any other name which may be recognized in some local context - not being recommended as an international specification.

4.2 LC_IDENTIFICATION

The LC_IDENTIFICATION category defines properties of the FDCC-set, and which specification methods the FDCC-set is conforming to. Values must be supplied for all unless otherwise noted, and the operands are strings. The following keywords are defined:

- title: Title of the FDCC-set.
- source: Organization name of provider of the source.
- address: Organization postal address.
- contact: Name of contact person. This keyword is optional.
- email: Electronic mail address of the organization, or contact person. This keyword is optional.
- tel: Telephone number for the organization, in international format. This keyword is optional.
- fax: Fax number for the organization, in international format. This keyword is optional.
- language: Natural language to which the FDCC-set applies, as specified in ISO 639. If a two-letter code exists for this language, it is used, else the three-letter code is used. This keyword is optional.
- territory: The geographic extent where the FDCC-set applies (where applicable), as two-letter form of ISO 3166. This keyword is optional.
- audience: If not for general use, an indication of the intended user audience. This keyword is optional.
- application: If for use of a special application, a description of the application. This keyword is optional.
- abbreviation: Short name for provider of the source. This keyword is optional.
- revision: Revision number consisting of digits and zero or more full stops (".").
- date: Revision date in the format according to this example: "1995-02-05" meaning the 5th of February, 1995.

If required information is not present in ISO 639 or ISO 3166, the string should be given as empty, and the relevant Maintenance Authority should be approached to get the needed item registered.

Note: Only one language per territory can be addressed with a single FDCC-set; an additional FDCC-set is required for each additional language for that territory.
category  Is used to define that a category is present and what specification the category is claiming conformance to. The first operand is a string in double-quotes that describes the specification that the category is claiming conformance to, and the following values are defined:
"i18n:2002"
"posix:1993"
The second operand is a string with the category name, where the category names of clause 4 are defined. More than one "category" keyword may be given, but only one per category name.

The "i18n" LC_IDENTIFICATION category is:

LC_IDENTIFICATION
% This is the ISO/IEC TR 14652 "i18n" definition for % the LC_IDENTIFICATION category.
% title                 "ISO/IEC TR 14652 i18n FDCC-set"
source                "ISO/IEC Copyright Office"
address               "Case postale 56, CH-1211 Geneve 20, Switzerland"
contact               "" email               ""
tel                   "" fax                   ""
language              "" territory             ""
revision              "1.0" date                  "2001-12-08"
% category  "i18n:2002";LC_IDENTIFICATION
category  "i18n:2002";LC_CTYPE
category  "i18n:2002";LC_COLLATE
category  "i18n:2002";LC_TIME
category  "i18n:2002";LC_NUMERIC
category  "i18n:2002";LC_MONETARY
category  "i18n:2002";LC_MESSAGES
category  "i18n:2002";LC_NAME
category  "i18n:2002";LC_ADDRESS
category  "i18n:2002";LC_TELEPHONE

END LC_IDENTIFICATION

4.3 LC_CTYPE (controversial)

The LC_CTYPE category defines character classification, case conversion, character transformation, and other character attribute mappings. Support for the portable character set is required.

A series of characters in a specification can be represented by the hexadecimal symbolic ellipsis symbol ".." (two dots), the decimal symbolic ellipses symbols "...." (4 dots), or the absolute ellipses "..." (3 dots).

The hexadecimal symbolic ellipsis ("..") specification is only valid between symbolic character names. The symbolic names consists of zero or more nonnumeric characters from the set shown with visible glyphs in Table 1 of clause 3.2.3, followed by an integer formed by one or more hexadecimal digits, using uppercase letters only for the range "A" to "F". The characters preceding the hexadecimal integer are identical in the two symbolic names,
and the integer formed by the hexadecimal digits in the second symbolic name are identical to or greater than the integer formed by the hexadecimal digits in the first name. This is interpreted as a series of symbolic names formed from the common part and each of the integers in hexadecimal format using uppercase letters only between the first and the second integer, inclusive, and with a length of the symbolic names generated that is equal to the length of the first (and also the second) symbolic name. As an example, \texttt{<U010E>..<U0111>} is interpreted as the symbolic names \texttt{<U010E>}, \texttt{<U010F>}, \texttt{<U0110>}, and \texttt{<U0111>}, in that order.

The **decimal symbolic ellipsis** ("....") specification is only valid between symbolic character names. The symbolic names consist of zero or more nonnumeric characters from the set shown with visible glyphs in Table 1 of clause 3.2.3, followed by an integer formed by one or more decimal digits. The characters preceding the decimal integer are identical in the two symbolic names, and the integer formed by the decimal digits in the second symbolic name is identical to or greater than the integer formed by the decimal digits in the first name. This is interpreted as a series of symbolic names formed from the common part and each of the integers in decimal format between the first and the second integer, inclusive, and with a length of the symbolic names generated that is equal to the length of the first (and also the second) symbolic name. As an example, \texttt{<j0101>....<j0104>} is interpreted as the symbolic names \texttt{<j0101>}, \texttt{<j0102>}, \texttt{<j0103>}, and \texttt{<j0104>}, in that order.

The **absolute ellipsis** specification is only valid within a single encoded character set. An ellipsis is interpreted as including in the list all characters with an encoded value higher than the encoded value of the character preceding the ellipsis and lower than the encoded value of the character following the ellipsis. The absolute ellipsis specification is deprecated, as this is only relevant to FDCC-sets not using symbolic characters. As an example, \texttt{\x30;...;\x39} includes in the character class all characters with encoded values between the endpoints.

### 4.3.1 Character classification keywords

The following keywords are recognized. In the descriptions, the term "automatically included" means that it is not an error to either include the referenced characters or to omit them; the interpreting system provides them if missing and accept them silently if present.

- **copy**  Specify the name of an existing FDCC-set to be used as the source for the definition of this category. If this keyword is specified, no other keyword is specified.

- **upper**  Define characters to be classified as uppercase letters. No character specified for the keywords "cntrl", "digit", "punct", or "space" is specified. The uppercase letters A through Z of the portable character set, automatically belong to this class, with application-defined character values. The keyword may be omitted.

- **lower**  Define characters to be classified as lowercase letters. No character specified for the keywords "cntrl", "digit", "punct", or "space" is specified. The lowercase letters a through z of the portable character set, automatically belong to this class, with application-defined character values. The keyword may be omitted.

- **alpha**  Define characters to be classified as used to spell out the words for natural languages; such as letters, syllabic or ideographic characters. No character specified for the keywords "cntrl", "digit", "punct", or "space" is specified.
In addition, characters classified as either "upper" or "lower" automatically belong to this class. The keyword may be omitted.

**digit**
Define the characters to be classified as decimal 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, and in ascending order of the values they represent. The digits of the portable character set are automatically included. 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. The "digit" keyword is used to specify which characters are accepted as digits in input to an application, such as characters typed in or scanned in from an input text file, and should list digits used with all the scripts supported by the FDCC-set. The keyword may be omitted.

**alnum**
Define the characters to be classified as used to spell out the words for natural languages, and numeric digits. The characters of the "alpha" and "digit" classes are automatically included in this class. The keyword may be omitted.

**outdigit**
Define the characters to be classified as decimal digits for output from an application, such as to a printer or a display or a output text file. Decimal 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 decimal digits are used for output, including numeric and monetary formatting, and date and time formatting. Only one set of 10 decimal digits may be specified. If this keyword is not specified, the decimal digits 0 through 9 of the portable character set automatically belong to this class, with application-defined character values. The keyword may be omitted.

**blank**
Define characters to be classified as "blank" characters. If this keyword is unspecified, the characters <space> and <tab>, with application-defined character values, belong to this character class.

**space**
Define characters to be classified as white-space characters, to find syntactical boundaries. No character specified for the keywords "upper", "lower", "alpha", "digit", "graph", or "xdigit" is specified. If this keyword is not specified, the characters <space>, <form-feed>, <newline>, <carriage-return>, <tab>, and <vertical-tab>, automatically belong to this class, with application-defined character values. Any characters included in the class "blank" are automatically included. The class should not include the NO-BREAK spaces characters <U00A0>, <U2007>, <UFEFF>, as these characters should not be used for word boundaries. The keyword may be omitted.

**cntrl**
Define characters to be classified as control characters. No character specified for the keywords "upper", "lower", "alpha", "digit", "punct", "graph", "print", or "xdigit" is specified. If this keyword is not specified, the characters <space>, <form-feed>, <newline>, <carriage-return>, <tab>, and <vertical-tab>, automatically belong to this class, with application-defined character values. Any characters included in the class "blank" are automatically included. The class should not include the NO-BREAK spaces characters <U00A0>, <U2007>, <UFEFF>, as these characters should not be used for word boundaries. The keyword may be omitted.

**punct**
Define characters to be classified as punctuation characters. No character specified for the keywords "upper", "lower", "alpha", "digit", "cntrl", "xdigit", or as the <space> character is specified. The keyword is specified.

**xdigit**
Define the characters to be classified as hexadecimal digits. Only the characters defined for the class "digit" are specified, in ascending sequence by numerical value, followed by sets of six characters representing the hexadecimal digits 10 through 15 in ascending order (for example <A>, <B>, <C>, <D>, <E>, <F>, <a>, <b>, <c>, <d>, <e>, <f>). 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.

**graph**

Define characters to be classified as printable characters, not including the <space> character. If this keyword is not specified, characters specified for the keywords "upper", "lower", "alpha", "digit", "xdigit", and "punct" belong to this character class. No character specified for the keyword "cntrl" is specified.

**print**

Define characters to be classified as printable characters, including the <space> character. If this keyword is not provided, characters specified for the keywords upper, lower, alpha, digit, xdigit, punct, graph, and the <space> character belong to this character class. No character specified for the keyword "cntrl" is specified.

**toupper**

Define the mapping of lowercase letters to uppercase letters. The operand consists of character pairs, separated by semicolons. The characters in each character pair are separated by a comma and the pair enclosed by parentheses. The first character in each pair is the lowercase letter, the second the corresponding uppercase letter. Only characters specified for the keywords "lower" and "upper" are specified. If this keyword is not specified, the lowercase letters <a> through <z>, and their corresponding uppercase letters <A> through <Z>, are automatically included, with application-defined character values.

**tolower**

Define the mapping of uppercase letters to lowercase letters. The operand consists of character pairs, separated by semicolons. The characters in each character pair are separated by a comma and the pair enclosed by parentheses. The first character in each pair is the uppercase letter, the second the corresponding lowercase letter. Only characters specified for the keywords "lower" and "upper" are specified. If this keyword is specified, the uppercase letters <A> through <Z>, and their corresponding lowercase letters <a> through <z>, are automatically included, with the reverse mapping of the one specified for toupper. If this keyword is not specified, the mapping is the reverse mapping of the one specified for toupper.

**class**

(Controversial) 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. The class names "upper", "lower", "alpha", "digit", "space", "cntrl", "punct", "graph", "print", "xdigit", and "blank" are taken to mean the classes defined by the respective keywords.

**width**

(Controversial) 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.

**map**

(Controversial) Define the mapping of characters to other characters. The first operand is a string, defining the name of the mapping. The string only contains letters, digits and `<hyphen-minus>` and `<low-line>` from the portable character set. The following operands consist of character pairs, separated by semicolons. The characters in each character pair are separated by a comma and the pair enclosed by parentheses. The first character in each pair is the character to map from, the second the corresponding character to map to. This keyword is optional. The keyword can only be specified once per named mapping.

The mapping names "toupper", and "tolower" are taken to mean the mapping defined by the respective keywords.

Example of use of the "map" keyword:

```
map "kana",(<U30AB>,<U304B>);(<U30AC>,<U304C>);(<U30AD>,<U304D>)
```

This example introduces a new mapping "kana" that maps three Katakana characters to corresponding Hiragana characters.

Table 2 shows the allowed character class combinations.

<table>
<thead>
<tr>
<th>Class</th>
<th>upper</th>
<th>lower</th>
<th>alpha</th>
<th>digit</th>
<th>space</th>
<th>cntrl</th>
<th>punct</th>
<th>graph</th>
<th>print</th>
<th>xdigit</th>
<th>blank</th>
</tr>
</thead>
<tbody>
<tr>
<td>upper</td>
<td>+ A</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>A</td>
<td>A</td>
<td>+</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lower</td>
<td>+ A</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>A</td>
<td>A</td>
<td>+</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alpha</td>
<td>+</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>A</td>
<td>A</td>
<td>+</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>digit</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>space</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>x</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>cntrl</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>punct</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>+</td>
<td>x</td>
<td>A</td>
<td>x</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>graph</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>x</td>
<td>+</td>
<td>A</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>print</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>x</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>xdigit</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>A</td>
<td>A</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>blank</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>A</td>
<td>+</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Explanation of codes:
A Automatically included; see text
+ Permitted
x Mutually exclusive
* See note 2

Note 2: The `<space>` character, which is part of the "space" and "blank" class, cannot belong to "punct" or "graph", but automatically belong to the "print" class. Other "space" or "blank" characters can be classified as "punct", "graph", and/or "print".

4.3.2 "i18n" LC_CTYPE category

The "i18n" FDCC-set for the LC_CTYPE is defined as follows:

```
LC_CTYPE
% The following is the ISO/IEC TR 14652 i18n fdcc-set LC_CTYPE category.
% It covers ISO/IEC 10646-1 including Cor.1 and AMD 1 thru 9
% COLLECTION numbers and names are from ISO/IEC 10646-1 Annex A
```
% in extended identifiers.
alpha /
% COLLECTION 1 BASIC LATIN/
<U0041>..<U005A>;
% COLLECTION 2 LATIN-1 SUPPLEMENT/
<U0061>..<U007A>;
% COLLECTION 3 LATIN EXTENDED-A/
<U0100>..<U017F>;
% COLLECTION 4 LATIN EXTENDED-B/
<U0180>..<U01AF>;
% COLLECTION 5 IPA EXTENSIONS/
<U0250>..<U02A8>;
% COLLECTION 30 LATIN EXTENDED ADDITIONAL/
<U0E00>..<U0E07>;
% COLLECTION 33 SUPERSCRIPTS AND SUBSCRIPTS/
<U2070>..<U207B>;
% COLLECTION 8 BASIC GREEK/
<U0386>..<U03B0>;
% COLLECTION 9 GREEK SYMBOLS AND COPTIC/
<U03D0>..<U03F5>;
% COLLECTION 10 CYRILLIC/
<U0401>..<U04FF>;
% COLLECTION 11 ARMENIAN/
<U0531>..<U05C0>;
% COLLECTION 13 HEBREW EXTENDED/
<U05B5>..<U05CB>;
% COLLECTION 15 ARABIC EXTENDED/
<U0621>..<U06FF>;
% COLLECTION 16 DEVANAGARI/
<U0900>..<U097F>;
% COLLECTION 17 BENGALI/
<U0980>..<U09FF>;
% COLLECTION 18 GURMUKHI/
<U0A00>..<U0A7B>;
% COLLECTION 19 GUJARATI/
<U0A80>..<U0B05>;
% COLLECTION 20 ORIYA/
<U0B06>..<U0B0F>;
% COLLECTION 21 TAMIL/
<U0B20>..<U0C0F>;
% COLLECTION 22 TELUGU/
<U0C10>..<U0D2F>;
% COLLECTION 23 KANNADA/
<U0C80>..<U0D2F>;
% COLLECTION 24 MALAYALAM/
<U0D20>..<U0D7F>;
% COLLECTION 25 THAI/
<U0E00>..<U0E7F>;
4.4 LC_COLLATE

A collation sequence definition defines the relative order between collating elements (characters and multicharacter collating elements) in the FDCC-set. This order is expressed in terms of collation values; i.e., by assigning each element one or more collation values (also known as collation weights). This does not imply that applications assign such values, but that ordering of strings using the resultant collation definition in the FDCC-set behaves as if such assignment is done and used in the collation process. The collation sequence definition is used by regular expressions, pattern matching. When no weights are specified the collation sequence definition also is used for sorting, else the weighting defines the sorting. The following capabilities are provided:
(1) Multicharacter collating elements. Specification of multicharacter collating elements (i.e., sequences of two or more characters to be collated as an entity).

(2) User-defined ordering of collating elements. Each collating element is assigned a collation value defining its order in the character (or basic) collation sequence. This ordering is used by regular expressions and pattern matching and, unless collation weights are explicitly specified, also as the collation weight to be used in sorting.

(3) Multiple weights and equivalence classes. Collating elements can be assigned one or more (up to the limit (COLL_WEIGHTS_MAX)) collating weights for use in sorting. The first weight is hereafter referred to as the primary weight.

(4) One-to Many mapping. A single character is mapped into a string of collating elements.

(5) Many-to-Many substitution. A string of one or more characters is substituted by another string (or an empty string, i.e., the character or characters are ignored for collation purposes).

(6) Equivalence class definition. Two or more collating elements have the same collation value (primary weight).

(7) Ordering by weights. When two strings are compared to determine their relative order, the two strings are first broken up into a series of collating elements, and each successive pair of elements are compared according to the relative primary weights for the elements. If equal, and more than one weight has been assigned, then the pairs of collating elements are recompared according to the relative subsequent weights, until either a pair of collating elements compare unequal or the weights are exhausted.

(8) Easy reordering of characters. ISO/IEC 14651 has a template for collation specification that with just a few modifications can be culturally correct for a specific culture. Here the "reorder-after" keyword gives a convenient way to modify a FDCC-set template.

(9) Easy reordering of sections. The template in ISO/IEC 14651 gives an ordering of the sections that may not be culturally acceptable in certain cultures. The keyword "reorder-section-after" gives a convenient way to modify the order of sections in a FDCC-set template.

The following keywords are recognized in a collation sequence definition. Some of them are described in detail in the following subclauses. The keywords are mandatory unless otherwise noted.

**copy**
Specify the name of an existing FDCC-set to be used as the source for the definition of this category. If this keyword is specified, only the "reorder-after", "reorder-end", "reorder-section-after" and "reorder-section-end" keywords may also be specified. The FDCC-set is copied in source form.

**coll_weight_max**
Define as a decimal number the number of collation levels that an interpreting system needs to support for this FDCC-set, this value is elsewhere referred to as the COLL_WEIGHT_MAX limit (e.g. in the "order_start" statement). An interpreting system caters for up to 7 collating levels.

**section-symbol**
Define a section symbol representing a set of collation order statements. The section is defined with the "order_start" keyword until the next "order_start" or "order_end" keyword. This keyword is optional.
4.4.1 Collation statements

The "order_start", "reorder-after" and "section" keywords are followed by collating statements. The syntax for the collating statements is

"%s %s;%s;...;%s
",<collating-identifier>,<weight>,<weight>,...

Each <collating-identifier> consists of either a character (in any of the forms defined in 4.1.1), a <collating-element>, a <collating-symbol>, an ellipsis, or the special symbol "UNDEFINED". The weights for each of the collation elements determines the character collation sequence - such that each collation statement does not need to be in collation order, and weights could be rearranged via for example the "reorder-after" keyword. No character has any specific predetermined placement in the collation sequence. The order in which collating elements are specified determines the character collation sequence, such that each collating element compares less than the elements following it.

A <collating-element> is used to specify multicharacter collating elements, and indicates that the character sequence specified via the <collating-element> is to be collated as a unit and in the relative order specified by its place in the list of collating statements.

A <collating-symbol> is used to define a position in the relative order for use in weights.
The absolute ellipsis symbol ("...") specifies that a sequence of characters collate according to their encoded character values. It is interpreted as indicating that all characters with a coded character set value higher than the value of the character in the preceding line, and lower than the coded character set value for the character in the following line, in the current coded character set, are placed in the character collation order between the previous and the following character in ascending order according to their coded character set values. An initial ellipsis is interpreted as if the preceding line specified the <NUL> character, and a trailing ellipsis as if the following line specified the highest coded character set value in the current coded character set. An ellipsis is treated as invalid if the preceding or following lines do not specify characters in the current coded character set. The use of the ellipsis symbol ties the definition to a specific coded character set and may preclude the definition from being portable between applications, and is depreciated. Symbolic ellipses may be used as the ellipses symbol, but generating symbolic character names, and thus have a better chance of portability between applications.

The symbolic ellipses (".." or "....") specifies a sequence of collating statements. It is interpreted as indicating that all characters with symbolic names higher than the symbolic name of the character in the preceding line, and lower in the sequence of symbolic names for the character in the following line, is placed in the character collation order between the previous and the following character in ascending order.

The symbol "UNDEFINED" is interpreted as including all coded character set values not specified explicitly or via the ellipsis or one of the symbolic ellipses symbols. Such characters are inserted in the character collation order at the point indicated by the symbol, and in ascending order according to their coded character set values. If no "UNDEFINED" symbol is specified, and the current coded character set contains characters not specified in this clause, the utility issues a warning message and place such characters at the end of the character collation order.

The optional operands for each collation-element are used to define the primary, secondary, or subsequent weights for the collating element. The first operand specifies the relative primary weight, the second the relative secondary weight, and so on. Two or more collation-elements can be assigned the same weight; they belong to the same equivalence class if they have the same primary weight. Collation behaves as if, for each weight level, "IGNORE"d elements are removed. Then each successive pair of elements is compared according to the relative weights for the elements. If the two strings compare equal, the process is repeated for the next weight level, up to the limit "COLL_WEIGHTS_MAX" of the associated FDCC-set.

Weights are expressed as characters (in any of the forms specified here), <collating-symbol>s, <collating-element>s, an ellipsis, or the special symbol "IGNORE". A single character, a <collating-symbol>, or a <collating-element> represent the relative order in the character collating sequence of the character or symbol, rather than the character or characters themselves.

One-to-many mapping is indicated by specifying two or more concatenated characters or symbolic names. Thus, if the character <ss> is given the string <s><s> as a weight, comparisons are performed as if all occurrences of the character <ss> are replaced by <s><s>. If it is desirable to define <ss> and <s><s> as an equivalence class, then a collating-element must be defined for the string "ss", as in the example below.
All characters specified via an ellipsis are by default assigned unique weights, equal to the relative order of characters. Characters specified via an explicit or implicit "UNDEFINED" special symbol are by default assigned the same primary weight (i.e., belong to the same equivalence class). An ellipsis symbol as a weight is interpreted to mean that each character in the sequence has unique weights, equal to the relative order of their character in the character collation sequence. Secondary and subsequent weights have unique values. The use of the ellipsis as a weight is treated as an error if the collating element is neither an ellipsis nor the special symbol "UNDEFINED".

The special keyword "IGNORE" as a weight indicates that when strings are compared using the weights at the level where "IGNORE" is specified, the collating element is ignored; i.e., as if the string did not contain the collating element. In regular expressions and pattern matching, all characters that are "IGNORE"d in their primary weight form an equivalence class.

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

An empty operand is interpreted as the collating-element itself.

For example, the collation statement

```
<a>    <a>;<a>
```

is equal to

```
<a>
```

An ellipsis (absolute or symbolic) can be used as an operand if the collating-element was an ellipsis, and is interpreted as the value of each character defined by the ellipsis.

```
Example:

collating-element <ch> from "<c><h>"
collating-element <Ch> from "<C><h>"
order_start    forward;backward
UNDEFINED      IGNORE;IGNORE
<LOW>
<space>        <LOW>;<space>
...            <LOW>;
<a>            <a>;<a>
<a'>           <a>;<a'>
<A>            <a>;<A>
<A'>           <a>;<A'>
<ch>           <ch>;<ch>
<Ch>           <ch>;<Ch>
<s>            <s>;<s>
<ss>           "<s><s>";"<ss><ss>"

order_end
```

This example is interpreted as follows:

1. The UNDEFINED means that all characters not specified in this definition (explicitly or via the ellipsis) is ignored.
2. "<LOW>" defines the first collating weight, and thus the lowest weight in this example.
3. All characters between <space> and <a> have the same primary equivalence class <LOW> and individual secondary weights based on their ordinal encoded values. (The use of absolute ellipses is deprecated, but used here to illustrate generic use of ellipses. Symbolic ellipses should be used instead).
4. All characters based on the upper or lowercase character "a" belong to the same primary equivalence class.
(5) The multicharacter collating element \(<c><h>\) is represented by the collating symbol \(<ch>\) and belongs to the same primary equivalence class as the multicharacter collating element \(<C><h>\).

(6) The \(<ss>\) collating element has two weights on the primary level, and it is in the same primary equivalence class as two consecutive \(<s>-es; on the secondary level the collating element has two weights of the equivalence class \(<ss>\).

4.4.2 "copy" keyword

This keyword specifies the name of an existing FDCC-set to be used as the source for the definition of this category. The syntax is

"copy %s\n", <FDCC-set-name>

The <FDCC-set-name> consists of one or more characters (in any of the forms defined in 4.1.1). The FDCC-set is copied in source form.

4.4.3 "coll_weight_max" keyword

This keyword defines as a decimal number the number of collation levels that an interpreting system needs to support. An interpreting system caters for up to 7 collating levels. The syntax is

"coll_weight_max %d\n", <value>

4.4.4 "section-symbol" keyword

This keyword is used to define symbols for use in section related statements; such as the "order_start", and "reorder-section-after" keywords and section-reordering statements. The syntax is

"section-symbol %s\n", <section-symbol>

The <section-symbol> is a symbolic name, enclosed between angle brackets (< and >), and does not duplicate any symbolic name in the current charmap (if any), or any other symbolic name defined in this collation definition. A <section-symbol> defined via this keyword is only defined within the LC_COLLATE category.

Example:
section-symbol <LATIN>
section-symbol <ARABIC>

4.4.5 "collating-element" keyword

In addition to the collating elements in the character set, the collating-element keyword is used to define multicharacter collating elements. The syntax is

"collating-element %s from %s\n",<collating-symbol>,<string>

The <collating-symbol> operand is a symbolic name, enclosed between angle brackets (< and >), and does not duplicate any symbolic name in the current charmap or repertoiremap file (if any), or any other symbolic name defined in this collation definition. The string operand is a string of two or more characters that collates as an entity. A <collating-element> defined via this keyword is only defined within the LC_COLLATE category.
Example with ISO/IEC 10646-1:
collating-element <ch> from "<c><h>"
collating-element <e-acute> from "<e><combining-acute>"
collating-element <aa> from "<a><a>

Note: The problem of comparing a fully composed character of ISO/IEC 10646 with a decomposed representation of the same text is sometimes handled by the two strings comparing equal up to level 3 (the case level) of ISO/IEC 14651, but distinguishing the two at the 4th level.

4.4.6 "collating-symbol" keyword

This keyword is used to define symbols for use in collation sequence statements; e.g., between the order_start and the order_end keywords. The syntax is

"collating-symbol %s;%s;%s...%s\n", <collating-symbol>, <collating-symbol> ...

The <collating-symbol> is a symbolic name, enclosed between angle brackets (< and >), and does not duplicate any symbolic name in the current charmap (if any), or any other symbolic name defined in this collation definition. A <collating-symbol> defined via this keyword is only defined within the LC_COLLATE category. More than one <collating-symbol> may be defined with one "collating-symbol" keyword, and symbolic ellipses may be used.

Example:
collating-symbol <CAPITAL>
collating-symbol <HIGH>

4.4.7 "symbol-equivalence" keyword

This keyword is used to define symbols for use in collation sequence statements; and assign the same weight as another defined symbol. The syntax is

"symbol-equivalence %s %s\n", <collating-symbol-1>, <collating-symbol-2>

The <collating-symbol-1> and <collating-symbol-2> are symbolic names, enclosed between angle brackets (< and >). <collating-symbol-1> does not duplicate any symbolic name in the current charmap (if any), or any other symbolic name defined in this collation definition. <collating-symbol-2> is defined elsewhere in the LC_COLLATE category as a collating-symbol. The use of <collating-symbol-2> is equivalent to using the <collating-symbol-1> in the LC_COLLATE category. A <collating-symbol-1> defined via this keyword is only defined within the LC_COLLATE category.

Example
collating-symbol <CAP>
symbol-equivalence <CAPITAL> <CAP>

4.4.8 "order_start" keyword

The "order_start" keyword precedes collation order entries and also defines the number of weights for this collation sequence definition, the collation section name and other collation rules.

The syntax of the "order_start" keyword has two forms:
"order_start %s;%s;...;%s\n", <sort-rule>, <sort-rule> ...

and

"order_start %s;%s;...;%s\n", <section-symbol>, <sort-rules>, <sort-rules> ...

The operands to the order_start keyword are optional. If present, the operands define rules to be applied when strings are compared. The first operand may be a <section-symbol> surrounded by "<" and ">"> and the set of collating statements following the "order_start" keyword until the "order_end" keyword are identified with this <section-symbol> or another "order_start" keyword is encountered. The remaining number of operands define how many weights each element is assigned; if no operands are present, one forward operand is assumed. If present, the first operand defines rules to be applied when comparing strings using the first (primary) weight; the second when comparing strings using the second weight, and so on. Operands are separated by semicolons (;). Each operand consists of one or more collation directives, separated by commas (,). If the number or operands exceeds the (COLL_WEIGHTS_MAX) limit, a utility parsing the FDCC-set description issues a warning message. The following directives are supported:

- **forward**
  Specifies that the direction of scanning a part of a string at a given point in a string is done towards the logical end of the whole string for this weight level.

- **backward**
  Specifies that the direction of scanning a part of a string at a given point in a string is done towards the logical beginning of the whole string for this weight level.

- **position**
  Specifies that comparison operations for the weight level will consider the relative position of non-"IGNORE"d elements in the strings. The string containing a non-"IGNORE"d element after the fewest IGNOREd collating elements from the start of the compare collates first. If both strings contain a non-"IGNORE"d character in the same relative position, the collating values assigned to the elements determine the ordering. In case of equality, subsequent non-IGNOREd characters are considered in the same manner.

The directives "forward" and "backward" are mutually exclusive at a given level. The directives "backward" and "position" are mutually exclusive at a given level.

Examples:
order_start forward;backward
order_start <CYRILLIC>;forward;forward

If no operands are specified, a single forward operand is assumed.

4.4.9 "order_end" keyword

The collating order entries are terminated with an "order_end" keyword.

4.4.10 "reorder-after" keyword

The "reorder-after" keyword is used to specify a modification to a copied collation specification of an existing FDCC-set. There can be more than one "reorder-after" statement in a collating specification. The syntax is:

"reorder-after %s\n",<collating-symbol>

The <collating-symbol> operand is a symbolic name, enclosed between angle brackets, and is present in the source FDCC-set copied via the "copy" keyword.
The "reorder-after" statement is followed by one or more collation statements as described in the "Collating Order" clause (4.4.5), with the exception that the ellipsis symbol (...) is not used.

Each collation statement reassigns character collation values and collation weights to collating elements existing in the copied collation specification, by removing the collating statement from the copied specification, and inserting the collating element in the collating sequence with the new collation weights after the preceding collating element of the "reorder-after" specification, the first collating element in the collation sequence being the <collating-symbol> specified in the "reorder-after" statement.

A "reorder-after" specification is terminated by another "reorder-after" specification or the "reorder-end" statement.

4.4.10.1 Example of "reorder-after"

reorder-after <y8>
 <U:> <Y>;<U:>;<CAPITAL>
 <u:> <Y>;<U:>;<SMALL>
reorder-after <z8>
 <AE> <AE>;<NONE>;<CAPITAL>
 <ae> <AE>;<NONE>;<SMALL>
 <A:> <AE>;<DIAERESIS>;<CAPITAL>
 <a:> <AE>;<DIAERESIS>;<SMALL>
 <O/> <O/>;<NONE>;<CAPITAL>
 <o/> <O/>;<NONE>;<SMALL>
 <AA> <AA>;<NONE>;<CAPITAL>
 <aa> <AA>;<NONE>;<SMALL>
reorder-end

The example is interpreted as follows (using the "i18nrep" repertoiremap):

1. The collating element <U:> is removed from the copied collating sequence and inserted after <y8> in the collating sequence with the new weights. The collating element <u:> is removed from the copied collating sequence and inserted in the resulting collation sequence after <U:> with the new weights. <y8> is used to indicate the position of the last y letter.

2. The second "reorder-after" statement terminates the first list of reordering collation identifier entries, and initiates a second list, rearranging the order and weights for the <AE>, <ae>, <A>, <a>, <O>, and <o> collating elements after the <z8> collating symbol in the copied specification. <z8> is used to indicate the position of the last z letter.

3. The "reorder-end" statement terminates the second list of reordering entries.

4. Thus for the original sequence

... (U u Ü ü ) V v W w X x Y y Z z

this example reordering gives

... U u V v W w X x (Y y Ü ü ) Z z (Æ æ Ä ä ) Ø ø Å å

where the parenthesis indicate ordering with the same weight on the first level for multiple upper/lowercase pairs.

4.4.11 "reorder-end" keyword

The "reorder-end" keyword specifies the end of a list of collating statements, initiated by the "reorder-after" keyword.
4.4.12 "section" keyword

The "section" keyword is used to define a section of the table. A section consists of a set of collation elements with their associated collation weights. A section can be moved as a whole via the "reorder-section-after" keyword.

Each "section" keyword has the syntax:

"section %s %s;...;%s\n", <section-symbol>, <collation-symbol>, ....

The <section-symbol> is a symbolic name, enclosed between angle brackets "<" and ">", and it defines the name of the section in question. It may have been defined in a "section-symbol" statement.

The <collation-symbol> is a symbolic name, enclosed between angle brackets "<" and ">", and it references a collating element previously specified, with associated weights. More than one <collating-symbol> may be referenced in one "section" statement, and symbolic ellipses may be used. The <collation-symbol>s identified via this list are removed from other parts of the collation specification. The list of <collation-symbol>s is optional.

A section consists of the collating elements identified on the "section" keyword line and with relative order and weights as specified earlier, plus the collation elements defined via the optionally following collating statements as described in 4.4.1. The section is terminated by another keyword line.

4.4.13 "reorder-section-after" keyword

The "reorder-section-after" keyword is used to specify a modification to a copied collation specification of an existing FDCC-set. The "reorder-section-after" statement is followed by one or more statements consisting of section reordering statements.

Each "reorder-section-after" keyword has either the syntax:

"reorder-section-after %s\n", <collation-symbol>

or:

"reorder-section-after %s %s", <section-symbol>, <collation-symbol>

The <collation-symbol> is a symbolic name, enclosed between angle brackets "<" and ">", and it references a collating element previously specified.

The <section-symbol> is a symbolic name, enclosed between angle brackets "<" and ">", and it refers to the name of the section in question, previously defined in a "section-symbol" or "section" keyword, and with contents allocated via a "order_start" or "section" keyword.

If there is no <section-symbol> given with the keyword, the keyword is followed by a number of section reordering statements, terminated by a "reorder-section-end" keyword.
The collating elements and associated weights of the section given with the keyword line, or the sections given on the following lines, are removed from the current sorting table, possibly reassigned sorting rules according to the section reordering statements, and inserted in the sorting table after the <collating-symbol>.

4.4.13.1 section reordering statements

The section reordering statements rearranges the set of collating entries and changes sorting rules for the set of collating entries identified by a section symbol in a preceding "order_start" statement. Each section reorder statement has the syntax:

"%s %s;...%s
", <section-symbol>, <sort-rule>, <sort-rule> ...

The <section-symbol> identifies the set of collating entries. The <section-symbol> is defined via a "section-symbol" or the "section" keyword, and values identified by the <section-symbol> is assigned via the "order_start" or "section" keywords.

The <sort-rule>s are as described for the "order_start" keyword. Specified <sort-rule>s replace the specification of the ordering given on the first "order_start" statement, for the section identified by the <section-symbol>. The <sort-rule>s are optional, and <sort-rule>s not to be changed from the first "order_start" specification is given by empty specifications on the "section" statement.

Note: The <sort-rule> capability is an extension over ISO/IEC 14651 functionality.

The order of the section reordering statements rearranges the assignment of collation entries for the sets of collation entries identified by the <section-symbols> to the order that the <section-symbols> occur after the "reorder-section-after" statement.

The section reordering statements are terminated by a "reorder-section-end" statement.

4.4.13.2 Example of section reordering

copy "i18n"
section <DEVANAGARI> <U0905>..<U0939>;<U093D>..<U0950>
reorder-section-after <DEVANAGARI> <U3361>

This example is interpreted as follows: The LC_COLLATE category of the "i18n" FDCC-set is copied. Then a definition of the section <DEVANAGARI> is done, and the collating elements of this section is removed from the table and inserted in the same relative order and with the same weights after the collating element <U3361>, which is the last of the digits. In this way the <DEVANAGARI> section is reordered to be sorted before all other letters.

4.4.14 "reorder-section-end" keyword

The "reorder-section-end" keyword specifies the end of a list of section symbols, initiated by the "reorder-section-after" keyword.

4.4.15 "i18n" LC_COLLATE category

The "i18n" LC_COLLATE category is defined as the following, which includes the tailorable template in ISO/IEC 14651.

LC_COLLATE
% This is the ISO/IEC TR 14652 i18n fdcc-set definition for
% the LC_COLLATE category.
%
% equivalences
symbol-equivalence <NONE>              <BASE>
symbol-equivalence <CAPITAL>           <CAP>
symbol-equivalence <SMALL>             <MIN>
symbol-equivalence <CAPITAL-SMALL>     <COMPATCAP>
symbol-equivalence <SMALL-CAPITAL>     <COMPAT>
symbol-equivalence <MACRON>            <MACRO>
symbol-equivalence <STROKE>            <OBLIK>
symbol-equivalence <ACUTE>             <AIGUT>
symbol-equivalence <CIRCUMFLEX>        <CIRCF>
symbol-equivalence <RING>              <CRCLE>
symbol-equivalence <DIAERESIS>         <TREMA>
symbol-equivalence <DOT>               <POINT>
symbol-equivalence <CEDILLA>           <CEDIL>
symbol-equivalence <OGONEK>            <OGONK>
symbol-equivalence <HOOK>              <CROOK>
symbol-equivalence <HORN>               <HORNU>
symbol-equivalence <DOT-BELOW>         <POINS>

% Copy the template from ISO/IEC 14651
copy "ISO14651_2000_TABLE1.txt"
reorder-after <SFFFF>
order_start forward;forward;forward;forward,position
reorder-end

END LC_COLLATE

4.5 LC_MONETARY (controversial)

The LC_MONETARY category defines the rules and symbols that are used to format monetary numeric information. The operands are strings. For some keywords, the strings can contain only integers. More than one set of monetary values may be provided, and for each set a period of validity and conversion rate may be given. Keywords that are not provided, string values set to the empty string ""

\footnotesize

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy</td>
<td>Specify the name of an existing FDCC-set to be used as the source for the definition of this category. If this keyword is specified, no other keyword is specified.</td>
</tr>
<tr>
<td>valid_from</td>
<td>One or more strings separated by semicolons, representing a Gregorian date in the form &quot;YYYYMMDD&quot; 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.</td>
</tr>
<tr>
<td>valid_to</td>
<td>One or more strings, separated by semicolons, each representing a Gregorian date in the form &quot;YYYYMMDD&quot; according to ISO 8601, that specify the last 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.</td>
</tr>
</tbody>
</table>
conversion_rate

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 is separated by a <slash>. The default value is "1/100". This keyword is optional.

Note: The two integers are used instead of a floating point value, to be able to cater for legal requirements on Euro conversion where a multiplication and division is prescribed, instead of just one floating point multiplication.

currency_symbol

One or more strings separated by semicolons that are used as the local currency symbol.

mon_decimal_point

The operand is a string containing the symbol that is used as the decimal delimiter in monetary formatted quantities. In contexts where other standards limit the "mon_decimal_point" to a single byte, the result of specifying a multibyte operand is unspecified. The keyword is specified, unless the "copy" keyword is used.

mon_thousands_sep

The operand is a string containing the symbol that is used as a separator for groups of digits to the left of the decimal delimiter in formatted monetary quantities. In contexts where other standards limit the "mon_thousands_sep" to a single byte, the result of specifying a multibyte operand is unspecified. The keyword is specified, unless the "copy" keyword is used.

mon_grouping

Define the size of each group of digits in formatted monetary quantities. The operand is a sequence of integers separated by semicolons. Each integer specifies the number of digits in each group, with the initial integer defining the size of the group immediately preceding the decimal delimiter, and the following integers defining the preceding groups. If the last integer is not -1, then the size of the previous group (if any) is repeatedly used for the remainder of the digits. If the last integer is -1, then no further grouping is performed. The keyword is specified, unless the "copy" keyword is used.

positive_sign

A string that is used to indicate a nonnegative-valued formatted monetary quantity. The keyword is specified, unless the "copy" keyword is used.

negative_sign

A string that is used to indicate a negative-valued formatted monetary quantity. The keyword is specified, unless the "copy" keyword is used.

frac_digits

One or more integers separated by semicolons, representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using "currency_symbol". The keyword is specified,
unless the "copy" keyword is used.

**p_cs_precedes**

One or more integers separated by semicolons, set to 1 if the "currency_symbol" precedes the value for a nonnegative formatted monetary quantity, and set to 0 if the symbol succeeds the value. The keyword is specified, unless the "copy" keyword is used.

**p_sep_by_space**

One or more integers separated by semicolons, set to 0 if no space separates the "currency_symbol" from the value for a nonnegative formatted monetary quantity, set to 1 if a space separates the symbol from the value, and set to 2 if a space separates the symbol and the sign string, if adjacent. The keyword is specified, unless the "copy" keyword is used.

**n_cs_precedes**

One or more integers separated by semicolons, set to 1 if the "currency_symbol" precedes the value for a negative formatted monetary quantity, and set to 0 if the symbol succeeds the value. The keyword is specified, unless the "copy" keyword is used.

**n_sep_by_space**

One or more integers separated by semicolons, set to 0 if no space separates the "currency_symbol" from the value for a negative formatted monetary quantity, set to 1 if a space separates the symbol from the value, and set to 2 if a space separates the symbol and the sign string, if adjacent. The keyword is specified, unless the "copy" keyword is used.

**p_sign_posn**

One or more integers separated by semicolons, set to a value indicating the positioning of the "positive_sign" for a nonnegative formatted monetary quantity using the "currency_symbol". The following integer values are defined:

- 0 Parentheses enclose the quantity and the "currency_symbol".
- 1 The sign string precedes the quantity and the "currency_symbol".
- 2 The sign string succeeds the quantity and the "currency_symbol".
- 3 The sign string immediately precedes the "currency_symbol".
- 4 The sign string immediately succeeds the "currency_symbol".

The keyword is specified, unless the "copy" keyword is used.

**n_sign_posn**

One or more integers separated by semicolons, set to a value indicating the positioning of the "negative_sign" for a negative formatted monetary quantity using the "currency_symbol". The following integer values are defined:

- 0 Parentheses enclose the quantity and the "currency_symbol".
- 1 The sign string precedes the quantity and the "currency_symbol".
- 2 The sign string succeeds the quantity and the "currency_symbol".
- 3 The sign string immediately precedes the "currency_symbol".
4. The sign string immediately succeeds the "currency Symbol".

The keyword is specified, unless the "copy" keyword is used.

**int_curr_symbol**

One or more strings separated by semicolons that are used as the international currency symbols. Each operand is a four character string, with the first three characters containing the alphabetic international currency symbol in accordance with those specified in ISO 4217, *Codes for the representation of currencies and funds*. The fourth character is the character used to separate the international currency symbol from the monetary quantity. The keyword is specified, unless the "copy" keyword is used.

**int_frac_digits**

One or more integers separated by semicolons, representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using "int_curr_symbol". The keyword is specified, unless the "copy" keyword is used.

**int_p_cs_precedes**

One or more integers separated by semicolons; set to 1 if the "int_curr_symbol" precedes the value for a nonnegative formatted monetary quantity, and set to 0 if the symbol succeeds the value. If not specified, the value of "p_cs_precedes" is taken.

**int_p_sep_by_space**

One or more integers separated by semicolons; set to 0 if no space separates the "int_curr_symbol" from the value for a nonnegative formatted monetary quantity, set to 1 if a space separates the symbol from the value, and set to 2 if a space separates the symbol and the sign string, if adjacent. If not specified, the value of "p_sep_by_space" is taken.

**int_n_cs_precedes**

One or more integers separated by semicolons; set to 1 if the "int_curr_symbol" precedes the value for a negative formatted monetary quantity, and set to 0 if the symbol succeeds the value. If not specified, the value of "n_cs_precedes" is taken.

**int_n_sep_by_space**

One or more integers separated by semicolons; set to 0 if no space separates the "int_curr_symbol" from the value for a negative formatted monetary quantity, set to 1 if a space separates the symbol from the value, and set to 2 if a space separates the symbol and the sign string, if adjacent. If not specified, the value of "n_sep_by_space" is taken.

**int_p_sign_posn**

One or more integers separated by semicolons, set to a value indicating the positioning of the "positive_sign" for a nonnegative formatted monetary quantity using the "int_curr_symbol". The following integer values are defined:

0 Parentheses enclose the quantity and the "int_curr_symbol".
1 The sign string precedes the quantity and the "int_curr_symbol".
2 The sign string succeeds the quantity and the "int_curr_symbol".
3 The sign string immediately precedes the "int_curr_symbol".
The sign string immediately succeeds the "int_curr_symbol".
If no "int_p_sign_posn" is present the value of the "p_sign_posn" is taken.

**int_n_sign_posn**

One or more integers separated by semicolons, set to a value indicating the positioning of the "negative_sign" for a negative formatted monetary quantity using the "int_curr_symbol". The following integer values are defined:

0 Parentheses enclose the quantity and the "int_curr_symbol".
1 The sign string precedes the quantity and the "int_curr_symbol".
2 The sign string succeeds the quantity and the "int_curr_symbol".
3 The sign string immediately precedes the "int_curr_symbol".
4 The sign string immediately succeeds the "int_curr_symbol".
If no "int_n_sign_posn" is present the value of the "n_sign_posn" is taken.

The "i18n" FDCC-set is defined as follows for the LC_MONETARY category.

```plaintext
LC_MONETARY
% This is the 14652 i18n fdcc-set definition for
% the LC_MONETARY category.
%
  int_curr_symbol  ""
  currency_symbol  ""
  mon_decimal_point  ",U002C"
  mon_thousands_sep  ""
  mon_grouping  -1
  positive_sign  ""
  negative_sign  ",U002E"
  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
```

### 4.6 LC_NUMERIC

The LC_NUMERIC category defines the rules and symbols that are used to format nonmonetary numeric information. The operands are strings. For some keywords, the strings only can contain integers. 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. The following keywords are defined:

**copy**

Specify the name of an existing FDCC-set to be used as the source for the definition of this category. If this keyword is specified, no other keyword is specified.
decimal_point  The operand is a string containing the symbol that is used as the
decimal delimiter in numeric, nonmonetary formatted quantities.
This keyword cannot be omitted and cannot be set to the empty
string. In contexts where other standards limit the decimal point
to a single byte, the result of specifying a multibyte operand is
unspecified.

thousands_sep  The operand is a string containing the symbol that is used as a
separator for groups of digits to the left of the decimal delimiter
in numeric, nonmonetary formatted monetary quantities. In
contexts where other standards limit the "thousands_sep" to a
single byte, the result of specifying a multibyte operand is
unspecified.

grouping  Define the size of each group of digits in formatted non-
monetary quantities. The operand is a sequence of integers
separated by semicolons. Each integer specifies the number of
digits in each group, with the initial integer defining the size of
the group immediately preceding the decimal delimiter, and the
following integers defining the preceding groups. If the last
integer is not -1, then the size of the previous group (if any) is
repeatedly used for the remainder of the digits. If the last integer
is -1, then no further grouping is performed.

The "i18n" FDCC-set is for the LC_NUMERIC category:

```
LC_NUMERIC
% This is the 14652 i18n fdcc-set definition for
% the LC_NUMERIC category.
%
decimal_point  "$\langle U002C\rangle$"
thousands_sep    ""
grouping        -1
%
END LC_NUMERIC
```

4.7  LC_TIME (controversial)

The LC_TIME category defines the rules and symbols that are used to format date and
time information.

Note: ISO 8601 allows different formats for dates, one form is YYYY-MM-DD, another
is YYYYMMDD. Each clause in this specification specifies which specific format of ISO
8601 that is used there.

The following keywords are defined:

copy  Specify the name of an existing FDCC-set to be used as the source
for the definition of this category. If this keyword is specified, no
other keyword is specified.

abday  Define the abbreviated weekday names for calendar systems with
weeks of constant length, to be referenced by the %a field descriptor.
The length of the week and a Gregorian date for the first weekday is
defined by the "week" keyword. The operand consists of semicolon-
separated strings. The first string is the abbreviated name of the day
corresponding to the first day of the week (default Sunday), the
second the abbreviated name of the day corresponding to the second
day of the week (default Monday), and so on.
Define the full weekday names for calendar systems with weeks of constant length, to be referenced by the %A field descriptor. The length of the week and a Gregorian date for the first weekday is defined by the "week" keyword. The operand consists of semicolon-separated strings. The first string is the full name of the day corresponding to the first day of the week (default Sunday), the second the full name of the day corresponding to the second day of the week (default Monday), and so on.

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.

Define the abbreviated month names, to be referenced by the %b field descriptor. The operand consists of twelve or thirteen semicolon-separated strings. The first string is the abbreviated name of the first month of the year (January), the second the abbreviated name of the second month, and so on.

Define the full month names, to be referenced by the %B field descriptor. The operand consists of twelve or thirteen semicolon-separated strings. The first string is the full name of the first month of the year (January), the second the full name of the second month, and so on.

Define the appropriate date and time representation, to be referenced by the %c field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain field descriptors defined in Table 3.

Define the appropriate date representation, to be referenced by the %x field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain field descriptors defined in Table 3.

Define the appropriate time representation, to be referenced by the %X field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain field descriptors defined in Table 3.

Define the appropriate representation of the ante meridiem and post meridiem strings, to be referenced by the %p field descriptor. The operand consists of two strings, separated by a semicolon. The first string represents the antemeridiem designation, the last string the postmeridiem designation. The keyword is optional. If unspecified, the %p field descriptor refers to the empty string.
Define the appropriate time representation in the 12-hour clock format with "am_pm", to be referenced by the %r field descriptor. The operand consists of a string and can contain any combination of characters and field descriptors. If the string is empty, the 12-hour format is not supported in the FDCC-set.

The following keywords are all optional:

**era**
Define how years are counted and displayed for each era in a locale. The operand shall consist of semicolon-separated strings. Each string shall be an era description segment with the format:

 direction:offset:start_date:end_date:era_name:era_format

to describe the different eras. There can be as many era description segments as are necessary to describe the different eras.

**NOTE:** The start of an era might not be the earliest point in the era - it may be the AD 1, and increases with earlier time.

**direction**
Either a ‘+’ or a ‘-’ character. The ‘+’ character shall indicate that years closer to the start_date have lower numbers than those closer to the end_date. The ‘-’ character shall indicate that years closer to the start_date have higher numbers than those closer to the end_date.

**offset**
The number of the year closest to the start_date in the era, corresponding to the %Ey conversion specification.

**start_date**
A date in the format YYYYMMDD, where YYYY, MM, and DD are the year, month, and day numbers respectively according to ISO 8601 of the start of the era. Years prior to AD 1 shall be represented as negative numbers.

**end_date**
The ending date of the era, in the same format as the start_date, or one of the two special values "-*" or "+*". The value "-*" shall indicate that the ending date is the beginning of time. The value "+*" shall indicate that the ending date is the end of time.

**era_name**
A string representing the name of the era, corresponding to the %EC conversion specification.

**era_format**
A string for formatting the year in the era, corresponding to the %EY conversion specification.

**era_year**
Define the format of the year in alternate Era format, corresponding to the %EY field descriptor.

**era_d_t_fmt**
Define the format of the date and time in alternate Era notation, corresponding to the %Ec field descriptor.

**era_d_fmt**
Define the format of the date in alternate Era notation, corresponding to the %Ex field descriptor.

**era_t_fmt**
Define the format of the time in alternate Era notation, corresponding to the %EX field descriptor.

**alt_digits**
Define alternate symbols for digits, corresponding to the %O field descriptor modifier. The operand consists of semicolon-separated strings. The first string is the alternate symbol corresponding with zero, the second string the symbol corresponding with one, and so on. Up to 100 alternate symbol strings can be specified. The %O modifier indicates that the string corresponding to the value specified via the field descriptor is used instead of the value.
**first_weekday** Define the first day to be displayed, for example in a calendar display utility. The operand is an integer specifying the day number (1 = first) according to the information specified with the "day" keyword. The keyword may be omitted, and then the value 1 is taken, corresponding to Sunday for a week beginning Sunday, or to Monday for a week beginning Monday.

**first_workday** Define the first workday as an integer according to the day numbering specified with the "week" keyword.

**cal_direction** Define the direction of the display of dates, for example in a calendar display utility. The operand is an integer, and the following values are defined:

1. left-right from top
2. top-down from left
3. right-left from top

The keyword may be omitted, and then the value 1 is taken.

**timezone** Define one or more timezones, each defined by a string, and the strings separated by a <semicolon>. In the following the characters <, >, [ and ] are used as metacharacters. Only characters with a visible glyph from the portable character set may be used, except in the <std> and <dst> fields. The syntax of a string is:

```
<std><offset><dst>[[<offset>] [,<rule> [,<rule> ...]]];
```

where

- `<std>` and `<dst>` Indicates no less than three, nor more than 10 characters that are the designation for the standard `<std>`, or Daylight Savings Time or summer time `<dst>` zone. Only `<std>` is required; if `<dst>` is missing, then Daylight Savings Time or summer time does not apply in this category. Upper- and lowercase letters are explicitly allowed. Any characters except a leading colon `<:` or digits, the comma `<,>`, the minus `<->`, the plus `<+>`, and the null character are permitted to appear in these fields, but their meaning is unspecified.

- `<offset>` Indicates the value one must add to the local time to arrive at the Coordinated Universal Time. The `<offset>` has the form:

```
hh[mm[ss]]
```

The minutes (mm) and seconds (ss) are optional. The hour (hh) is required and may be a single digit. The `<offset>` following `<std>` is required. If no `<offset>` follows `<dst>`, summer time is assumed to be one hour ahead of standard time. One or more digits may be used; the value is always interpreted as a decimal number. The hour is between zero and 24, and
the minutes (and seconds) - if present - is between zero and 59. If preceded by a "-", the time zone is east of the Prime Meridian; otherwise it is west of (which may be indicated by an optional preceding "+").

<rule> A specification for Daylight Savings Time changes that indicates when to change to and back from summer time. The <rule> has the form:

<date>[/<time>/<year>],[<date>[/<time> /<year>]

where the first <date> describes when the change from standard time to summer time occurs, and the second <date> describes when the change back happens. Each <time> field describes when, in current local time, the change to the other time is made. The first <year> field defines the beginning of the validity of this rule, and the second <year> field defines the end of the validity of the rule. A number of rules may be given.

The format of <date> is one of the following:

- J<n> The Julian day <n> (1 <= n <= 365) Leap years are not counted. That is, in all years - including leap years - February 28 is day 59 and March 1 is day 60. It is impossible to explicitly refer to the occasional February 29.
- <n> The zero-based Julian day (0 <= n <= 365). Leap years are counted and it is possible to refer to February 29.
- M<m>.<n>.<d> the <d>th day (0 <= d <= 7) of week <n> of month <m> (1 <= n <= 5, 1 <= m <= 12, where week 5 means "the last <d> day in month <m>" which may occur in either the fourth or fifth week). Week 1 is the first week in which the <d>th day occurs. Day zero and day seven is Sunday.

The <time> has the same format as <offset> except that no leading sign ("-" or "+") is allowed. The default, if <time> is not given, is "02:00:00".
The <year> has the format YYYY.

NOTE: This way of specifying the timezone is compatible with the format for the environment variable TZ described in Section 8.1.1 of POSIX.1.

4.7.1 Date Field Descriptors

The LC_TIME category defines the interpretation of a number of field descriptors. The field descriptors are also available in the definitions with the following LC_TIME keywords: "d_t_fmt", "d_fmt", "t_fmt", "t_fmt_ampm", "era", "era_d_t_fmt", "era_d_fmt", and "era_t_fmt". A field descriptor may not be used with the LC_TIME keywords defining it.

Table 3: Field descriptors for the date field

| %a | FDCC-set’s abbreviated weekday name. |
| %A | FDCC-set’s full weekday name.       |
| %b | FDCC-set’s abbreviated month name.  |
| %B | FDCC-set’s full month name.         |
| %c | FDCC-set’s appropriate date and time representation. |
| %C | Century (a year divided by 100 and truncated to integer) as decimal number (00-99). |
| %d | Day of the month as a decimal number (01-31). |
| %D | Date in the format mm/dd/yy.         |
| %e | Day of the month as a decimal number (1-31 in at two-digit field with leading <space> fill). |
| %F | The date in the format YYYY-MM-DD (An ISO 8601 format). |
| %g | Week-based year within century, as a decimal number (00-99). |
| %G | Week-based year with century, as a decimal number (for example 1997). |
| %h | A synonym for %b.                   |
| %H | Hour (24-hour clock), as a decimal number (00-23). |
| %I | Hour (12-hour clock), as a decimal number (01-12). |
| %j | Day of the year, as a decimal number (001-366). |
| %m | Month, as a decimal number (01-13).   |
| %M | Minute, as a decimal number (00-59).  |
| %n | A <newline> character.              |
| %p | FDCC-set’s equivalent of either AM or PM. |
| %r | 12-hour clock time (01-12), using the AM/PM notation. |
| %R | 24-hour clock time, in the format "%H:%M". |
| %S | Seconds, as a decimal number (00-61). |
| %t | A <tab> character.                  |
| %T | 24-hour clock time, in the format HH:MM:SS. |
| %u | Weekday, as a decimal number (1(Monday)-7). |
| %U | Week number of the year (Sunday as the first day of the week) as a decimal number (00-53). All days in a new year preceding the first Sunday are considered to be in week 0. |
| %v | Week number of the year, as a decimal number with two digits including a possible leading zero, according to "week" keyword. |
| %V | Week of the year (Monday as the first day of the week), as a decimal number (01-53). The method for determining the week number is as specified by ISO 8601. |
| %w | Weekday, as a decimal number (0(Sunday)-6). |
| %W | Week number of the year (Monday as the first day of the week), as a
decimal number (00-53). All days in a new year preceding the first
Monday are considered to be in week 0.
%<x> FDCC-set’s appropriate date representation.
%<X> FDCC-set’s appropriate time representation.
%<y> Year within century (00-99).
%<Y> Year with century, as a decimal number.
%<z> The offset from UTC in the ISO 8601 format "-0430" (meaning 4 hours 30
minutes behind UTC, west of Greenwich), or by no characters if no time
zone is determinable.
%<Z> Time-zone name, or no characters if no time zone is determinable.
%% A <percent-sign> character.

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 first Monday of the year is the 2nd, 3rd or 4th, the preceding days are part of the last
week of the preceding year; thus, for Saturday 2nd January 1999, %G is replaced by 1998 and %V is
replaced by 53. If the 29th, 30th or 31st December 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.

4.7.2 Modified Field Descriptors

Some field descriptors can be modified by the E and O modifier characters to indicate a
different format or specification as specified in the LC_TIME FDCC-set description. If the
 corresponding keyword (see "era", "era_year", "era_d_t_fmt", "era_d_fmt", "era_t_fmt" and
"alt_digits") is not specified for the current FDCC-set, the unmodified field descriptor
value is used.

%<Ec> FDCC-set’s alternate date and time representation.
%<EC> The name of the base year (period) in the FDCC-set’s alternate
representation.
%<Ex> FDCC-set’s alternate date representation.
%<EX> FDCC-set’s alternate time representation.
%<Ey> Offset from %<EC> (year only) in the FDCC-set’s alternate representation.
%<EY> Full alternate year representation.
%<Od> Day of month using the FDCC-set’s alternate numeric symbols.
%<Oe> Day of month using the FDCC-set’s alternate numeric symbols.
%<Of> Weekday as a decimal number according to alt_day (1 is first day).
%<OH> Hour (24-hour clock) using the FDCC-set’s alternate numeric symbols.
%<OI> Hour (12-hour clock) using the FDCC-set’s alternate numeric symbols.
%<OM> Month using the FDCC-set’s alternate numeric symbols.
%<OS> Seconds using the FDCC-set’s alternate numeric symbols.
%<Ou> Weekday as a number in the alternate representation of the FDCC-set
(Monday=1).
%<OU> Week number of the year (Sunday as the first day of the week) using the
FDCC-set’s alternate numeric symbols.
%<OV> Week number of the year (Monday as the first day of the
week, ISO 8601 rules) using the alternate numeric symbols of
the FDCC-set.
%<Ow> Weekday as number in the FDCC-set’s alternate representation
(Sunday=0).
%OW Week number of the year (Monday as the first day of the week) using the
FDCC-set’s alternate numeric symbols.
%Oy Year (offset from %C) in alternate representation.

4.7.3 "i18n" LC_TIME category

The "i18n" LC_TIME category is (following ISO 8601):

```
LC_TIME
% This is the ISO/IEC TR 14652 "i18n" definition for
% the LC_TIME category.
%
% Weekday and week numbering according to ISO 8601
abday  "<U0031>";"<U0032>";"<U0033>";"<U0034>";/
       "<U0035>";"<U0036>";"<U0037>"
day    "<U0031>";"<U0032>";"<U0033>";"<U0034>";/
       "<U0035>";"<U0036>";"<U0037>"
week   7;19971201;4
abmon  "<U0030><U0031>";"<U0030><U0032>";"<U0030><U0033>";/
       "<U0030><U0034>";"<U0030><U0035>";"<U0030><U0036>";/
       "<U0030><U0037>";"<U0030><U0038>";"<U0030><U0039>";/
       "<U0031><U0030>";"<U0031><U0031>";"<U0031><U0032>
mon    "<U0030><U0031>";"<U0030><U0032>";"<U0030><U0033>";/
       "<U0030><U0034>";"<U0030><U0035>";"<U0030><U0036>";/
       "<U0030><U0037>";"<U0030><U0038>";"<U0030><U0039>";/
       "<U0031><U0030>";"<U0031><U0031>";"<U0031><U0032>
am_pm  "";""% date formats following ISO 8601
% Appropriate date and time representation (%c)
%  "%F %T"
d_t_fmt  "<U0025><U0046><U0020><U0025><U0025><U0025><U0025>
% Appropriate date representation (%x)  "%F"
d_fmt   "<U0025><U0046>"%
% Appropriate time representation (%X)  "%T"
t_fmt   "<U0025><U0054>"
t_fmt_ampm  ""
%
END LC_TIME
```

4.8 LC_MESSAGES

The LC_MESSAGES category defines the format and values for affirmative and negative
responses. The operands are strings or extended regular expressions to specify which
response strings that should be considered matches; see ISO/IEC 9945-2:1993 clause 2.8.4
for a definition of extended regular expressions. The following keywords are defined:

**copy** Specify the name of an existing FDCC-set to be used as the source for the
definition of this category. If this keyword is specified, no other keyword
is specified.

**yesexpr** The operand consists of an extended regular expression that describes the
acceptable affirmative response to a question expecting an affirmative or
negative response.

**noexpr** The operand consists of an extended regular expression that describes the
acceptable negative response to a question expecting an affirmative or
negative response.

The "i18n" LC_MESSAGES category is:

```
LC_MESSAGES
% This is the ISO/IEC 14652 "i18n" definition for
% the LC_MESSAGES category.
```
Note: This uses regular expression syntax with brackets ([]) to for example specify that both <+> and <1> is allowed as an affirmative answer.

4.9 LC_XLITERATE (controversial)

The LC_XLITERATE category defines formats to transform strings, by transforming substrings in the source to substrings in the target string. The target is the culture of the FDCC-set in question. The capabilities can be used for simple transliteration or fallback based on substring substitution, while more advanced transliteration schemes, for example based on pattern matching, sound equivalences, or using a database, is either cumbersome to specify, or not addressed. The transliteration may for example be from the Cyrillic script to the Latin script.

Transliteration of an incoming character string to a character string in a FDCC-set can be specified with the following transliteration keywords and transliteration statements.

- **copy** Specify the name of an existing FDCC-set to be used as the source for the definition of this category. If this keyword is specified, no other keyword is specified.
- **include** The name of the FDCC-set in text form to transliterate from, and the repertoiremap for the FDCC-set to be used for the definition of the transliteration statements. Other transliteration statements may follow to replace specification of the copied FDCC-set. This keyword is optional.
- **default_missing** defines a string of one or more characters to be put in the output string if no transliteration statement can be applied to a input <transliteration-source>. This keyword is optional.
- **translit_ignore** defines a set of characters, separated by semicolons, that are to be ignored in the incoming character string, that is, each of the occurrences of such characters is treated as the empty string. The characters may use the notations defined in 4.3 for lists of characters. This keyword is optional.
- **redefine** This keyword introduces a list of transliteration statements where each of the <transliteration_source> strings have been defined previously in the specification, and the new transliteration statements then replaces the old transliteration statements for the <transliteration_source> strings specified. This keyword is optional.

### 4.9.1 Transliteration statements

The syntax for a transliteration statement is:

```
"%s %s;%s;...;%s
",<transliteration_source>,<transliteration_string>,...
```

Each <transliteration_source> consists of one or more characters (in any of the forms defined in 4.1.1). The <transliteration_source> that is the longest in terms of number of characters that match the input string is the one selected for transliteration.
If a transliteration statement contains more than one `<transliteration_string>`, the order that each `<transliteration_string>` occurs in the transliteration statement defines the precedence order for choosing a particular `<transliteration_string>` to substitute for the `<transliteration_source>`. When a process makes use of a transliteration statement to transliterate text, and that transliteration statement contains more than one `<transliteration_string>`, that process chooses the first `<transliteration_string>`, in the defined precedence order, that satisfies the requirements of the transliteration.

Note: the exact definition of the concept of satisfying the requirements of the transliteration is outside the context of this Technical Report. If, for example, a transliteration involves a change in the coded character set of a string, a `<transliteration_string>` must be chosen, all of whose elements are members of that coded character set. In order to determine this, it would be expected that a repertoire describing which characters are to be present in the resulting transformed string be available to the transliteration API. Also, a transliteration may involve requirements such as that string length not change under transliteration. Such requirements may also affect the choice among alternative `<transliteration_string>` values.

If more than one transliteration statement is given for a given `<transliteration_source>` this is an error, and duplicate transliteration statements are ignored. Tailoring of transliteration statements may be done via the "redefine" keyword.

4.9.2 "include" keyword

The "include" keyword specifies a set of transliteration statements in text form to be included in the applied transliteration. The syntax of the "include" statement is:

"include %s;%s\n", <FDCC-set>, <repertoiremap>

<FDCC-set> is a string identifying the FDCC-set to be included from.

<repertoiremap> is a string identifying the repertoiremap used in the FDCC-set being included, and is used to map character specifications from the specified FDCC-set into the current FDCC-set.

4.9.3 Example of use of transliteration

LC_XLITERATE
include "de_DE";"de_remap"
default_missing "?"
translit_ignore "U3200", "UFAFF"
<a> \a;\e*;"a\<e">;"e*" 
<\s> \s*;<\s>
"<K><O>" <KO>
END LC_XLITERATE

The "LC_XLITERATE" statement introduces the transliteration category.

The "include" keyword specifies that the FDCC-set "de_DE" is copied and that the repertoiremap "de_remap" is used to define the symbolic character names in the FDCC-set "de_DE".

The "default_missing" keyword introduces the character sequence "?" as the string to transform into for input characters that cannot be transformed into other strings, because no transliteration statement is applicable to the character.

The "translit_ignore" keyword specifies that a set of Ideographic characters, Hangul, East Asian symbols and the private use area etc. (the range "U3200", "UFAFF") is ignored for the transliteration.

The next 3 lines are transliteration statements.

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.

The transliteration category is terminated via the "END LC_XLITERATE" statement in the above example.

There is no "i18n" entry for the LC_XLITERATE category

4.10 LC_NAME

The LC_NAME category defines formats to be used in addressing a person, e.g., in a postal address or in a letter. The following keywords are defined:

**copy** Specify the name of an existing FDCC-set to be used as the source for the definition of this category. If this keyword is specified, no other keyword is specified.

**name_fmt** Define the appropriate representation of a person’s name and title. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain field descriptors defined below.

**name_gen** The operand is a string defining a salutation valid for all persons.

**name_miss** The operand is a string defining a salutation valid for unmarried females.

**name_mr** The operand is a string defining a salutation valid for males.

**name_mrs** The operand is a string defining a salutation valid for married females.

**name_ms** The operand is a string defining a salutation valid for all females.

NOTE: There are a number of variations for addressing a person among the cultures. Middle names are not used in many countries and even the family name is not used in some countries. In other countries there is extensive use of one or more middle names and corresponding initials. The specification below should be regarded as a starting point for this problem.

The LC_NAME category defines the interpretation of a number of field descriptors. The field descriptors are also available in the definitions with the following LC_NAME keywords: "name_fmt".

Field descriptors for the "name_fmt" keyword:

- `%f` Family names.
- `%F` Family names in uppercase.
- `%g` First given name.
- `%G` First given initial.
- `%l` First given name with Latin letters. In some cultures, eg on Taiwan it is customary to also have a first name written with Latin letters, although the rest of the name is written in another script.
- `%o` Other shorter name, eg. "Bill".
- `%m` Additional given names.
- `%M` Initials for additional given names.
- `%p` Profession.
- `%s` Salutation, such as "Doctor"
- `%S` Abbreviated salutation, such as "Mr." or "Dr."
- `%d` Salutation, using the FDCC-sets conventions, with 1 for the name_gen, 2 for name_mr, 3 for name_mrs, 4 for name_miss, 5 for name_ms.
- `%t` If the preceding field descriptor resulted in an empty string, then the empty string, else a <space>.

Each field descriptor may have an <R> after the <%= to specify that the information is taken from a Romanized version string of the entity. An initial is any string, normally
consisting of one letter and a punctuation mark; the Dutch "IJ" is an example of a two character initial.

The "i18n" LC_NAME category is:

```
LC_NAME
% 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>"
% This corresponds to "%p%t%g%m%t%f" which is
% Profession Primary Additionals Family
END LC_NAME
```

### 4.11 LC_ADDRESS

The LC_ADDRESS category defines formats to be used in specifying a location like a person’s home or office, for use in a postal address or in a letter, and other items related to geography, including natural language. All keywords are strings and may contain non-digits, and all keywords are optional. The following keywords are recognized:

**copy** Specify the name of an existing FDCC-set to be used as the source for the definition of this category. If this keyword is specified, no other keyword is specified.

**postal_fmt** 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 field descriptors defined below.

**country_name** The operand is a string with the name of the country in the language of the FDCC-set.

**country_post** The operand is a string with the abbreviation of the country, used for postal addresses, for example by the CEPT-MAILCODE codes designating countries in Europe. Other abbreviation systems are also allowed, and there is no specific way to identify which abbreviation system is being used.

**lang_name** The operand is a string with the name of the language in the language of the FDCC-set.

**lang_ab2** The operand is a string with the two-letter abbreviation of the language, according to ISO 639.

**lang_ab3_term** The operand is a string with the three-letter abbreviation of the language for terminology use, according to ISO 639-2.

**lang_ab3_lib** The operand is a string with the three-letter abbreviation of the language for library use, according to ISO 639-2. If not specified, the value of the "lang_ab3_term" keyword is taken.

Note: The "lang_ab3_term" and "lang_ab3_lib" keywords will in most cases contain the same value, but they may differ, e.g. the values for the German language is "deu" and "ger" respectively.

The LC_ADDRESS category defines the interpretation of a number of field descriptors. The field descriptors are also available in the definitions with the following LC_ADDRESS keywords: "postal_fmt".
Field descriptors for the "postal_fmt" keyword:

- %n Person’s name, possibly constructed with the LC_NAME "name_fmt" keyword.
- %a Care of person, or organization.
- %f Firm name.
- %d Department name.
- %b Building name.
- %s Street or block (eg. Japanese) name.
- %h House number or designation.
- %N Insert an <end-of-line> if the previous descriptor’s value was not an empty string; otherwise ignore.
- %t Insert a <space> if the previous descriptor’s value was not an empty string; otherwise ignore.
- %r Room number, door designation.
- %e Floor number.
- %C Country designation, from the <country_post> keyword.
- %l Local township within town or city
- %z Zip number, postal code.
- %T Town, city.
- %S State, province, or prefecture.
- %c Country, as taken from data record.

Each field descriptor may have an <R> after the <*> to specify that the information is taken from a Romanized version string of the entity.

NOTE: There are a number of variations for specifying a location among the cultures. Some of the information, like the middle names, or even the family name, is not used in some cultures. The specification here should be regarded as a starting point for this problem.

Examples:

A specification for the USA could be:

"%n%n%a%n%d%n%f%n%b%n%s%n%e %r%n%n%c-%z %T%, %S%n%e%N"

Giving:

Person’s name
C/o address
Department
Firm
Building
number street
floor room
Local Town
City, State Zip
Country

An example for South Korea could be:

"%S%T%l%n%h%n%f%n%b %e %r%n%n%a%n%z"

Giving:

State
City
Town
Street number
Firm department
Building
floor room
Person’s name
C/o address
Zip

The "i18n" LC_ADDRESS category is:
LC_ADDRESS
% This is the ISO/IEC TR 14652 "i18n" definition for
% the LC_ADDRESS category.
% postal_fmt    "<U0025><U006E><U0025><U004E>/
<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><U006C><U0025><U004E><U0025><U0066><U0025><U004E>/
<U0020><U0025><U0068><U0020><U0025><U0065><U0020><U0025><U0072>/
<U0025><U004E><U0025><U006C><U0025><U004E><U0025><U0066><U0025><U004E>/
<U0025><U0053><U0025><U004E><U0025><U0066><U0025><U004E><U0025><U004E>"}{n%N%a%N%r%N%W%N%C-%z %T%N%S%N%e%N} resulting in
% Person’s_Name
% C/o_person_or_org
% Firm
% Department
% Building_name
% Street_or_block number floor room
% Local_township
% Country-Zip City
% State_or_province
% Country
%
END LC_ADDRESS

4.12  LC_TELEPHONE

The LC_TELEPHONE category defines formats to be used with telephone services. All
keywords are optional. The strings are not restricted in what characters they can contain.
The following keywords are defined:

copy      Specify the name of an existing FDCC-set to be used as the source
         for the definition of this category. If this keyword is specified, no
         other keyword is specified.

tel_int_fmt Define the appropriate representation of a telephone number for
         international use. The operand consists of a string, and can contain
         any combination of characters and field descriptors. In addition, the
         string can contain field descriptors defined below.

tel_dom_fmt Define the appropriate representation of a telephone number for
         domestic use. The operand consists of a string, and can contain any
         combination of characters and field descriptors. In addition, the string
         can contain field descriptors defined below.

int_select The operand is a string with the digits used to call international
         telephone numbers.

int_prefix The operand is a string with the prefix used from other countries to
         call the area.

The LC_TELEPHONE category defines the interpretation of a number of field descriptors.
The field descriptors are also available in the definitions with the following
LC_TELEPHONE keywords: "tel_int_fmt" and "tel_dom_fmt".

%a area code without nationwide prefix (prefix is often <0>).
%A area code including nationwide prefix (prefix is often <0>).
%l local number (within area code).
%e extension (to local number)
%c country code
%C alternate carrier service code used for dialling abroad
%t Insert a <space> if the previous descriptor’s value was not an empty
      string; otherwise ignore.
The "i18n" LC_TELEPHONE category is:

```plaintext
LC_TELEPHONE
% This is the ISO/IEC TR 14652 "i18n" definition for
% the LC_TELEPHONE category.
%
tel_int_fmt       "<U002B><U0025><U0063><U0020><U0025><U0061><U0025><U0074>/
                     <U0025><U006C>"
% "+%c %a%t%l" which is
% +country area local
END LC_TELEPHONE
```

5. CHARMAP

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

A conforming charmap to be used with a FDCC-set supports the portable character set specified in Table 1 of clause 3.2.3.

Conforming charmaps specify certain character and character set attributes, as defined in 5.1.

5.1 Character Set Description Text

The character set description text (charmap) describes the mapping between symbolic character names and actual encoding of a coded character set. It is used to bind the symbolic character names in a FDCC-set to an actual encoding, so an application can process data in this encoding.

The following declarations can precede the character definitions. Each consist of the symbol shown in the following list, starting in column 1, including the surrounding brackets, followed by one of more "blank"s, followed by the value to be assigned to the symbol. If any of the declarations are included, they are specified in the order shown in the following list:

- `<code_set_name>`: The name of the coded character set for which the character set description text is defined. The characters of the name are taken from the set of characters with visible glyphs defined in Table 1 of clause 3.2.3.
- `<mb_cur_max>`: The maximum number of bytes in a multibyte character. This defaults to 1.
- `<mb_cur_min>`: An unsigned positive integer value that defines the minimum number of bytes in a character for the encoded character set. The value is less or equal to "mb_cur_max". If not specified, the minimum number is equal to "mb_cur_max".
- `<escape_char>`: 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.

The name of the repertoiremap used to define the symbolic character names in the charmap. The characters of the name are taken from the set of characters with visible glyphs defined in Table 1 of clause 3.2.3.

defines the escape sequences for ISO 2022 shifting for the coded character set defined by the charmap. The semicolon-separated operands are all strings with characters taken from the set of characters with visible glyphs defined in table 1. The first operand defines the g-set or c-set to be defined, and the following values are defined: c0, c1, g0, g1, g2, g3. The second operand defines what range of characters in the charmap is affected, and the values defined are: c0, c1, g0, g1. The third operand is the escape sequence that is defined.

the name of the charmap to be added to the current coded character set, and to be selected by the escape sequences defined by <escseq2022> of the added charmap.

include the encoding of another charmap in the current charmap. The semicolon-separated operands are all strings with characters taken from the set of characters with visible glyphs defined in table 1. The first operand defines the g-set or c-set to be defined in the current charmap, and the following values are defined: c0, c1, g0, g1, g2, g3. The second operand defines a range of characters in the referenced charmap, and the values defined are: c0, c1, g0, g1. The third operand is the name of the charmap to be included. The coded character sets are defined initially for the encoding, and therefore do not need escape sequences for identification. If two g0 sets are defined, the second is switched to using the SHIFT OUT control character, while the first is shifted to using the SHIFT IN control character.

The character set mapping definitions are all the lines immediately following an identifier line containing the string "CHARMAP" starting in column 1, and preceding a trailer line containing the string "END CHARMAP" starting in column 1. Empty lines and lines containing a <comment_char> in the first column are ignored. Each non-comment line of the character set mapping definition (i.e., between the "CHARMAP" and "END CHARMAP" lines of the text) is in one of the following syntaxes.

"%s %s %s
", <symbolic-name>,<encoding>,<comments>

"%s...%s %s %s
", <symbolic-name>,<symbolic-name>,<encoding>,<comments>

"%s....%s %s %s
", <symbolic-name>,<symbolic-name>,<encoding>,<comments>

"%s..%s %s %s
", <symbolic-name>,<symbolic-name>,<encoding>,<comments>
In the first syntax, the line of the character set mapping definition starts with the symbolic name, immediately preceded by a `<less-than>` character and immediately followed by a `<greater-than>` character. Symbolic names only contain characters from the set shown with a visible glyph in Table 1 of clause 3.2.3.

The same symbolic name may occur several times, with different values. The first value is the one used when generating an encoding, while the other values are accepted in decoding. Symbolic names may be included to identify values that can overlap with each other or with the values of the symbolic names shown in Table 1 of clause 3.2.3. It is possible to specify symbolic names for which no encoding exists in the encoded character set, by not specifying a value.

In the second and third syntax (symbolic decimal ellipsis), the line in the character set mapping defines a range of one or more symbolic names. The difference between the second and the third syntax is the number of dots in the ellipsis: the second has 3 dots, the third has 4 dots. In these forms the symbolic names consist of zero or more nonnumeric characters from the set shown with visible glyphs in Table 1 of clause 3.2.3, followed by an integer formed by one or more decimal digits. The characters preceding the integer are identical in the two symbolic names, and the integer formed by the digits in the second symbolic name are identical to or greater than the integer formed by the digits in the first name. This is interpreted as a series of symbolic names formed from the common part and each of the integers in decimal format between the first and the second integer, inclusive, and with a length of the symbolic names generated that is equal to the length of the first (and also the second) symbolic name. As an example, `<j0101>....<j0104>` is interpreted as the symbolic names `<j0101>`, `<j0102>`, `<j0103>`, and `<j0104>`, in that order.

Note: The rationale to allow both a 3-dot and a 4-dot symbol for symbolic decimal ellipses is that in the POSIX standard the decimal symbolic ellipses was defined by a 3-dot symbol for charmaps, while the 3-dot symbol was an absolute ellipses for POSIX locales, and this Technical Report specifies a 4-dot symbol for the decimal symbolic ellipses. The 3-dot symbolic decimal ellipses in charmaps is deprecated.

In the fourth syntax (symbolic hexadecimal ellipsis, with two dots), the line in the character set mapping defines a range of one or more symbolic names. In this form the symbolic names consist of zero or more nonnumeric characters from the set shown with visible glyphs in Table 1 of clause 3.2.3, followed by an integer formed by one or more hexadecimal digits, using uppercase letters only for the range "A" to "F". The characters preceding the hexadecimal integer are identical in the two symbolic names, and the integer formed by the hexadecimal digits in the second symbolic name is identical to or greater than the integer formed by the hexadecimal digits in the first name. This is interpreted as a series of symbolic names formed from the common part and each of the integers in hexadecimal format using uppercase letters only between the first and the second integer, inclusive, and with a length of the symbolic names generated that is equal to the length of the first (and also the second) symbolic name. As an example, `<U010E>..<U0111>` is interpreted as the symbolic names `<U010E>`, `<U010F>`, `<U0110>`, and `<U0111>`, in that order.

The encoding part is expressed as one (for single-byte values) or more concatenated decimal, octal or hexadecimal constants (hexadecimal constants are recommended). Decimal constants are represented by two or three decimal digits, preceded by the escape character and the lowercase letter "d"; for example /d05, /d97, or /d143. Hexadecimal constants are represented by two hexadecimal digits, preceded by the escape character and the lowercase letter "x"; for example /x05, /x61, or /x8f. Octal constants are represented by two or three octal digits, preceded by the escape character; for example /05, /141, or...
In a charmap, each constant should represent an 8 bit byte for portability reasons. Applications supporting other byte sizes may allow constants to represent values larger than those that can be represented in 8 bit bytes, and to allow additional digits in constants. When constants are concatenated for multibyte character values, they may be of different types, and interpreted in byte order from the first to the last with the least significant byte of the multibyte character specified by the last byte. The manner in which these constants are represented in the character stored in the system is application defined. Omitting bytes from a multibyte character produces undefined results.

In lines defining ranges of symbolic names, the encoded value is the value for the first symbolic name in the range (the symbolic name preceding the ellipsis). Subsequent symbolic names defined by the range have encoding values in increasing order. For example the line

\texttt{\textbackslash j0101}....\texttt{\textbackslash j0104}        /d129/d254

is interpreted as

\texttt{\textbackslash j0101}        /d129/d254
\texttt{\textbackslash j0102}        /d129/d255
\texttt{\textbackslash j0103}        /d130/d000
\texttt{\textbackslash j0104}        /d130/d001

The comments parameter is optional.

Example of using ISO 2022 techniques:

The following example defines two coded character sets, a 7-bit and a 14-bit. They are then merged into one encoding. It is an example on how encodings used in Eastern Asia could be specified.

The 7-bit charmap

\begin{verbatim}
<escape_char> /
<comment_char> %
% The 7-bit charmap defines both control and graphic characters
<code_set_name>  "eastern7bit"
<esqseq2022>     "c0";"c0","/x21/x40"
<esqseq2022>     "g0";"g0","/x28/x48"
<esqseq2022>     "g1";"g0","/x29/x48"
<esqseq2022>     "g2";"g0","/x2A/x48"
<esqseq2022>     "g3";"g0","/x2B/x48"
CHARMAP
<tab>            /x08
<newline>        /x0D
<a>             /x61
% more character encodings to be defined here
END CHARMAP
\end{verbatim}

The 14-bit charmap

\begin{verbatim}
<escape_char> /
<comment_char> %
<code_set_name>  "eastern14bit"
<mb_cur_max>      2
<esqseq2022>     "g0";"g0","/x24/x40"
<esqseq2022>     "g1";"g0","/x24/x29/x40"
<esqseq2022>     "g2";"g0","/x24/x2A/x40"
<esqseq2022>     "g3";"g0","/x24/x2B/x40"
CHARMAP
<U0165>           /d036/d055   % the character codes are only examples
<U0166>           /d036/d056
\end{verbatim}
The merged encoding

\[
\begin{align*}
<\text{escape_char}> & / \\
<\text{comment_char}> & \% \\
<\text{code_set_name}> & \text{"shift-eastern"} \\
<\text{mb_cur_max}> & 2 \\
<\text{mb_cur_min}> & 1 \\
<\text{include}> & \text{"c0";"c0";"eastern7bit"} \\
<\text{include}> & \text{"g0";"g0";"eastern7bit"} \\
<\text{include}> & \text{"g1";"g0";"eastern14bit"} \\
\% & \text{This defines the g0 values of "eastern14bit" (without the 8th bit set) to be the g1 in this encoding (with the 8th bit set).} \\
\% & \text{So the bytes without the 8th bit set is from the "eastern7bit" coded character set, while bytes with the 8th bit set are from the 14-bit set.}
\end{align*}
\]

Another merged encoding using the same charmaps:

\[
\begin{align*}
<\text{escape_char}> & / \\
<\text{comment_char}> & \% \\
<\text{code_set_name}> & \text{"EUC-eastern"} \\
<\text{mb_cur_max}> & 2 \\
<\text{mb_cur_min}> & 1 \\
<\text{include}> & \text{"c0";"c0";"eastern7bit"} \\
<\text{include}> & \text{"g0";"g0";"eastern7bit"} \\
<\text{include}> & \text{"g0";"g0";"eastern14bit"} \\
\% & \text{As there are two "g0" sets defined, the first referenced is the initial g0 set, while the second can be shifted to via the SHIFT OUT control character. The first can then be shifted to by the SHIFT IN control character.}
\end{align*}
\]

**WIDTH section**

After the "END CHARMAP" statement the following declarations may follow. Each consists of the keyword shown in the following list, starting in column 1, followed by the value(s) to be associated to the keyword, as defined below.

**WIDTH** An unsigned positive integer value defining the column width for the characters in the coded character set. Coded character values are defined using symbolic character names followed by a column width value. Defining a character with more than one WIDTH produces undefined results. The END WIDTH keyword is used to terminate the WIDTH definitions.

**WIDTH_DEFAULT** An unsigned positive integer value defining the column width for any character not listed by one of the WIDTH keywords. If no WIDTH_DEFAULT keyword is included in the cmap, the default character width is 1.

Example:

After the "END CHARMAP" statement, a syntax for width definition would be:

\[
\begin{align*}
\text{WIDTH} \\
<\text{A}> & 1 \\
<\text{B}> & 1 \\
<\text{j0101}>...<\text{j0195}> & 2 \\
<\text{U4E00}>...<\text{U9FA5}> & 2 \\
\text{END WIDTH} \\
\text{WIDTH_DEFAULT} & 1
\end{align*}
\]
In this example, the code point values represented by <A> and <B> are assigned a width of 1. The code point values <j0101>...<j0195> (decimal ellipses) and <U9FA5> are assigned a width of 2. The last line defines the DEFAULT_WIDTH to 1.

6  REPERTOIREMAP (controversial)

FDCC-set and Charmap sources may be specified in a coded character set independent way, using symbolic character names. The relation between the symbolic character names and characters may be specified via a Repertoiremap, which defines the repertoire of characters defined for a FDCC-set, and the symbolic character names and corresponding abstract character (by a reference to ISO/IEC 10646).

The repertoire mapping is defined by specifying the symbolic character name and the ISO/IEC 10646 code position in hexadecimal form (with a preceding 'U') and optionally the long ISO/IEC 10646 character name in the following syntax:

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

The symbolic character name and the short identifier are each surrounded by angle brackets <>, and the fields are separated by one or more spaces or tabs on a line. If a right angle bracket or an escape character is used within a symbolic name, it is preceded by the escape character. The short identifier is either a ISO/IEC 10646 short identifier, or, if that does not exists, a short identifier in the range <P0000>..<PFFFF> or <P00000000>..<P7FFFFFFF>.

The escape character can be redefined from the default reverse solidus (\) with the first line of the Repertoiremap containing the string "escape_char" followed by one or more spaces or tabs and then the escape character.

Several symbolic character names can refer to the same abstract character, and are then used as synonyms in FDCC-sets and charmaps. The set of <U0000>..<UFFFFF> and <U00000000>..<U7FFFFFFF> symbolic names (no lowercase letters) are predefined and refer to the corresponding code points of ISO/IEC 10646 with the same short identifier.

The "i18nrep" repertoiremap is defined to accommodate prior art, such as defined in Annex G of the ISO/IEC 9945-2:1993 standard, and used by ISO and IEC member bodies in their national POSIX locale specifications, and as used in POSIX locales distributed by the ISO/IEC POSIX working group and The Open Group. Many POSIX charmaps registered with ISO/IEC 15897 use these symbolic names. It also reflects use on the Internet, and many of the Internet registered charsets are specified using these symbolic names. The "i18nrep" repertoiremap thus facilitates reuse of both POSIX locale data and POSIX charmaps with data from this Technical Report. The sequence <a8>..<z8> are used as hooks for tailoring to denote the last accented Latin letter of each of the ISO/IEC 646 letters <a>..<z>, so that tailorings that need to have specifications after the last letter of such a family, for example to introduce a new letter of an alphabet, can do so with a reference that is stable over different versions of the "i18n" FDCC-set. The "i18nrep" repertoiremap is not intended to cover the complete repertoire that the "i18n" FDCC-set covers. The contents of the "i18nrep" repertoiremap is as follows:

escape_char /
<U0000> NULL (NUL)
<U0001> START OF HEADING (SOH)
<U0002> START OF TEXT (STX)
<U0003> END OF TEXT (ETX)
<i8> Weight indicating the position of the last i
<P0009> Weight indicating the position of the last j
<K8> Weight indicating the position of the last k
<P0011> Weight indicating the position of the last l
<M8> Weight indicating the position of the last m
<P0013> Weight indicating the position of the last n
<N8> Weight indicating the position of the last o
<P0015> Weight indicating the position of the last p
<P0017> Weight indicating the position of the last q
<R8> Weight indicating the position of the last r
<P0019> Weight indicating the position of the last s
<T8> Weight indicating the position of the last t
<br> Weight indicating the position of the last u
<br> Weight indicating the position of the last v
<br> Weight indicating the position of the last w
<br> Weight indicating the position of the last x
<br> Weight indicating the position of the last y
<br> Weight indicating the position of the last z

<NU> NULL (NUL)
<SH> START OF HEADING (SOH)
<SX> START OF TEXT (STX)
<EX> END OF TEXT (ETX)
<ET> END OF TRANSMISSION (EOT)
<EQ> ENQUIRY (ENQ)
<AK> ACKNOWLEDGE (ACK)
<BL> BELL (BEL)
<BS> BACKSPACE (BS)
<HT> CHARACTER TABULATION (HT)
<LF> LINE FEED (LF)
<VT> LINE TABULATION (VT)
<FF> FORM FEED (FF)
<CR> CARRIAGE RETURN (CR)
<C0> SHIFT OUT (SO)
<S1> SHIFT IN (SI)
<D1> DATALINK ESCAPE (DLE)
<D2> DEVICE CONTROL ONE (DC1)
<D3> DEVICE CONTROL TWO (DC2)
<D4> DEVICE CONTROL THREE (DC3)
<D5> DEVICE CONTROL FOUR (DC4)
<NK> NEGATIVE ACKNOWLEDGE (NAK)
<SY> SYNCHRONOUS IDLE (SYN)
<EB> END OF TRANSMISSION BLOCK (ETB)
<CN> CANCEL (CAN)
<EM> END OF MEDIUM (EM)
<SB> SUBSTITUTE (SUB)
<EC> ESCAPE (ESC)
<FS> FILE SEPARATOR (FS)
<GS> GROUP SEPARATOR (GS)
<RS> RECORD SEPARATOR (RS)
<US> UNIT SEPARATOR (US)
<DT> DELETE (DEL)
<PA> PAD CHAR (PAD)
<BS> HIGH OCTET PRESET (BOP)
<br> BREAK PERMITTED HERE (BPH)
<br> NO BREAK HERE (NBH)
<br> INDEX (IND)
<br> NEXT LINE (NEL)
<br> START OF SELECTED AREA (SSA)
<br> END OF SELECTED AREA (ESA)
<br> CHARACTER TABULATION SET (HTS)
<br> CHARACTER TABULATION WITH JUSTIFICATION (HTJ)
<br> LINE TABULATION SET (LTS)
<br> PARTIAL LINE FORWARD (PLF)
<br> PARTIAL LINE BACKWARD (PLB)
<br> REVERSE LINE FEED (RLL)
<br> SINGLE-SHIFT TWO (SS2)
<br> SINGLE-SHIFT THREE (SS3)
<br> DEVICE CONTROL STRING (DQS)
<br> PRIVATE USE ONE (PUI)
<br> PRIVATE USE TWO (PUS)
<br> SET TRANSMIT STATE (STS)
<br> CANCEL CHARACTER (CHC)
<br> MESSAGE WAITING (MWS)
<br> START OF GUARDED AREA (SPA)
<br> END OF GUARDED AREA (EPA)
<br> START OF STRING (SOS)
<br> SINGLE GRAPHIC CHARACTER INTRODUCER (SGCI)
<br> SINGLE CHARACTER INTRODUCER (SCI)
<br> CONTROL SEQUENCE INTRODUCER (CSI)
<br> STRING TERMINATOR (ST)
<br> OPERATING SYSTEM COMMAND (OSC)
<br> PRIVACY MESSAGE (PM)
<br> APPLICATION PROGRAM COMMAND (APC)
<br> SPACE
<br> EXCLAMATION MARK
<br> QUOTATION MARK
<br> NUMBER SIGN
<!!> VERTICAL LINE
<!)> RIGHT CURLY BRACKET
<!?> TILDE
<NS> NO-BREAK SPACE
<IN> INVERTED EXCLAMATION MARK
<CENT> CENT SIGN
<POUND> POUND SIGN
<CURRE> CURRENCY SIGN
<yen> YEN SIGN
<BRAK> BROKEN RAP
<SECTION> SECTION SIGN
<DIAC> DIAERESIS
<COPYRIGHT> COPYRIGHT SIGN
<FEMININE> FEMININE ORDINAL INDICATOR
<LEFT> LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
<NOT> NOT SIGN
<SOFT> SOFT HYPHEN
<REGISTERED> REGISTERED SIGN
<MACRON> MACRON
<DEGREE> DEGREE SIGN
<PLUS-Minus> PLUS-MINUS SIGN
<SUPERScript TWO> SUPERSCRIPRT TWO
<SUPERScript THREE> SUPERSCRIPRT THREE
<ACUTE ACCENT> ACUTE ACCENT
<MICRO> MICRO SIGN
<PILCROW> PILCROW SIGN
<MIDDLE DOT> MIDDLE DOT
<CEDILLA> CEDILLA
<SUPERSCRIPT ONE> SUPERSCRIPT ONE
<MASCULINE> MASCULINE ORDINAL INDICATOR
<RIGHT POINTING DOUBLE ANGLE QUOTATION MARK
<VULGAR FRACTION ONE QUARTER> VULGAR FRACTION ONE QUARTER
<VULGAR FRACTION HALF> VULGAR FRACTION ONE QUARTER
<VULGAR FRACTION THREE QUARTERS> VULGAR FRACTION THREE QUARTERS
<INVERTED QUESTION MARK>
<LATIN CAPITAL LETTER A WITH GRAVE>
<LATIN CAPITAL LETTER A WITH ACUTE>
<LATIN CAPITAL LETTER A WITH CIRCUMFLEX>
<LATIN CAPITAL LETTER A WITH TILDE>
<LATIN CAPITAL LETTER A WITH DIAERESIS>
<LATIN CAPITAL LETTER AE (ash)>
<LATIN CAPITAL LETTER C WITH CEDILLA>
<LATIN CAPITAL LETTER E WITH GRAVE>
<LATIN CAPITAL LETTER E WITH ACUTE>
<LATIN CAPITAL LETTER E WITH CIRCUMFLEX>
<LATIN CAPITAL LETTER E WITH DIAERESIS>
<LATIN CAPITAL LETTER I WITH GRAVE>
<LATIN CAPITAL LETTER I WITH ACUTE>
<LATIN CAPITAL LETTER I WITH CIRCUMFLEX>
<LATIN CAPITAL LETTER ETH (Icelandic)>
<LATIN CAPITAL LETTER N WITH TILDE>
<LATIN CAPITAL LETTER O WITH GRAVE>
<LATIN CAPITAL LETTER O WITH ACUTE>
<LATIN CAPITAL LETTER O WITH CIRCUMFLEX>
<LATIN CAPITAL LETTER O WITH DIAERESIS>
<LATIN CAPITAL LETTER THER (Icelandic)>
<LATIN SMALL LETTER SHARP S (German)>
<LATIN SMALL LETTER A WITH GRAVE>
<LATIN SMALL LETTER A WITH ACUTE>
<LATIN SMALL LETTER A WITH CIRCUMFLEX>
<LATIN SMALL LETTER A WITH TILDE>
<LATIN SMALL LETTER A WITH DIAERESIS>
<LATIN SMALL LETTER C WITH CEDILLA>
<LATIN SMALL LETTER E WITH GRAVE>
<LATIN SMALL LETTER E WITH ACUTE>
<LATIN SMALL LETTER E WITH CIRCUMFLEX>
<LATIN SMALL LETTER E WITH DIAERESIS>
<LATIN SMALL LETTER I WITH GRAVE>
<LATIN SMALL LETTER I WITH ACUTE>
<LATIN SMALL LETTER I WITH CIRCUMFLEX>
<LATIN SMALL LETTER I WITH DIAERESIS>
<LATIN SMALL LETTER N WITH TILDE>
<LATIN SMALL LETTER O WITH GRAVE>
<LATIN SMALL LETTER O WITH ACUTE>
<LATIN SMALL LETTER O WITH CIRCUMFLEX>
GREEK SMALL LETTER IOTA WITH DIALYTIKA AND TONOS
GREEK CAPITAL LETTER ALPHA
GREEK CAPITAL LETTER BETA
GREEK CAPITAL LETTER GAMMA
GREEK CAPITAL LETTER DELTA
GREEK CAPITAL LETTER EPSILON
GREEK CAPITAL LETTER ETA
GREEK CAPITAL LETTER THETA
GREEK CAPITAL LETTER IOTA
GREEK CAPITAL LETTER KAPPA
GREEK CAPITAL LETTER LAMDA
GREEK CAPITAL LETTER MU
GREEK CAPITAL LETTER XI
GREEK CAPITAL LETTER OMICRON
GREEK CAPITAL LETTER PI
GREEK CAPITAL LETTER RHO
GREEK CAPITAL LETTER SIGMA
GREEK CAPITAL LETTER TAU
GREEK CAPITAL LETTER UPSILON
GREEK CAPITAL LETTER PHI
GREEK CAPITAL LETTER CHI
GREEK CAPITAL LETTER PSI
GREEK CAPITAL LETTER OMEGA
GREEK SMALL LETTER IOTA WITH DIALYTIKA
GREEK SMALL LETTER UPSILON WITH DIALYTIKA
GREEK SMALL LETTER ALPHA WITH TONOS
GREEK SMALL LETTER EPSILON WITH TONOS
GREEK SMALL LETTER ETA WITH TONOS
GREEK SMALL LETTER IOTA WITH TONOS
GREEK SMALL LETTER UPSILON WITH DIALYTIKA AND TONOS
GREEK SMALL LETTER ALPHA
GREEK SMALL LETTER BETA
GREEK SMALL LETTER GAMMA
GREEK SMALL LETTER DELTA
GREEK SMALL LETTER EPSILON
GREEK SMALL LETTER ETA
GREEK SMALL LETTER THETA
GREEK SMALL LETTER IOTA
GREEK SMALL LETTER KAPPA
GREEK SMALL LETTER LAMDA
GREEK SMALL LETTER MU
GREEK SMALL LETTER XI
GREEK SMALL LETTER OMICRON
GREEK SMALL LETTER PI
GREEK SMALL LETTER RHO
GREEK SMALL LETTER FINAL SIGMA
GREEK SMALL LETTER SIGMA
GREEK SMALL LETTER TAU
GREEK SMALL LETTER UPSILON
GREEK SMALL LETTER PHI
GREEK SMALL LETTER CHI
GREEK SMALL LETTER PSI
GREEK SMALL LETTER OMEGA
GREEK SMALL LETTER IOTA WITH DIALYTIKA
GREEK SMALL LETTER UPSILON WITH DIALYTIKA
GREEK SMALL LETTER OMICRON WITH TONOS
GREEK SMALL LETTER UPSILON WITH TONOS
GREEK SMALL LETTER OMELGA WITH TONOS
GREEK BETA SYMBOL
GREEK LETTER SIGMA
GREEK LETTER DIGAMMA
GREEK LETTER KOPPA
GREEK LETTER SAMPI
CYRILLIC CAPITAL LETTER IO
CYRILLIC CAPITAL LETTER DJE (Serbocroatian)
CYRILLIC CAPITAL LETTER GJE
CYRILLIC CAPITAL LETTER UKRAINIAN IE
CYRILLIC CAPITAL LETTER DZE
CYRILLIC CAPITAL LETTER BYELORUSSIAN-UKRAINIAN I
CYRILLIC CAPITAL LETTER VI (Ukrainian)
CYRILLIC CAPITAL LETTER JE
CYRILLIC CAPITAL LETTER LJE
CYRILLIC CAPITAL LETTER NJE
CYRILLIC CAPITAL LETTER TSHE (Serbocroatian)
CYRILLIC CAPITAL LETTER KJE
CYRILLIC CAPITAL LETTER SHORT U (Byelorussian)
CYRILLIC CAPITAL LETTER DSHE
CYRILLIC CAPITAL LETTER A
CYRILLIC CAPITAL LETTER BE
CYRILLIC CAPITAL LETTER VE
CYRILLIC CAPITAL LETTER GHE
CYRILLIC CAPITAL LETTER DE
CYRILLIC CAPITAL LETTER IE
<table>
<thead>
<tr>
<th>Character</th>
<th>Unicode Codepoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;r&gt;</code></td>
<td><code>&lt;U1E59&gt;</code></td>
<td>LATIN SMALL LETTER R WITH DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;R.-&gt;</code></td>
<td><code>&lt;U1E5A&gt;</code></td>
<td>LATIN CAPITAL LETTER R WITH DOT BELOW AND MACRON</td>
</tr>
<tr>
<td><code>&lt;r.-&gt;</code></td>
<td><code>&lt;U1E5B&gt;</code></td>
<td>LATIN SMALL LETTER R WITH DOT BELOW AND MACRON</td>
</tr>
<tr>
<td><code>&lt;R--&gt;</code></td>
<td><code>&lt;U1E5C&gt;</code></td>
<td>LATIN CAPITAL LETTER R WITH LINE BELOW AND MACRON</td>
</tr>
<tr>
<td><code>&lt;r--&gt;</code></td>
<td><code>&lt;U1E5D&gt;</code></td>
<td>LATIN SMALL LETTER R WITH LINE BELOW AND MACRON</td>
</tr>
<tr>
<td><code>&lt;R_&gt;</code></td>
<td><code>&lt;U1E5E&gt;</code></td>
<td>LATIN CAPITAL LETTER R WITH LINE BELOW</td>
</tr>
<tr>
<td><code>&lt;r_&gt;</code></td>
<td><code>&lt;U1E5F&gt;</code></td>
<td>LATIN SMALL LETTER R WITH LINE BELOW</td>
</tr>
<tr>
<td><code>&lt;S.</code></td>
<td><code>&lt;U1E60&gt;</code></td>
<td>LATIN CAPITAL LETTER S WITH DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;s.</code></td>
<td><code>&lt;U1E61&gt;</code></td>
<td>LATIN SMALL LETTER S WITH DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;S-.&gt;</code></td>
<td><code>&lt;U1E62&gt;</code></td>
<td>LATIN CAPITAL LETTER S WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;s-.&gt;</code></td>
<td><code>&lt;U1E63&gt;</code></td>
<td>LATIN SMALL LETTER S WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;S</code>.&gt;</td>
<td><code>&lt;U1E64&gt;</code></td>
<td>LATIN CAPITAL LETTER S WITH ACUTE AND DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;s</code>.&gt;</td>
<td><code>&lt;U1E65&gt;</code></td>
<td>LATIN SMALL LETTER S WITH ACUTE AND DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;S-.&gt;</code></td>
<td><code>&lt;U1E66&gt;</code></td>
<td>LATIN CAPITAL LETTER S WITH DOT BELOW AND MACRON</td>
</tr>
<tr>
<td><code>&lt;s-.&gt;</code></td>
<td><code>&lt;U1E67&gt;</code></td>
<td>LATIN SMALL LETTER S WITH DOT BELOW AND MACRON</td>
</tr>
<tr>
<td><code>&lt;T.</code></td>
<td><code>&lt;U1E68&gt;</code></td>
<td>LATIN CAPITAL LETTER T WITH DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;t.</code></td>
<td><code>&lt;U1E69&gt;</code></td>
<td>LATIN SMALL LETTER T WITH DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;T-&gt;</code></td>
<td><code>&lt;U1E6A&gt;</code></td>
<td>LATIN CAPITAL LETTER T WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;t-&gt;</code></td>
<td><code>&lt;U1E6B&gt;</code></td>
<td>LATIN SMALL LETTER T WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;T.</code></td>
<td><code>&lt;U1E6C&gt;</code></td>
<td>LATIN CAPITAL LETTER T WITH LINE BELOW</td>
</tr>
<tr>
<td><code>&lt;t.</code></td>
<td><code>&lt;U1E6D&gt;</code></td>
<td>LATIN SMALL LETTER T WITH LINE BELOW</td>
</tr>
<tr>
<td><code>&lt;U--&gt;</code></td>
<td><code>&lt;U1E6E&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH DIAERESIS BELOW</td>
</tr>
<tr>
<td><code>&lt;u--&gt;</code></td>
<td><code>&lt;U1E6F&gt;</code></td>
<td>LATIN SMALL LETTER U WITH DIAERESIS BELOW</td>
</tr>
<tr>
<td><code>&lt;U-.&gt;</code></td>
<td><code>&lt;U1E70&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH TILDE BELOW</td>
</tr>
<tr>
<td><code>&lt;u-.&gt;</code></td>
<td><code>&lt;U1E71&gt;</code></td>
<td>LATIN SMALL LETTER U WITH TILDE BELOW</td>
</tr>
<tr>
<td><code>&lt;U?.&gt;</code></td>
<td><code>&lt;U1E72&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH TILDE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;u?.&gt;</code></td>
<td><code>&lt;U1E73&gt;</code></td>
<td>LATIN SMALL LETTER U WITH TILDE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;U?&gt;</code></td>
<td><code>&lt;U1E74&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH TILDE AND DIACRITICAL MARK</td>
</tr>
<tr>
<td><code>&lt;u?&gt;</code></td>
<td><code>&lt;U1E75&gt;</code></td>
<td>LATIN SMALL LETTER U WITH TILDE AND DIACRITICAL MARK</td>
</tr>
<tr>
<td><code>&lt;U-?&gt;</code></td>
<td><code>&lt;U1E76&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH TILDE AND MACRON</td>
</tr>
<tr>
<td><code>&lt;u-?&gt;</code></td>
<td><code>&lt;U1E77&gt;</code></td>
<td>LATIN SMALL LETTER U WITH TILDE AND MACRON</td>
</tr>
<tr>
<td><code>&lt;U-?&gt;</code></td>
<td><code>&lt;U1E78&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH TILDE AND U+0301</td>
</tr>
<tr>
<td><code>&lt;u-?&gt;</code></td>
<td><code>&lt;U1E79&gt;</code></td>
<td>LATIN SMALL LETTER U WITH TILDE AND U+0301</td>
</tr>
<tr>
<td><code>&lt;U-.</code></td>
<td><code>&lt;U1E7A&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH DIAERESIS AND DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;u-.</code></td>
<td><code>&lt;U1E7B&gt;</code></td>
<td>LATIN SMALL LETTER U WITH DIAERESIS AND DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;U-!&gt;</code></td>
<td><code>&lt;U1E7C&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH TILDE AND GRAVE</td>
</tr>
<tr>
<td><code>&lt;u-!&gt;</code></td>
<td><code>&lt;U1E7D&gt;</code></td>
<td>LATIN SMALL LETTER U WITH TILDE AND GRAVE</td>
</tr>
<tr>
<td><code>&lt;U!&gt;</code></td>
<td><code>&lt;U1E7E&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH TILDE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;u!&gt;</code></td>
<td><code>&lt;U1E7F&gt;</code></td>
<td>LATIN SMALL LETTER U WITH TILDE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;U&gt;</code></td>
<td><code>&lt;U1E80&gt;</code></td>
<td>LATIN CAPITAL LETTER U WITH MORE DIYAERESIS</td>
</tr>
<tr>
<td><code>&lt;u&gt;</code></td>
<td><code>&lt;U1E81&gt;</code></td>
<td>LATIN SMALL LETTER U WITH MORE DIYAERESIS</td>
</tr>
<tr>
<td><code>&lt;W&gt;</code></td>
<td><code>&lt;U1E82&gt;</code></td>
<td>LATIN CAPITAL LETTER W WITH ACUTE</td>
</tr>
<tr>
<td><code>&lt;w&gt;</code></td>
<td><code>&lt;U1E83&gt;</code></td>
<td>LATIN SMALL LETTER W WITH ACUTE</td>
</tr>
<tr>
<td><code>&lt;W-&gt;</code></td>
<td><code>&lt;U1E84&gt;</code></td>
<td>LATIN CAPITAL LETTER W WITH DIAERESIS</td>
</tr>
<tr>
<td><code>&lt;w-&gt;</code></td>
<td><code>&lt;U1E85&gt;</code></td>
<td>LATIN SMALL LETTER W WITH DIAERESIS</td>
</tr>
<tr>
<td><code>&lt;W-!&gt;</code></td>
<td><code>&lt;U1E86&gt;</code></td>
<td>LATIN CAPITAL LETTER W WITH TILDE AND GRAVE</td>
</tr>
<tr>
<td><code>&lt;w-!&gt;</code></td>
<td><code>&lt;U1E87&gt;</code></td>
<td>LATIN SMALL LETTER W WITH TILDE AND GRAVE</td>
</tr>
<tr>
<td><code>&lt;W!&gt;</code></td>
<td><code>&lt;U1E88&gt;</code></td>
<td>LATIN CAPITAL LETTER W WITH TILDE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;w!&gt;</code></td>
<td><code>&lt;U1E89&gt;</code></td>
<td>LATIN SMALL LETTER W WITH TILDE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;W&gt;</code></td>
<td><code>&lt;U1E8A&gt;</code></td>
<td>LATIN CAPITAL LETTER W WITH HOOK ABOVE</td>
</tr>
<tr>
<td><code>&lt;w&gt;</code></td>
<td><code>&lt;U1E8B&gt;</code></td>
<td>LATIN SMALL LETTER W WITH HOOK ABOVE</td>
</tr>
<tr>
<td><code>&lt;W-!&gt;</code></td>
<td><code>&lt;U1E8C&gt;</code></td>
<td>LATIN CAPITAL LETTER W WITH TILDE AND HOOK ABOVE</td>
</tr>
<tr>
<td><code>&lt;w-!&gt;</code></td>
<td><code>&lt;U1E8D&gt;</code></td>
<td>LATIN SMALL LETTER W WITH TILDE AND HOOK ABOVE</td>
</tr>
<tr>
<td><code>&lt;W&gt;</code></td>
<td><code>&lt;U1E8E&gt;</code></td>
<td>LATIN CAPITAL LETTER W WITH TILDE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;w&gt;</code></td>
<td><code>&lt;U1E8F&gt;</code></td>
<td>LATIN SMALL LETTER W WITH TILDE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;Z&gt;</code></td>
<td><code>&lt;U1E90&gt;</code></td>
<td>LATIN CAPITAL LETTER Z WITH CIRCUMFLEX</td>
</tr>
<tr>
<td><code>&lt;z&gt;</code></td>
<td><code>&lt;U1E91&gt;</code></td>
<td>LATIN SMALL LETTER Z WITH CIRCUMFLEX</td>
</tr>
<tr>
<td><code>&lt;Z-&gt;</code></td>
<td><code>&lt;U1E92&gt;</code></td>
<td>LATIN CAPITAL LETTER Z WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;z-&gt;</code></td>
<td><code>&lt;U1E93&gt;</code></td>
<td>LATIN SMALL LETTER Z WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;Z&gt;</code></td>
<td><code>&lt;U1E94&gt;</code></td>
<td>LATIN CAPITAL LETTER Z WITH DIAERESIS</td>
</tr>
<tr>
<td><code>&lt;z&gt;</code></td>
<td><code>&lt;U1E95&gt;</code></td>
<td>LATIN SMALL LETTER Z WITH DIAERESIS</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EA0&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EA1&gt;</code></td>
<td>LATIN SMALL LETTER A WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;A-!&gt;</code></td>
<td><code>&lt;U1EA2&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH HOOK ABOVE</td>
</tr>
<tr>
<td><code>&lt;a-!&gt;</code></td>
<td><code>&lt;U1EA3&gt;</code></td>
<td>LATIN SMALL LETTER A WITH HOOK ABOVE</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EA4&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH HOOK AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EA5&gt;</code></td>
<td>LATIN SMALL LETTER A WITH HOOK AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EA6&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH HOOK AND GRAVE</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EA7&gt;</code></td>
<td>LATIN SMALL LETTER A WITH HOOK AND GRAVE</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EA8&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH HOOK AND ACUTE AND DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EA9&gt;</code></td>
<td>LATIN SMALL LETTER A WITH HOOK AND ACUTE AND DOT ABOVE</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EAA&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH HOOK AND MACRON</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EAB&gt;</code></td>
<td>LATIN SMALL LETTER A WITH HOOK AND MACRON</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EAC&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH HOOK AND DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EAD&gt;</code></td>
<td>LATIN SMALL LETTER A WITH HOOK AND DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EAE&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH BREVE</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EAF&gt;</code></td>
<td>LATIN SMALL LETTER A WITH BREVE</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EB0&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH BREVE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EB1&gt;</code></td>
<td>LATIN SMALL LETTER A WITH BREVE AND ACUTE</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EB2&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH BREVE AND GRAVE</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EB3&gt;</code></td>
<td>LATIN SMALL LETTER A WITH BREVE AND GRAVE</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EB4&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH BREVE AND TILDE</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EB5&gt;</code></td>
<td>LATIN SMALL LETTER A WITH BREVE AND TILDE</td>
</tr>
<tr>
<td><code>&lt;A&gt;</code></td>
<td><code>&lt;U1EB6&gt;</code></td>
<td>LATIN CAPITAL LETTER A WITH BREVE AND DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;a&gt;</code></td>
<td><code>&lt;U1EB7&gt;</code></td>
<td>LATIN SMALL LETTER A WITH BREVE AND DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;E&gt;</code></td>
<td><code>&lt;U1EB8&gt;</code></td>
<td>LATIN CAPITAL LETTER E WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;e&gt;</code></td>
<td><code>&lt;U1EB9&gt;</code></td>
<td>LATIN SMALL LETTER E WITH DOT BELOW</td>
</tr>
<tr>
<td><code>&lt;E&gt;</code></td>
<td><code>&lt;U1EBA&gt;</code></td>
<td>LATIN CAPITAL LETTER E WITH HOOK ABOVE</td>
</tr>
<tr>
<td><code>&lt;e&gt;</code></td>
<td><code>&lt;U1EBB&gt;</code></td>
<td>LATIN SMALL LETTER E WITH HOOK ABOVE</td>
</tr>
</tbody>
</table>
<RK> <U25A8> SQUARE WITH UPPER RIGHT TO LOWER LEFT FILL
<RB> <U25A9> SQUARE WITH DIAGONAL CROSSHATCH FILL
<SB> <U25AA> BLACK SMALL SQUARE
<SR> <U25AC> BLACK RECTANGLE
<Or> <U25AD> WHITE RECTANGLE
<UT> <U25B2> BLACK UP-POINTING TRIANGLE
<uT> <U25B3> WHITE UP-POINTING TRIANGLE
<Tr> <U25B7> WHITE RIGHT-POINTING TRIANGLE
<PR> <U25BA> BLACK RIGHT-POINTING POINTER
<Db> <U25C6> BLACK DIAMOND
<LZ> <U25CA> LOZENGE
<0m> <U25CB> WHITE CIRCLE
<0o> <U25CE> BULLSEYE
<0M> <U25CF> BLACK CIRCLE
<0L> <U25D0> CIRCLE WITH LEFT HALF BLACK
<0R> <U25D1> CIRCLE WITH RIGHT HALF BLACK
<Sn> <U25D8> INVERSE BULLET
<In> <U25D9> INVERSE WHITE CIRCLE
<Dr> <U25DA> BLACK LOWER RIGHT TRIANGLE
<Dr> <U25DB> BLACK LOWER LEFT TRIANGLE
<Ci> <U25DC> LARGE CIRCLE
<Tr> <U25DD> BLACK STAR
<0t> <U25DE> WHITE STAR
<Tel> <U260E> BLACK TELEPHONE
<tel> <U260F> WHITE TELEPHONE
<Ch> <U261C> WHITE LEFT POINTING INDEX
</h> <U261D> WHITE RIGHT POINTING INDEX
</u> <U261E> WHITE RIGHT POINTING INDEX
<0u> <U263A> WHITE SMILING FACE
<0u> <U263B> BLACK SMILING FACE
<u> <U263C> WHITE SUN WITH RAYS
<Fm> <U2640> FEMALE SIGN
<Ml> <U2642> MALE SIGN
<cs> <U2660> BLACK SPADE SUIT
<ch> <U2661> WHITE HEART SUIT
<cd> <U2662> WHITE DIAMOND SUIT
<cc> <U2663> BLACK CLUB SUIT
<cs> <U2664> WHITE SPADE SUIT
<ch> <U2665> BLACK HEART SUIT
<cd> <U2666> BLACK DIAMOND SUIT
<cc> <U2667> WHITE CLUB SUIT
<Md> <U2669> QUARTER NOTE
<M> <U266A> EIGHTH NOTE
<M> <U266B> BEAMED EIGHTH NOTES
<M> <U266C> BEAMED SIXTEENTH NOTES
<M> <U266D> MUSIC FLAT SIGN
<M> <U266E> MUSIC NATURAL SIGN
<M> <U266F> MUSIC SHARP SIGN
<OK> <U2713> CHECK MARK
<xx> <U2717> BALLOT X
<-> <U2720> MALTESE CROSS
<IS> <U3000> IDEOGRAPHIC SPACE
<>,<> <U3001> IDEOGRAPHIC COMMA
,<>,<> <U3002> IDEOGRAPHIC FULL STOP
<""> <U3003> DITTO MARK
<jis> <U3004> JAPANESE INDUSTRIAL STANDARD SYMBOL
<jis> <U3005> IDEOGRAPHIC ITERATION MARK
</> <U3006> IDEOGRAPHIC CLOSING MARK
<0> <U3007> IDEOGRAPHIC NUMBER ZERO
<0> <U3008> LEFT DOUBLE ANGLE BRACKET
<""> <U3009> RIGHT DOUBLE ANGLE BRACKET
<""> <U300A> LEFT CORNER BRACKET
<""> <U300B> RIGHT CORNER BRACKET
<""> <U300C> LEFT WHITE CORNER BRACKET
<""> <U300D> RIGHT WHITE CORNER BRACKET
<""> <U300E> LEFT BLACK CORNER BRACKET
<""> <U300F> RIGHT BLACK CORNER BRACKET
<""> <U3010> LEFT BLACK LENTICULAR BRACKET
<""> <U3011> RIGHT BLACK LENTICULAR BRACKET
<""> <U3012> POSTAL MARK
<""> <U3013> GETA MARK
<""> <U3014> LEFT TORTOISE SHELL BRACKET
<""> <U3015> RIGHT TORTOISE SHELL BRACKET
<""> <U3016> LEFT WHITE LENTICULAR BRACKET
<""> <U3017> RIGHT WHITE LENTICULAR BRACKET
<""> <U3018> WAVE DASH
<:<:> <U3020> POSTAL MARK FACE
<as> <U3041> HIRAGANA LETTER SMALL A
<as> <U3042> HIRAGANA LETTER A
<as> <U3043> HIRAGANA LETTER SMALL I
<as> <U3044> HIRAGANA LETTER I
<as> <U3045> HIRAGANA LETTER SMALL U
<as> <U3046> HIRAGANA LETTER U
<as> <U3047> HIRAGANA LETTER SMALL E
<as> <U3048> HIRAGANA LETTER E
<as> <U3049> HIRAGANA LETTER SMALL O
<table>
<thead>
<tr>
<th>U+306E</th>
<th>KATAKANA LETTER O</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+306A</td>
<td>KATAKANA LETTER KA</td>
</tr>
<tr>
<td>U+306C</td>
<td>Katakana Letter GO</td>
</tr>
<tr>
<td>U+306D</td>
<td>KATAKANA LETTER KE</td>
</tr>
<tr>
<td>U+306F</td>
<td>KATAKANA LETTER GE</td>
</tr>
<tr>
<td>U+3073</td>
<td>KATAKANA LETTER KO</td>
</tr>
<tr>
<td>U+306B</td>
<td>KATAKANA LETTER GO</td>
</tr>
<tr>
<td>U+3065</td>
<td>KATAKANA LETTER SA</td>
</tr>
<tr>
<td>U+3066</td>
<td>KATAKANA LETTER SA</td>
</tr>
<tr>
<td>U+3067</td>
<td>KATAKANA LETTER SI</td>
</tr>
<tr>
<td>U+3068</td>
<td>KATAKANA LETTER ZI</td>
</tr>
<tr>
<td>U+3069</td>
<td>KATAKANA LETTER SU</td>
</tr>
<tr>
<td>U+306A</td>
<td>KATAKANA LETTER ZU</td>
</tr>
<tr>
<td>U+306B</td>
<td>KATAKANA LETTER SE</td>
</tr>
<tr>
<td>U+306C</td>
<td>KATAKANA LETTER ZU</td>
</tr>
<tr>
<td>U+306D</td>
<td>KATAKANA LETTER SO</td>
</tr>
<tr>
<td>U+306E</td>
<td>KATAKANA LETTER ZO</td>
</tr>
<tr>
<td>U+306F</td>
<td>KATAKANA LETTER TA</td>
</tr>
<tr>
<td>U+3070</td>
<td>KATAKANA LETTER DA</td>
</tr>
<tr>
<td>U+3071</td>
<td>KATAKANA LETTER TI</td>
</tr>
<tr>
<td>U+3072</td>
<td>KATAKANA LETTER DI</td>
</tr>
<tr>
<td>U+3073</td>
<td>KATAKANA LETTER SMALL TU</td>
</tr>
<tr>
<td>U+3074</td>
<td>KATAKANA LETTER TU</td>
</tr>
<tr>
<td>U+3075</td>
<td>KATAKANA LETTER DE</td>
</tr>
<tr>
<td>U+3076</td>
<td>KATAKANA LETTER TO</td>
</tr>
<tr>
<td>U+3077</td>
<td>KATAKANA LETTER DO</td>
</tr>
<tr>
<td>U+3078</td>
<td>KATAKANA LETTER NA</td>
</tr>
<tr>
<td>U+3079</td>
<td>KATAKANA LETTER NI</td>
</tr>
<tr>
<td>U+307A</td>
<td>KATAKANA LETTER NU</td>
</tr>
<tr>
<td>U+307B</td>
<td>KATAKANA LETTER NE</td>
</tr>
<tr>
<td>U+307C</td>
<td>KATAKANA LETTER NO</td>
</tr>
<tr>
<td>U+307D</td>
<td>KATAKANA LETTER HA</td>
</tr>
<tr>
<td>U+307E</td>
<td>KATAKANA LETTER BA</td>
</tr>
<tr>
<td>U+307F</td>
<td>KATAKANA LETTER PA</td>
</tr>
<tr>
<td>U+3081</td>
<td>KATAKANA LETTER HI</td>
</tr>
<tr>
<td>U+3082</td>
<td>KATAKANA LETTER BI</td>
</tr>
<tr>
<td>U+3083</td>
<td>KATAKANA LETTER PI</td>
</tr>
<tr>
<td>U+3084</td>
<td>KATAKANA LETTER RU</td>
</tr>
<tr>
<td>U+3085</td>
<td>KATAKANA LETTER BU</td>
</tr>
<tr>
<td>U+3086</td>
<td>KATAKANA LETTER FU</td>
</tr>
<tr>
<td>U+3087</td>
<td>KATAKANA LETTER RE</td>
</tr>
<tr>
<td>U+3088</td>
<td>KATAKANA LETTER BE</td>
</tr>
<tr>
<td>U+3089</td>
<td>KATAKANA LETTER PE</td>
</tr>
<tr>
<td>U+308A</td>
<td>KATAKANA LETTER RO</td>
</tr>
<tr>
<td>U+308B</td>
<td>KATAKANA LETTER BO</td>
</tr>
<tr>
<td>U+308C</td>
<td>KATAKANA LETTER PO</td>
</tr>
<tr>
<td>U+308D</td>
<td>KATAKANA LETTER MA</td>
</tr>
<tr>
<td>U+308E</td>
<td>KATAKANA LETTER MI</td>
</tr>
<tr>
<td>U+308F</td>
<td>KATAKANA LETTER MU</td>
</tr>
<tr>
<td>U+3090</td>
<td>KATAKANA LETTER ME</td>
</tr>
<tr>
<td>U+3091</td>
<td>KATAKANA LETTER MO</td>
</tr>
<tr>
<td>U+3092</td>
<td>KATAKANA LETTER YA</td>
</tr>
<tr>
<td>U+3093</td>
<td>KATAKANA LETTER YU</td>
</tr>
<tr>
<td>U+3094</td>
<td>KATAKANA LETTER YO</td>
</tr>
<tr>
<td>U+3095</td>
<td>KATAKANA LETTER RA</td>
</tr>
<tr>
<td>U+3096</td>
<td>KATAKANA LETTER RI</td>
</tr>
<tr>
<td>U+3097</td>
<td>KATAKANA LETTER RU</td>
</tr>
<tr>
<td>U+3098</td>
<td>KATAKANA LETTER RE</td>
</tr>
<tr>
<td>U+3099</td>
<td>KATAKANA LETTER RO</td>
</tr>
<tr>
<td>U+309A</td>
<td>KATAKANA LETTER WA</td>
</tr>
<tr>
<td>U+309B</td>
<td>KATAKANA LETTER WA</td>
</tr>
<tr>
<td>U+309C</td>
<td>KATAKANA LETTER WE</td>
</tr>
<tr>
<td>U+309D</td>
<td>KATAKANA LETTER WE</td>
</tr>
<tr>
<td>U+309E</td>
<td>KATAKANA LETTER MN</td>
</tr>
<tr>
<td>U+309F</td>
<td>KATAKANA LETTER MV</td>
</tr>
<tr>
<td>U+30A0</td>
<td>KATAKANA LETTER YA</td>
</tr>
<tr>
<td>U+30A1</td>
<td>KATAKANA LETTER YU</td>
</tr>
<tr>
<td>U+30A2</td>
<td>KATAKANA LETTER YO</td>
</tr>
<tr>
<td>U+30A3</td>
<td>KATAKANA LETTER RA</td>
</tr>
<tr>
<td>U+30A4</td>
<td>KATAKANA LETTER RI</td>
</tr>
<tr>
<td>U+30A5</td>
<td>KATAKANA LETTER RU</td>
</tr>
<tr>
<td>U+30A6</td>
<td>KATAKANA LETTER RE</td>
</tr>
<tr>
<td>U+30A7</td>
<td>KATAKANA LETTER RO</td>
</tr>
<tr>
<td>U+30A8</td>
<td>KATAKANA LETTER WA</td>
</tr>
<tr>
<td>U+30A9</td>
<td>KATAKANA LETTER WA</td>
</tr>
<tr>
<td>U+30AA</td>
<td>KATAKANA LETTER WE</td>
</tr>
<tr>
<td>U+30AB</td>
<td>KATAKANA LETTER WE</td>
</tr>
<tr>
<td>U+30AC</td>
<td>KATAKANA LETTER N</td>
</tr>
<tr>
<td>U+30AD</td>
<td>KATAKANA LETTER VU</td>
</tr>
<tr>
<td>U+30AE</td>
<td>KATAKANA LETTER YA</td>
</tr>
<tr>
<td>U+30AF</td>
<td>KATAKANA LETTER YU</td>
</tr>
<tr>
<td>U+30B0</td>
<td>KATAKANA LETTER YO</td>
</tr>
<tr>
<td>U+30B1</td>
<td>KATAKANA LETTER RA</td>
</tr>
<tr>
<td>U+30B2</td>
<td>KATAKANA LETTER RI</td>
</tr>
<tr>
<td>U+30B3</td>
<td>KATAKANA LETTER RU</td>
</tr>
<tr>
<td>U+30B4</td>
<td>KATAKANA LETTER RE</td>
</tr>
<tr>
<td>U+30B5</td>
<td>KATAKANA LETTER RO</td>
</tr>
<tr>
<td>U+30B6</td>
<td>KATAKANA LETTER WA</td>
</tr>
<tr>
<td>U+30B7</td>
<td>KATAKANA LETTER WA</td>
</tr>
<tr>
<td>U+30B8</td>
<td>KATAKANA LETTER WE</td>
</tr>
<tr>
<td>U+30B9</td>
<td>KATAKANA LETTER WE</td>
</tr>
<tr>
<td>U+30BA</td>
<td>KATAKANA LETTER VN</td>
</tr>
<tr>
<td>U+30BB</td>
<td>KATAKANA LETTER MV</td>
</tr>
<tr>
<td>U+30BC</td>
<td>KATAKANA LETTER YA</td>
</tr>
<tr>
<td>U+30BD</td>
<td>KATAKANA LETTER YU</td>
</tr>
<tr>
<td>U+30BE</td>
<td>KATAKANA LETTER YO</td>
</tr>
<tr>
<td>U+30BF</td>
<td>KATAKANA LETTER RA</td>
</tr>
</tbody>
</table>
Annex A
(informative)

Differences from the ISO/IEC 9945-2 standard

This Technical Report originated from the locale and charmap specifications in the ISO/IEC 9945-2 POSIX shell and utilities standard, and it intends to be backwards compatible, so that what is conformant to that standard should also be conformant to this Technical Report.

A number of enhancements have been made and a number of restrictions have been lifted in comparison to the POSIX standard:

A.1 Restrictions removed

1. Dependence on specific meaning of the character NUL as termination of a string (from the C standard) has been removed, to cater for other programming languages than C.

A.2 Enhancements

1. A description of a "repertoiremap" definition was added to facilitate descriptions of FDCC-sets without charmaps, and also to provide binding from a FDCC-set using one set of character names to charmaps using another naming set.

2. The specific POSIX locale has been replaced with the "i18n" FDCC-set, defined on the repertoire on ISO/IEC 10646.

3. Transliteration support has been added in the LC_CTYPE category.

4. Terminology has been aligned with ISO/IEC TR 11017, especially the POSIX term "locale" has been changed to "FDCC-set".

5. A date escape format "%F" has been added for ISO 8601 dates, and another date escape format "%f" has been added for weekday number with Monday being the first day of the week.

6. Added to LC_MONETARY to accommodate differences between local and international formats:
   - int_p_cs_precedes
   - int_p_sep_by_space
   - int_n_cs_precedes
   - int_n_sep_by_space

7. Section symbols have been added via the "section-symbol" keyword in the LC_COLLATE category.

8. The "order_start" keyword has got an optional "section-symbol" identifier

9. The keywords "reorder-section-after" and "reorder-section_end" have been introduced to reorder sections.

10. Symbolic ellipses (both decimal and hexadecimal) has been introduced as a notation.
11. The "print" LC_CTYPE class includes automatically all "graph" characters.

12. The <Uxxxx> and <Uxxxxxxxx> notations have been introduced as predefined symbolic character names, together with a number of symbolic character names derived from POSIX and the Internet.

13. New categories LC_IDENTIFICATION, LC_XLITERATE, LC_NAME, LC_ADDRESS, and LC_TELEPHONE, have been introduced.

14. The LC_CTYPE has got support for new classes, via the new keywords class and map, which corresponds to the C standard library functions iswctype() and towctrans() respectively.

15. The "digit" keyword now supports digits for multiple scripts.

16. The LC_MONETARY category provides support for multiple currencies, such as the native currency and the Euro in some European countries.

17. The LC_TIME has got a number of enhancements to cater for alternate calendars, and timezone information may be given.

18. The charmap specification has been enhanced to support ISO 2022.
Annex B
(informative)

Rationale

B.1 FDCC-set Rationale

The description of FDCC-sets is based on work performed in the UniForum Technical Committee Subcommittee on Internationalisation and POSIX. Wherever appropriate, keywords were taken from the C Standard or the ISO/IEC 9945-2:1993 POSIX standard. The C and POSIX term "locale" has been changed into the term "FDCC-set" from ISO/IEC TR 11017 to align with that specification.

The POSIX utility "localedef" compiles locale sources into object files. The "object" definitions need not be portable, as long as "source" definitions are. Strictly speaking, "source" definitions are portable only between applications using the same character set(s). Such "source" definitions can, if they use symbolic names only, easily be ported between systems using different code sets as long as the characters in the portable character set (ISO 646) have common values between the code sets; this is frequently the case in historical applications. Of course, this requires that the symbolic names used for characters outside the portable character set are identical between character sets.

To avoid confusion between an octal constant and a backreference, the octal, hexadecimal, and decimal constants must contain at least two digits. As single-digit constants are relatively rare, this should not impose any significant hardship. Each of the constants includes "two or more" digits to account for systems in which the byte size is larger than eight bits. For example, an ISO/IEC 10646 system that has defined 16-bit bytes may require six octal, four hexadecimal, and five decimal digits, for some coded characters.

As an international (ISO/IEC) Technical Report this Technical Report should follow the ISO/IEC guidelines, including the ISO/IEC TR 10176. This TR has a rule that characters outside the invariant part of ISO/IEC 646 should not be used in portable specifications. The backslash and the number-sign character are not in the invariant part. As far as general usage of these symbols, they are covered by the "grandfather clause" specifying previous practice in international standards and in the industry such as in specifications from The Open Group, but for newly defined interfaces, ISO has requested that specifications provide alternate representations, and this Technical Report then follows POSIX for backward compatibility. Consequently, while the default escape character remains the backslash, and the default comment character is the number-sign, applications are required to recognize alternative representations, identified in the applicable source text via the "escape_char" and "comment_char" keywords.

B.1.1 LC_IDENTIFICATION Rationale.

The LC_IDENTIFICATION category gives meta-information on the FDCC-set, such as who created it, and what is the level of conformance for each of the FDCC sets.

B.1.2 LC_CTYPE Rationale

The LC_CTYPE category primarily is used to define the encoding-independent aspects of a character set, such as character classification. In addition, certain encoding-dependent characteristics are also defined for an application via the LC_CTYPE category. This
Technical Report does not mandate that the encoding used in the FDCC-set is the same as the one used by the application, because an application may decide that it is advantageous to define a FDCC-set in a system-wide encoding rather than having multiple, logically identical FDCC-sets in different encodings, and to convert from the application encoding to the system-wide encoding on usage. Other applications could require encoding-dependent FDCC-sets. In either case, the LC_CTYPE attributes that are directly dependent on the encoding, such as "mb_cur_max" and the display width of characters, are not user-specifiable in a locale source, and are consequently not defined as keywords.

As the LC_CTYPE character classes are based on the C Standard character-class definition, the category does not support multicharacter elements. For instance, the German character <sharp-s> is traditionally classified as a lowercase letter. There is no corresponding uppercase letter; in proper capitalization of German text the <sharp-s> will be replaced by SS; i.e., by two characters. This kind of conversion is outside the scope of the "toupper" and "tolower" keywords.

The character classes "digit", "xdigit", "lower", "upper", and "space" have a set of automatically included characters. These only need to be specified if the character values (i.e. encoding) differs from the application default values. The definition of character class "digit" allows alternate digits (e.g., Hindi) to be specified here. The definition of character class "xdigit" requires that the characters included in character class "digit" are included here also, and allows for different symbols for the hexadecimal digits 10 through 15.

The "combining" and "combining-level3" classes are an IT-enablement of ISO/IEC 10646 definitions of combining characters. These can be used to check identifiers for consistence with the guidelines given in TR 10176 annex A.

**B.1.3 LC_COLLATE Rationale.**

The LC_COLLATE category governs the collation order in the FDCC-set, and may thus be useful for the processing of the ISO/IEC 14651 string ordering and comparison standard, the C Standard strxfrm() and strcoll() functions, as well as a number of ISO/IEC 9945-2:1993 POSIX utilities.

The rules governing collation depends to some extent on the use. At least five different levels of increasingly complex collation rules can be distinguished:

1. Byte/machine code order. This is the historical collation order in the UNIX system and many proprietary operating systems. Collation is here done character by character, without any regard to context. The primary virtue is that it usually is quite fast, and also completely deterministic; it works well when the native machine collation sequence matches the user expectations.
2. Character order. On this level, collation is also done character by character, without regard to context. The order between characters is, however, not determined by the code values, but on the user’s expectations of the correct order between characters. In addition, such a (simple) collation order can specify that certain characters collate equal (e.g., upper and lowercase letters).
3. String ordering. On this level, entire strings are compared based on relatively straightforward rules. At this level, several "passes" may be required to determine the order between two strings. Characters may be ignored in some passes, but not in others; the strings may be compared in different directions; and simple string substitutions may be made before strings are compared. This
level is best described as "dictionary" ordering; it is based on the spelling, not the pronunciation, or meaning, of the words.

(4) Text search ordering. This is a further refinement of the previous level, best described as "telephone book ordering"; some common homonyms (words spelled differently but with same pronunciation) are collated together; numbers are collated as if spelled with words, and so on.

(5) Semantic level ordering. Words and strings are collated based on their meaning; entire words (such as "the") are eliminated, the ordering is not deterministic. This may requires special software, and is highly dependent on the intended use.

While the historical collation order formally is at level 1, for the English language it corresponds roughly to elements at level 2. The user expects to see the output from the "ls" utility sorted very much as it would be in a dictionary. While telephone book ordering would be an optimal goal for standard collation, this was ruled out as the order would be language dependent. Furthermore, a requirement was that the order must be determined solely from the text string and the collation rules; no external information (e.g., "pronunciation dictionaries") could be required.

As a result, the goal for the collation support is at level 3. This also matches the requirements for the Canadian collation order standard, as well as other, known collation requirements for alphabetic scripts. It specifically rules out collation based on pronunciation rules, or based on semantic analysis of the text. 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, The Open Group, The Unicode Consortium Inc. and ISO, and it meets the requirements for level 3, and has been verified to produce the correct result with examples based on Canadian and Danish collation order.

The directives that can be specified in an operand to the order_start keyword are based on the requirements specified in several proposed standards and in customary use. The following is a rephrasing of rules defined for "lexical ordering in English and French" by the Canadian Standards Association (text is brackets is rephrased):

(1) Once special characters (punctuation) have been removed from original strings, the ordering is determined by scanning forward (left to right) [disregarding case and diacriticals].

(2) In case of equivalence, special characters are once again removed from original strings and the ordering is determined scanning backward (starting from the rightmost character of the string and back), character by character, (disregarding case but considering diacriticals).

(3) In case of repeated equivalence, special characters are removed again from original strings and the ordering is determined scanning forward, character by character, (considering both case and diacriticals).

(4) If there is still an ordering equivalence after rules (1) through (3) have been applied, then only special characters and the position they occupy in the string are considered to determine ordering. The string that has a special character in the lowest position comes first. If two strings have a special character in the same position, the character [with the lowest collation value] comes first. In case of equality, the other special characters are considered until there is a difference or all special characters have been exhausted.
It is estimated that the Technical Report covers the mechanisms to specify data to cover the requirements for all European languages, and Cyrillic and Middle Eastern scripts.

The Far East (particularly Japanese/Chinese) collations are often based on contextual information. In Japan, collations of strings containing CJK characters (ideograms) are often done considering some related information such as pronunciation, which needs a bulk dictionary (and some common sense). Such collation, in general, falls outside the desired goal of this Technical Report, and this Technical Report can support only a restricted of collations used in Japan. There are, however, several other collation rules (stroke/radical, or "most common pronunciation") which can be supported with the mechanism described here. Previous drafts contained a substitute statement, which performed a regular expression style replacement before string compares. It has been withdrawn based on balloter objections that it was not required for the types of ordering this Technical Report is aimed at.

The character (and collating element) order is defined by the order in which characters and elements are specified between the order_start and order_end keywords. This character order is used in range expressions in regular expressions. Weights assigned to the characters and elements define the collation sequence; in the absence of weights, the character order is also the collation sequence.

The position keyword was introduced to provide the capability to consider, in a compare, the relative position of non-IGNOREd characters. As an example, consider the two strings "o-ring" and "or-ing". Assuming the hyphen is IGNOREd on the first pass, the two strings will compare equal, and the position of the hyphen is immaterial. On second pass, all characters except the hyphen are IGNOREd, and in the normal case the two strings would again compare equal. By taking position into account, the first collates before the second.

This Technical Report adds a number of facilities over the ISO/IEC 9945:1993 POSIX standard, especially in the support for the ISO/IEC 10646 UCS character set. These extended facilities are in alignment with the ISO/IEC 14651 sorting standard. In addition to the facilities provided in ISO/IEC 14651, this specification contains mechanisms to put data into a FDCC-set environment, and has added facilities to sort sections differently, has facilities to reuse FDCC-sets in different notations via the "equivalence-symbol" keyword and tables.

B.1.3.1 "reorder-after" rationale

Much work has been done on FDCC-sets, making them quite general. The ISO/IEC 9945-2:1993 POSIX standard introduced a "copy" command for all categories of the POSIX locale. This is useful for many purposes and it ensures that two FDCC-sets are equivalent for this category. A further step in building on previous FDCC-set work is defined in this Technical Report.

Collating sequences often vary a bit from country to country, and from language to language, but generally much of the collating sequence is the same. For example the Danish sequence is for the most part the same as the German or English collation, but for about a dozen letters it differs. The same can be said for Swedish or Hungarian: generally the Latin collating sequence is the same, but a few characters are different.

This Technical Report defines a FDCC-set defined on the character repertoire of the ISO/IEC 10646 standard, in a character set independent way. The intention is that some of
the information from this FDCC-set will be acceptable in many cultures, and that it can serve as the basis for modifications in other cultures, to obtain a culturally acceptable specification. Using the "reorder-after" construct will also help improve the overview of what the changes really are for implementers and other users.

An example of the use of the "reorder-after" construct is the following. A default international ordering for the Latin alphabet may be adequate for Danish, with the exception of the collation rules for the letters Ü, ü, Æ, æ, Â, å, Ø, ø, Ö, ö, Å and å. By applying the "reorder-after" construct, the Danish specification can be made more easily by copying and reordering the existing international specification, rather than specifying collation parameters for all Latin letters (with or without diacritics). There is no obligation for Denmark to take this approach, but the "reorder-after" construct provides the mechanism for doing so if it is deemed desirable.

B.1.3.2 awk script for "reorder-after" construct

A script has been written in the "awk" language defined in the POSIX standard ISO/IEC 9945-2 to implement the "reorder-after" construct. It functions as follows: It reads all of the FDCC-set and if in the LC_COLLATE category, it processes the line, else it just outputs the line. For the LC_COLLATE category it reads the lines and puts it into a double linked list of strings identified by a line number; at the end of the LC_COLLATE category all the lines are output. If the line is a "copy" keyword and it reads the file referenced, extracting the LC_COLLATE section of the file in to the list of strings. If the line is a "reorder-after" keyword, it sets a pointer to be the line number of the symbol to of the "reorder-after" keyword. If the line is part of the "reorder-after" specification, it is entered into the double linked list at this point, and the previous entry in the double linked list for the <collation-element> is removed from the list. A "reorder-end" keyword terminates the reordering.

BEGIN { comment = "%"; back[0]= follow[0] = 0; }
/LC_COLLATE/ { coll=1 }
/END LC_COLLATE/ { coll=0; for (lnr= 1; lnr; lnr= follow[lnr]) print cont[lnr] }

{ if (coll == 0) print $0 ;
 else ( if ($1 == "copy") {
 file = $2
 while (getline < file )
 if ( $1 == "LC_COLLATE" ) copy_lc = 1
 else if ( $1 == "END" && $2 == "LC_COLLATE" ) copy_lc =0
 else if (copy_lc) {
 lnr++
 follow[lnr-1] = lnr; back [ lnr ] = lnr-1
 cont[lnr] = $0; symb[ $1 ] = lnr
 }
 close (file )
 }
 else if ($1 == "reorder-after") { ra=1 ; after = symb [ $2 ] }
 else if ($1 == "reorder-end") ra = 0
 else {
 lnr++
 if (ra) follow [ lnr ] = follow [ after ]
 if (ra) back [ follow [ after ] ] = lnr
 follow[after] = lnr; back [ lnr ] = after
 cont[lnr] = $0
 if ( ra && $1 != comment && $1 != """) {
 old = symb [ $1 ];
 follow [ back [ old ] ] = follow [ old ];
 back [ follow [ old ] ] = back [ old ];
 symb[ $1 ] = lnr;
 }
 after = lnr
 }
}
B.1.3.3 Sample FDCC-set specification for Danish

```
escape_char /
comment_char %
repertoiremap "i18nrep"
charset "ISO_8859-1:1987"
% Distribution and use is free, also
% for commercial purposes.

LC_VERSION
  title         "Danish language FDCC-set for Denmark"
  source        "Danish Standards Association"
  address       "Kollegievej 6, DK-2920 Charlottenlund, Danmark"
  contact       "Keld Simonsen"
  email         "Keld.Simonsen@dkuug.dk"
  phone         "+45 - 3996-6101"
  fax           "+45 - 3996-6202"
  language      "da"
  territory     "DK"
  revision      "4.2"
  date          "1997-12-22"

  category i18n:2000;LC_IDENTIFICATION
  category i18n:2000;LC_CTYPE
  category i18n:2000;LC_COLLATE
  category i18n:2000;LC_TIME
  category posix:1993;LC_NUMERIC
  category i18n:2000;LC_MONETARY
  category posix:1993;LC_MESSAGES
  category i18n:2000;LC_XLITERATE
  category i18n:2000;LC_NAME
  category i18n:2000;LC_ADDRESS
  category i18n:2000;LC_TELEPHONE

END LC_VERSION

LC_CTYPE
  copy "i18n"
END LC_CTYPE

LC_COLLATE
% The ordering algorithm is in accordance
% with Danish Standard DS 377 (1980)
% and the Danish Orthography Dictionary
% (Retskrivningsordbogen, 2. udgave, 1996).
% It is also in accordance with
% Greenlandic orthography.

  collating-element <A-A> from "<A><A>"
  collating-element <A-a> from "<A><a>"
  collating-element <a-A> from "<a><A>"
  collating-element <a-a> from "<a><a>"
  collating-symbol <SPECIAL>
  copy i18n
  reorder-after <CAPITAL>
  <CAPITAL>
  <CAPITAL-SMALL>
  <SMALL-CAPITAL>
  <SMALL>
  reorder-after <q8>
  <kk>  <Q>;<SPECIAL>;<SMALL>;IGNORE
  reorder-after <t8>  
  <TH>  "<T><H>";"<TH><TH>";"<CAPITAL><CAPITAL>";IGNORE
  <th>  "<T><H>";"<TH><TH>";"<SMALL><SMALL>";IGNORE
  reorder-after <y8>
% <U:> and <U" are treated as <Y> in Danish
  <U:>  <Y>;<U>;<CAPITAL>;IGNORE
  <u>  <Y>;<U>;<SMALL>;IGNORE
  <U">  <Y>;<U";<CAPITAL>;IGNORE
  <u">  <Y>;<U";<SMALL>;IGNORE
```
reorder-after <z8>
% <AE> is a separate letter in Danish
<AE>  <AE>;<NONE>;<CAPITAL>;IGNORE
<ae>  <AE>;<NONE>;<SMALL>;IGNORE
<AE'>  <AE>;<ACUTE>;<CAPITAL>;IGNORE
<ae'>  <AE>;<ACUTE>;<SMALL>;IGNORE
<A3>  <AE>;<MACRON>;<CAPITAL>;IGNORE
<a3>  <AE>;<MACRON>;<SMALL>;IGNORE
<A:>  <AE>;<SPECIAL>;<CAPITAL>;IGNORE
<a:>  <AE>;<SPECIAL>;<SMALL>;IGNORE
% <O//> is a separate letter in Danish
<O//>  <O//>;<NONE>;<CAPITAL>;IGNORE
<o//>  <O//>;<NONE>;<SMALL>;IGNORE
<O//'>  <O//>;<ACUTE>;<CAPITAL>;IGNORE
<o//'>  <O//>;<ACUTE>;<SMALL>;IGNORE
<O:>  <O//>;<DIAERESIS>;<CAPITAL>;IGNORE
&o>  <O//>;<DIAERESIS>;<SMALL>;IGNORE
% <AA> is a separate letter in Danish
<AA>  <AA>;<NONE>;<CAPITAL>;IGNORE
<aa>  <AA>;<NONE>;<SMALL>;IGNORE
<A-A>  <AA>;<A-A>;<CAPITAL>;IGNORE
<AA>  <AA>;<A-A>;<CAPITAL>;IGNORE
<AA>  <AA>;<A-A>;<CAPITAL>;IGNORE
<aa>  <AA>;<A-A>;<CAPITAL>;IGNORE
reorder-end
END LC_COLLATE

LC_MONETARY
int_curr_symbol  "<$K<$K$SP"
currency_symbol  "<k><r>
mon_decimal_point  "<,>
mon_thousands_sep  ".<.>
mon_grouping  3;3
positive_sign  "
negative_sign  "<->
int_frac_digits  2
frac_digits  2
p_cs_precedes  1
p_sep_by_space  2
n_cs_precedes  1
n_sep_by_space  2
p_sign_posn  4
n_sign_posn  4
END LC_MONETARY

LC_NUMERIC
decimal_point  "<,>
thousands_sep  ".<.>
grouping  3;3
END LC_NUMERIC

LC_TIME
abday       "<m><a><n>";
     "<t><I><r>";"<o><n><s>";/
     "<t><o><r>";"<f><e><r>";/
     "<l><o><r>";"<o><n><s>";
     "<t><o><r>";"<f><e><r>";/
     "<l><o><r>";"<o><n><s>";
day        "<m><a><n><d><a><g>";
     "<t><I><r><s><d><a><g>";/
     "<t><o><r><s><d><a><g>";/
     "<l><o><r><s><d><a><g>";
     "<t><o><r><s><d><a><g>";/
     "<l><o><r><s><d><a><g>";
week       7;19971201;4
abmon      "<j><a><n>";"<f><e><b>";/
     "<m><a><r>";"<a><p><r>";/
     "<m><a><r>";"<j><u><n>";
B.1.4 LC_MONETARY Rationale.

The currency symbol does not appear in LC_MONETARY because it is not defined in the C Standard’s C locale. The C Standard limits the size of decimal points and thousands delimiters to single-byte values. In FDCC-sets based on multibyte coded character sets this
cannot be enforced, obviously; this Technical Report does not prohibit such characters, but makes the behaviour unspecified (in the text "In contexts where other standards . . . ").

The grouping specification is based on, but not identical to, the C Standard. The "-1" signals that no further grouping is performed, the equivalent of (CHAR_MAX) in the C Standard).

The FDCC-set definition is an extension of the C Standard `localeconv()` specification. In particular, rules on how currency_symbol is treated are extended to also cover `int_curr_symbol`, and `p_set_by_space` and `n_sep_by_space` have been augmented with the value 2, which places a space between the sign and the symbol (if they are adjacent; otherwise it should be treated as a 0). The following table shows the result of various combinations:

<table>
<thead>
<tr>
<th>p_cs_precedes</th>
<th>p_sign_posn</th>
<th>p_sep_by_space</th>
<th>Formatted Value</th>
<th>C String</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>($ 1.25)</td>
<td>($ 1.25)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>+ $1.25</td>
<td>+$ 1.25</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>$1.25 +</td>
<td>$ 1.25+</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>+ $1.25</td>
<td>+$ 1.25</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>$ +1.25</td>
<td>$+ 1.25</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>(1.25 $)</td>
<td>(1.25 $)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>+1.25 $</td>
<td>+1.25 $</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>1.25 $ +</td>
<td>1.25 $+</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>1.25+ $</td>
<td>1.25+$</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>1.25$ +</td>
<td>1.25+$</td>
</tr>
</tbody>
</table>

The following is an example of the interpretation of the `mon_grouping` keyword. Assuming that the value to be formatted is 123456789 and the `mon_thousands_sep` is "'", then the following table shows the result. The third column shows the equivalent C Standard string that would be used to accommodate this grouping. It is the responsibility of the utility to perform mappings of the formats in this clause to those used by language bindings such as the C Standard.

<table>
<thead>
<tr>
<th>Mon_grouping</th>
<th>Formatted Value</th>
<th>C String</th>
</tr>
</thead>
<tbody>
<tr>
<td>3;-1</td>
<td>123456'789</td>
<td>&quot;\3\177&quot;</td>
</tr>
<tr>
<td>3</td>
<td>123'456'789</td>
<td>&quot;\3&quot;</td>
</tr>
<tr>
<td>3;2;-1</td>
<td>1234'56'789</td>
<td>&quot;\3\2\177&quot;</td>
</tr>
<tr>
<td>3;2</td>
<td>12'34'56'789</td>
<td>&quot;\3\2&quot;</td>
</tr>
<tr>
<td>-1</td>
<td>123456789</td>
<td>&quot;177&quot;</td>
</tr>
</tbody>
</table>

In these examples, the octal value of (CHAR_MAX) is 177.

The multiple currency support is specified such that a FDCC-set can be used without change during the transition period in a static environment. For example in the case of the Euro currency as being employed in a number of European countries, there is no need to change the FDCC-set when shifting from one currency to two concurrent currencies; and there is no need to change FDCC-set, when changing to the Euro as the only currency. Also the same application call can be made to be valid for countries with a single currency...
and countries with dual currencies. The specifications can also be used without change of the FDCC-set on an installation, when converting from one national currency to another, for example when removing some zeroes to form a new currency.

The following example illustrates the support for multiple currencies; the example is for the Euro in Germany:

```plaintext
LC_MONETARY
valid_from              ";";        "19990101"
valid_to                "20020630";      ""
conversion_rate         1/1;            195/100
int_curr_symbol         "<D><E><M><SP>";  "<E><U><R><SP>",
currency_symbol         "<D><M>;        "<E><U><R"
mon_decimal_point       ";,<>",
mon_thousands_sep       ";,
mon_grouping            3;3
positive_sign           ";
negative_sign           "<->"
int_frac_digits         2;            2
frac_digits             2;            2
p_cs_precedes           1;            1
p_sep_by_space          2;            2
n_cs_precedes           1;            1
n_sep_by_space          2;            2
p_sign_posn             4;            4
n_sign_posn             4;            4
END LC_MONETARY
```

B.1.5 LC_NUMERIC Rationale.

See the rationale for LC_MONETARY (B.1.3) for a description of the behaviour of grouping.

B.1.6 LC_TIME Rationale.

The LC_TIME descriptions of abday, day, and abmon imply a Gregorian style calendar (7-day weeks, 12-month years, leap years, etc.). Other calendars can be supported, for example calendars with a fixed week length.

In some FDCC-sets the field descriptors for weekday and month names will be given with an initial small letter. Programs using these fields may need to adjust the capitalization if the output is going to be used at the beginning of a sentence.

The field descriptors corresponding to the optional keywords consist of a modifier followed by a traditional field descriptor (for instance %Ex). If the optional keywords are not supported by the application or are unspecified for the current FDCC-set, these field descriptors are treated as the traditional field descriptor. For instance, assume the following keywords:

```
alt_digits "0th";"1st";"2nd";"3rd";"4th";"5th";"6th";"7th";"8th";"9th";"10th"
d_fmt "The %Od day of %B in %Y"
```

On 1776-07-04, the %x field descriptor would result in "The 4th day of July in 1776," while 1789-07-14 would come out as "The 14 day of July in 1789." It can be noted that the above example is for illustrative purposes only; the %o modifier is primarily intended to provide for Kanji or Hindi digits in date formats. While it is clear that an alternate year format is required, there is no consensus on the format or the requirements. As a result,
while these keywords are reserved, the details are left unspecified. It is expected that National Standards Bodies will provide specifications.

B.1.7 LC_MESSAGES Rationale.

The LC_MESSAGES category is described in clause 4 as affecting the language used by utilities for their output. The mechanism used by the application to accomplish this, other than the responses shown here in the FDCC-set definition, is not specified by this version of this Technical Report. The ISO internationalization working group ISO/IEC JTC1/SC22/WG20 is developing an interface that would allow applications (and, presumably some of the standard utilities) to access messages from various message catalogs, tailored to a user's LC_MESSAGES value.

B.1.8 LC_XLITERATE Rationale.

Transliteration is often language dependent, transliterating one specific language to another specific language. For example transliteration from Russian to English, and from Serbian to German would normally be quite different, although the same repertoire of characters would be transliterated. Even transliteration of two languages using the same script into one language (for example from Russian to Danish and from Serbian to Danish), or transliteration of the same language (for example Russian into English or German) may be different. The language to be transliterated to is identified with the FDCC-set, which may also be used to identify a specific language to be transliterated from. Transliteration may also be to a specific repertoire of characters, determined for example by limitations of displaying equipment, or what the user can intelligibly read. The capabilities here allows for multiple fallback, so that the specification can be valid for all target character repertoires, eliminating the need for specific data for each target repertoire.

B.1.9 LC_NAME Rationale.

The LC_NAME category gives information to prepare a text for addressing a person, for example as a part of a postal address on an envelope, or as a salutating line in a letter. The information is intended to be given to an API that has the various naming information as parameters and yields a formatted string as the return value.

The "profession" entry is intended for either the general profession of the person in question, or the job title, for use in letters or as part of the address on an envelope.

B.1.10 LC_ADDRESS Rationale.

The LC_ADDRESS category gives information to prepare a text for writing an address, for example as a part of a postal address on an envelope. The information is intended to be given to an API that has the various address information as parameters and yields a formatted string as the return value.

B.1.11 LC_TELEPHONE Rationale.

The LC_TELEPHONE category gives information to prepare a text for writing a telephone number. The information is intended to be given to an API that has the various information on a telephone number as parameters and yields a formatted string as the return value. Both an international and a domestic formatting possibility is available.
B.2 Character Set Rationale.

This Technical Report poses no requirement that multiple character sets or code sets be supported, leaving this as a marketing differentiation for implementors. Although multiple charmaps are supported, it is the responsibility of the application to provide the file(s); if only one is provided, only that one will be accessible.

The character set description text provides the capability to describe character set attributes (such as collation order or character classes) independent of character set encoding, and using only the characters in the portable character set. This makes it possible to create "generic" FDCC-set source texts for all code sets that share the portable character set (such as the ISO/IEC 8859 family or IBM Extended ASCII).

Applications are free to describe more than one code set in a character set description text. For example, if an application defines ISO/IEC 8859-1 as the primary code set, and ISO/IEC 8859-2 as an alternate set, with each character from the alternate code set preceded in data by a shift code, a character set description text could contain a complete description of the primary set and those characters from the secondary that are not identical, the encoding of the latter including the shift code.

Applications are free to choose their own symbolic names, as long as the names identified by this Technical Report are also defined; this provides support for already existing "character names".

The charmap was introduced to resolve problems with the portability of, especially, FDCC-set sources. While the portable character set (in Table 1) is a constant across all FDCC-sets for a particular application, this is not true for the extended character set. However, the particular coded character set used for an application does not necessarily imply different characteristics or collation: on the contrary, these attributes should in many cases be identical, regardless of codeset. The charmap provides the capability to define a common FDCC-set definition for multiple codesets (the same FDCC-set source can be used for codesets with different extended characters; the ability in the charmap to define "empty" names allows for characters missing in certain codesets).

In addition, some implementors have expressed an interest in using the charmap to define certain other characteristics of codesets, such as the <mb_cur_max> value for the particular codeset. (Note that <mb_cur_max> has to be equal to or lower than the C Standard [MB_LEN_MAX], which is the application limit). Such extensions are not described here; but may be added in a later revision of this Technical Report.

The <escape_char> declaration was added at the request of the international community to ease the creation of portable charmaps on terminals not implementing the default backslash escape. (This approach was adopted because this is a new interface invented by ISO/IEC 9945-2:1993 POSIX. Historical interfaces, such as the shell command language and awk, have not been modified to accommodate this type of terminal.)

The octal number notation was selected to match those of POSIX "awk" and "tr" utilities and is consistent with that used by the POSIX localedef utility.

The charmap capability implements a facility available at some X/Open compatible applications. Its prime virtue is to support "generic" collation sequence source definitions.
An implementor or an applications developer can produce a template definition that can be used to produce several codeset-dependent "compiled" FDCC-set definitions. The facility also removes any dependency in many source definitions on characters outside the character set defined in this clause.

The charmap allows specification of more than one encoding of a character. This allows for encodings that can encode items in more than one way. For example, an item can be encoded once as a fully composed character and again as a base character plus combining character. This would allow either representation to be recognized. As only the first occurrence of the character may be output, this technique could be used to normalize a character stream.

The ISO 2022 support introduced gives the possibility to refer other definitions via charmaps, so the full encoding does not have to be replicated. It supports shifting with G0, G1, G2 and G3 sets, and also general shifting of coded character sets via escape sequences.

B.3 Repertoiremap Rationale.

The repertoiremap was introduced to make FDCC-sets independent of the availability of charmaps. With the repertoiremap it is possible to use a FDCC-set encoded with one set of symbolic character names, together with charmaps with other symbolic character naming schemes, provided there are repertoiremaps available for both naming schemes.

Repertoiremaps are also useful to describe repertoires of characters, to be used for example for transliteration.
Annex C
(informative)

BNF Grammar

C.1 BNF Syntax Rules

The syntax used here is near to ISO/IEC 14977, but "_" is allowed in identifiers, and comma is not used as concatenator, as the items are just concatenated.

Definitions between <angle brackets> make use of terms not defined in this BNF syntax, and assume general English usage.

Other conventions:
* means 0 or more repetitions of a token.
+ means one or more repetitions of a token
Brackets [ ] indicate optional occurrence of a token.
Comments start with a % on a separate line.

There may be more specifications in the normative text that describes restrictions on the grammar.

C.2 Grammar for FDCC-sets

% The following is the overall FDCC-set grammar
FDCC_set_definition = [ global_statement* ] category+ ;
global_statement = ‘escape_char’ SP char_symbol EOL
| ‘comment_char’ SP char_symbol end_of_line
| ‘repertoiremap’ SP quoted_string EOL
| ‘charmap’ SP quoted_string EOL ;
category = lc_identification | lc CType | lc_collate
| lc_monetary | lc_numeric | lc_time
| lc_messages | lc_xliterate | lc_telephone
| lc_name | lc_address ;

% The following is the LC_IDENTIFICATION category grammar
lc_ident = ident_head ident_keyword* ident_tail
| ident_head copy_FDCC_set ident_tail ;
ident_head = ‘LC_IDENTIFICATION’ EOL ;
ident_keyword = ident_keyword_string SP quoted_string EOL
| ident_keyword_string = ‘title’ | ‘source’ | ‘address’ | ‘contact’
| ‘email’ | ‘tel’ | ‘fax’ | ‘language’
| ‘territory’ | ‘audience’ | ‘application’
| ‘abbreviation’ | ‘revision’ | ‘date’ ;
ident_tail = ‘END’ SP ‘LC_IDENTIFICATION’ EOL ;

% The following is the LC_CTYPE category grammar
lc CType = ctype_head ctype_keyword* ctype_tail
| ctype_head copy_FDCC_set ctype_tail ;
ctype_head = ‘LC_CTYPE’ EOL ;
ctype_keyword = charclass_keyword SP charclass_list EOL
| charconv_keyword SP charconv_list EOL
| ‘width’ SP width_list EOL;
| ‘class’ charclass_name semicolon ;
charclass_name = ’”’ combining”’ | ’”’ combining_level3”’
| ’”,’ identifier ’”’ ;
charclass_list = charclass_list semicolon char_symbol
| charclass_list semicolon ctype_abs_ellipsis
| charclass_list semicolon ctype_symbolic_ellipses charsymbol
| char_symbol;

width_list = charclass_list ':=' number
| width_list semicolon width_list;

charconv_keyword = 'toupper' | 'tolower'
| 'map' | identifier semicolon charconv_list
| charconv_entry;

charconv_entry = '(' char_symbol comma char_symbol ')' ;

cctype_symbolic_ellipses = '..' | '....';

cctype_abs_ellipsis = '...';

cctype_tail = 'END' SP 'LC_TYPE' EOL;

% The following is the LC_COLLATE category grammar

lc_collate = collate_head collate_keywords collate_tail;

collate_head = 'LC_COLLATE' EOL;

collate_keywords = opt_statement* order_statements | delta;

opt_statement = 'collating-symbol' SP collsymbol_list EOL
| 'collating-element' SP collelement SP 'from' SP collem_string EOL
| 'section-symbol' space+ section_symbol EOL
| 'col_weight_max' SP number EOL
| 'symbol-equivalence' SP collsymbol SP collsymbol EOL
| collation_statement;

collem_string = "" char_symbol "";

order_statements = order_start collation_order order_end;

order_start = 'order_start' SP order_params EOL;

order_params = [section_symbol] [semicolon order_opts];

order_opts = order_opt [ semicolon order_opt ];

order_opt = opt_word [ comma opt_word ];

opt_word = 'forward' | 'backward' | 'position';

section = 'section' SP section_symbol [ SP collsymbol_list ] EOL;

collation_order = ( order_start | section | collation_statement )*;

collation_statement = collating_element [ SP weight_list ] EOL;

collsymbol_list = collsymbol_element [ semicolon collsymbol_element ];

collsymbol_element = collsymbol |
collsymbol SP ellipses SP collsymbol;

collating_element = collsymbol SP ellipses SP collsymbol;

weight_list = weight_symbol [ semicolon weight_symbol ];

weight_symbol = <empty>
| char_symbol collsymbol
| "" elem_list "" |
| "" symb_list "" | 'IGNORE';

ellipses = .. | ....;

order_end = 'order_end' EOL;

delta = opt_statement*
| 'copy' SP FDCC_set_name EOL
| opt_statement*
| reordering_statement*;

reordering_statement = reorder_after_block
| reorder_section_after_1
| reorder_section_after_2 section_statement*

reorder_after_block = reorder_after (collation_order | reorder_after)* reorder_end;

reorder_after = 'reorder-after' SP sectionsymbol EOL;

reorder_end = 'reorder-end' EOL;

reorder_section_block = reorder_section_after_2 section_statement* reorder_section_end;

section_statement = section_symbol SP order_opts EOL;

reorder_section_after_1 = 'reorder-section-after' SP sectionsymbol SP
The following is the LC_MESSAGES category grammar:

```
lc_messages = messages_head messages_keyword* messages_tail
```

```
messages_head = 'LC_MESSAGES' EOL ;
```

```
messages_keyword = 'yesexpr' SP '"' extended_reg_expr '"' EOL |
                  'yesexpr' SP '"' extended_reg_expr '"' EOL ;
```

```
messages_tail = 'END' SP 'LC_MESSAGES' EOL ;
```

The following is the LC_MONETARY category grammar:

```
lc_monetary = monetary_head monetary_keyword* monetary_tail
```

```
monetary_head = 'LC_MONETARY' EOL ;
```

```
monetary_keyword = mon_keyword_string SP quoted_string EOL |
                  mon_keyword_strings SP mon_string_list EOL |
                  mon_keyword_char SP mon_number_list EOL |
                  mon_keyword_date SP mon_date_list EOL |
                  'conversion_rate' SP mon_conv_list EOL |
                  'mon_grouping' SP mon_group_list EOL ;
```

```
mon_keyword_string = 'mon_decimal_point' | 'mon_thousands_sep' |
                    'positive_sign' | 'negative_sign' ;
```

```
mon_keyword_strings = 'int_curr_symbol' | 'currency_symbol' ;
```

```
mon_keyword_char = 'int_frac_digits' | 'frac_digits' |
                  'p_cs_precedes' | 'p_sep_by_space' |
                  'n_cs_precedes' | 'n_sep_by_space' |
                  'int_p_cs_precedes' | 'int_p_sep_by_space' |
                  'int_n_cs_precedes' | 'int_n_sep_by_space' |
                  'p_sign_posn' | 'n_sign_posn' ;
```

```
mon_keyword_date = 'valid_from' | 'valid_to' ;
```

```
mon_date_list = mon_date | mon_date_list semicolon mon_date ;
```

```
mon_date = '"' 8 * digit '"' ;
```

```
mon_group_list = number | mon_group_list semicolon number ;
```

```
mon_string_list = quoted_string [ semicolon quoted_string]* ;
```

```
mon_number_list = mon_number | mon_number_list semicolon mon_number ;
```

```
mon_number = number | -1 ;
```

```
mon_conv_list = mon_pair | mon_conv_list semicolon mon_pair ;
```

```
mon_pair = number spaces* '/' spaces* number ;
```

```
monetary_tail = 'END' SP 'LC_MONETARY' EOL ;
```

The following is the LC_NUMERIC category grammar:

```
lc_numeric = numeric_head numeric_keyword* numeric_tail
```

```
numeric_head = 'LC_NUMERIC' EOL ;
```

```
numeric_keyword = num_keyword_string SP quoted_string EOL |
                  num_keyword_grouping SP num_group_list EOL ;
```

```
um_keyword_string = 'decimal_point' | 'thousands_sep' ;
```

```
um_keyword_grouping = 'grouping' ;
```

```
um_group_list = number |
                  num_group_list semicolon number ;
```

```
numeric_tail = 'END' SP 'LC_NUMERIC' EOL ;
```

The following is the LC_TIME category grammar:

```
lc_time = time_head time_keyword* time_tail
```

```
time_head = 'LC_TIME' EOL ;
```

```
time_keyword = time_keyword_name SP time_list EOL |
               time_keyword_fmt SP quoted_string EOL |
               time_keyword_opt SP time_list EOL |
               'week' SP number semicolon mon_date semicolon number EOL |
               'time_keyword_num' SP number EOL |
               'timezone' SP time_list EOL ;
```

```
time_keyword_name = 'abday' | 'day' | 'abmon' | 'mon' | 'am_pm' ;
```

```
time_keyword_fmt = 'd_t_fmt' | 'd_fmt' | 't_fmt' | 't_fmt_ampm' ;
```

```
time_keyword_opt = 'era' | 'era_year' | 'era_d_fmt' | 'alt_digits'
```
ISO/IEC TR 14652:2002(E)

time_keyword_week = 'week' ;
time_keyword_num = 'first_weekday' | 'first_workday' |
'cal_direction' ;
time_list = time_list semicolon quoted_string |
quoted_string ;
time_tail = 'END' SP 'LC_TIME' EOL ;

% The following is the LC_XLITERATE category grammar
lc_xliterate = translit_head [translit_include] |
[default_missing] translit_statement* |
translit_tail | translit_head copy_FDCC_set |
translit_tail ;
translit_head = 'LC_XLITERATE' EOL ;
translit_include = 'include' SP FDCC_set_name semicolon |
quoted_nonempty_string EOL ;
default_missing = 'default_missing' SP quoted_string EOL ;
translit_statement = char_or_string SP char_or_string |
[semicolon] char_or_string]* EOL ;
translit_tail = 'END' SP 'LC_XLITERATE' EOL ;

% The following is the LC_NAME category grammar
lc_name = name_head name_keyword* name_tail |
name_head SP name_keyword_string SP quoted_string EOL ;
name_keyword = name_keyword_string SP quoted_string EOL ;
name_keyword_string = 'name_fmt' | 'name_gen' | 'name_mr' |
'name_mrs' | 'name_ms' | 'name_miss' |
'name_ms' ;
name_tail = 'END' SP 'LC_NAME' EOL ;

% The following is the LC_ADDRESS category grammar
lc_address = address_head address_keyword* address_tail |
address_head SP address_keyword_string SP quoted_string EOL ;
address_keyword = address_keyword_string SP quoted_string EOL ;
address_keyword_string = 'postal_fmt' | 'country_name' |
'country_post' | 'lang_name' | 'lang_ab2' |
'lang_ab3_term' | 'lang_ab3_lib' ;
address_tail = 'END' SP 'LC_ADDRESS' EOL ;

% The following is the LC_TELEPHONE category grammar
lc_tel = tel_head tel_keyword* tel_tail |
tel_head SP tel_keyword_string SP quoted_string EOL ;
tel_keyword = tel_keyword_string SP quoted_string EOL ;
tel_keyword_string = 'tel_int_fmt' | 'tel_dom_fmt' |
'int_select' |
'int_prefix' ;
tel_tail = 'END' SP 'LC_TELEPHONE' EOL ;

% The following grammar rules are common to all categories
char = <any character except those that makes an End
Of Line>;
graphic_char = <any char except control_chars and space> ;
space = '' | <TAB> ;
SP = space+ ;
EOL = end_of_line | comment end_of_line ;
end_of_line = <anything that makes an End Of Line (EOL) in
the operating system employed> ;
comment_char = <defined by the 'comment_char' keyword> ;
escape_char = <defined by the 'escape_char' keyword> ;
charsymbol = simple_symbol | ucs_symbol ;
collsymbol = simple_symbol ;
collelement = simple_symbol ;
sectionsymbol = simple_symbol ;
occtdigit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' ;
digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
hex_upper = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | digit ;
hexdigit = hex_upper | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' ;
letter = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' ;
portable_graph_gtr = letter | digit | ! | " | \ # | $ | % | & | ’ | ( | ) | * | + | , | - | . | / | : | ; | < | = | ? | @ | [ | ] | ^ | _ | ` | { | | } | ~ |

portable_graph = portable_graph_gtr | ’ > ’ ;
portable_char = portable_graph | ’ ’ | < NUL > | < ALERT > |
| < BACKSPACE > | < TAB > | < CARRIAGE_RETURN > |
| < NEWLINE > | < VERTICAL_TAB > | < FORM_FEED > ;

octal_char = escape_char octdigit octdigit octdigit* ;
hex_char = escape_char ’ x ’ hexdigit hexdigit hexdigit* ;
decimal_char = escape_char ’ d ’ digit digit digit* ;
number = digit + ;
id_part = letter | digit | - | _ ;
four_digit_hex_string = hex_upper hex_upper hex_upper hex_upper ;
identifier = letter id_part* ;
simple_symbol = space* ’ < ’ portable_graph_gtr ’ > ’ ;
ucssymbol = space* ’ < U ’ four_digit_hex_string { four_digit_hex_string } [ four_digit_hex_string ] ’ > ’ ;
quoted_string = ’ ” ’ char_symbol ’ ” ’ ;
quoted_nonempty_string = ’ ” ’ char_symbol ’ ” ’ ;
char_symbol = char | charsymbol |
| octal_char | hex_char | decimal_char ;
elem_list = elem + ;
elem = char_symbol | collsymbol | collelement ;
symb_list = collsymbol + ;
FDCC_set_name = FDCC-name | ’ ” ’ FDCC-name ’ ” ’ ;
copy_FDCC_set = ’ copy ’ FDCC_set_name EOL ;
symbol = portable_graph + ;
semicolon = space* ’ ; ’ space* ;
comma = space* , space* ;
comment = comment_char ;
Annex D
(informative)

Outstanding issues

This Technical Report presents a trial for defining a general mechanism to specify cultural conventions. Though its contents are developed in order to form a standard, it has been decided that it will be a technical report in order to give information to public earlier.

The preparer of this report, ISO/IEC JTC1/SC22, expects the rapid progress of internationalization in the field of information technology will solve the issues mentioned below, and that this technical report will be used as a base for a new standard in the near future.

D.1 Comments from the Japanese member body

Japan considered this document should not be published as an international standard for the following reasons:

1) It is not clear whether the features which have their origin in ISO/IEC 9945-2 -- POSIX Part 2 - works well or not, after its separation from ISO/IEC 9945-2. Japan considers some mechanisms, e.g. "copy", will not work outside the POSIX environments.

2) It is not clear whether it makes sense or not to have a default value, which may be considered as a recommendation, for each cultural convention item. Japan is afraid that those default values are considered as Global Uniformity values - see ISO/IEC TR 11017:1998 for details.

3) It is not clear whether each specification form fits for world-wide cultural variations or not.

D.2 Comments from the U.S. member body

The U.S. National Body continues to be extremely disappointed with the contents of this Technical Report. Among the serious technical problems we see in this document are:

1. As an extension of the POSIX locale syntax (cf. ISO/IEC 9945-2), this document maintains the drawbacks of POSIX as a "specification method for cultural conventions" per se. In fact, it exacerbates the weaknesses of POSIX in this regard by conflating more, poorly justified LC_XXX formal definitions into a monolithic FDCC-set construct. This was clearly done with a particular implementation model in mind, but does not follow, nor even seem to be particularly informed by best current practice in the internationalization of software.

2. In an attempt to extend the POSIX LC_CTYPE specification to cover the repertoire of ISO/IEC 10646-1, this document blunders badly in asserting the cultural contextualization of character properties for the UCS. The treatment of LC_CTYPE as part of locales, i.e., as part of cultural adaptability, is an artifact of POSIX architecture and results from the need to have a place to put localized differences for case mapping. But by cloning other character properties having nothing to do with case mapping into LC_CTYPE, the net effect is to create a second source for specification of UCS character properties, with attendant dangers of divergence and errors, and with inevitable difficulties of maintenance.
and versioning. The clear intent is to influence other ISO standards to obtain their character property definitions from this document, instead of by reference to the widely implemented UCS property tables published by the Unicode Consortium. This will lead to confusion and interoperability problems for character properties. It has demonstrably already been a problem for the maintenance of the COBOL standard.

3. Each of the categories in the FDCC-set description has unaddressed problems and limitations. Rather than being resolved during the development of this document, many of these limitations were simply asserted to be "requirements". It appears to us that those are limitations of a particular envisioned implementation, engendered by legacy compatibility issues with POSIX, rather than requirements following from the legitimate needs for specification of cultural conventions. Because of this, implementers attempting to make use of the FDCC-set categories are immediately faced with an unexplained host of problems and mismatches to the actual cultural adaptability which they are trying to specify and implement to meet customer needs for information technology.

4. The repertoire map and LC_CTYPE sections deal with the repertoire of ISO/IEC 10646 as it was in 1998, but nearly 55,000 more characters have been added to ISO/IEC 10646-1:2000 and ISO/IEC 10646-2:2001. It would be a serious mistake for a technical report to be published in 2002 that uses an obsolete repertoire of characters. Even for the characters which are in the repertoire, there are problems in the LC_CTYPE section. The classes to which characters are assigned - or in which they do not appear -- often differ from comparable property lists in the Unicode Standard without any reasonable rationale being given. Since many implementations currently base their character properties on the data files in the Unicode Standard, arbitrary departure from those values is a recipe for interoperability problems. For example, the punct class includes many currency symbols, but for no apparent reason omits such currency symbols as the drachma, dong, and kip signs. The digit class includes a large group of digits from many cultures, but does not include Myanmar, Ethiopic, FullWidth, and others that are included in the comparable Unicode class.

Furthermore, the print and graph classes in LC_CTYPE do not include any Han ideographs, even though thousands of ideographs have been in ISO/IEC 10646-1 since 1993. And the tolower/toupper classes do not include the fullwidth Latin character pairs, even though Japanese national standards do include such characters, and implementations must support case mappings of the fullwidth Latin letters.

5. The repertoire map itself is a completely unnecessary addition to this document. It is intended to document and promulgate a particularly bad collection of character mnemonic short strings. The U.S. views these "mnemonics" as confusing and irrelevant to the supposed scope of the TR. The need for short identifiers for characters can be met much better by the standard short UCS identifiers spelled out in ISO/IEC 10646, which are in widespread use.

6. The LC_MONETARY section attempts to add support for multiple currencies, but does so incorrectly. The idea was to cover the time period when many European countries would be using individual national currencies and also the euro. However, the definition allows users to create multiple names for currencies, implying that the names are synonyms of each other. This is incorrect. Deutschmarks and euros are not synonyms; they are two different currencies that could be used within one country at the same time. Similarly, French francs and euros also are not synonyms, but parts of LC_MONETARY
are written as if two currencies like these are the same thing.

Besides the fact that the LC_MONETARY support for dual currencies is incorrect, it also is moot. By February 28, 2002, all 12 members of the European Union will have retired their national currencies and adopted the euro for all transactions. The functionality described in this technical report will be moot before the TR is even finalized.

7. The LC_TIME section includes some changes that are incompatible with POSIX.2. Some week definitions that have depended on Sunday being considered the first day of the week are changed in this TR to use Monday as the first day of the week. This would break existing implementations.

Also in the LC_TIME section, timezone information has been added. The U.S. National Body objects strongly to this because such information already is separately defined via the TIMEZONE environment variable and does not belong in a locale or FDCC-set. Many countries span multiple time zones, and including timezone information makes it impossible to write a locale or FDCC-set to support such countries.

8. The new LC_XLITERATE section for character transliteration is significantly incomplete. It also doesn’t belong in a locale or FDCC-set anyway. Such functionality, where defined, should be similar to code set conversion - users should be able to pick any source and target, rather than having some limited set of transliterations hard-coded in an FDCC-set.

Even if one believes transliteration should be in an FDCC-set, the support in this TR is inadequate for international needs. The syntax provided here will not work for many Asian languages (and some others), and cannot be expanded in a compatible way in the future to support such languages. The limited string conversion functionality defined here is inadequate to the general problem of transliteration and is inappropriate for inclusion in this TR.

D.3 Comments from the Norwegian member body

The Norwegian member body is pleased to see this work published as a Technical Report and looks forward to a future International Standard based on this TR. It is noted that there is already a considerable installed base of software implementing much of these specifications, plus some specifications that were present in previous committee drafts of this Technical Report. As this Technical Report is a record of the work done in the project, we proposed that these previous specifications should be present in this document, but the WG20 decided against it, and we then would like to see the specifications in an amendment or revision of the document. We have noted below in which documents the specifications, examples, rationale and BNF may be found. The issues in question are:

D.3.1 LC_PAPER

The LC_PAPER category specified paper size. It had three keywords, a "copy" keyword, a keyword "height" for the paper height, and a keyword "width" for the paper width. The operands to these keywords was the measurement in millimeters. The "i18n" FDCC-set had the height and width set to 297 and 210 mm respectively, reflecting the standard ISO A4 paper format. The latest specification of this was in 14652 FCD2, SC22 N2869 (WG20 N634).
D.3.2 LC_MEASUREMENT

The LC_MEASUREMENT category specified which measurement system to use. It had the "copy" keyword and the keyword "measurement" with 3 possible values: 1 - the ISO 1000 system, 2 - the U.S.A. system, 3 - other. The "i18n" FDCC-set had the "measurement" set to 1, reflecting use of ISO 1000 measurements. The latest specification of this was in 14652 FCD 1, SC22 N2638 (WG20 N553).

D.3.3 Double increment hexadecimal symbolic ellipses

The double increment hexadecimal symbolic ellipses "%(2)." construct for the LC_CTYPE category worked like the symbolic hexadecimal ellipses ":", but generated only every other symbolic character name. The latest specification of this was in 14652 DTR 1, JTC 1 N6404, SC22 N3227 (WG20 N822).
Annex E
(informative)

Index

abbreviation 4.2   col_weight_max 4.4, 4.4.3
abday 4.7   collating-element 4.4
abmon 4.7   collating statements 4.4.1
absolute ellipses 4.3   collating-symbol 4.4.6
address 4.2   collating element 3.1.13
addresses 4.11   collating sequence 3.1.15
addset 5.1   collating-element 4.4.5
alpha 4.3.1   collating-symbol 4.4
alt_digits 4.7   collation 3.1.12
am_pm 4.7   combining 4.3.1
application 4.2   combining_level3 4.3.1
audience 4.2   comment_char 4.1.4.1, 5.1
blank 4.3.1   contact 4.2
block_separator 4.3.1   continuation line 4.1.2
byte 3.1.1   control characters 4.3.1
cal_direction 4.7   conversion_rate 4.5
category 4.2   copy 4.1.3, 4.2, 4.3.1, 4.4.2,
category names 4.1   4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12
category trailer 4.1   country_ab2 4.11
category header 4.1   country_ab3 4.11
category body 4.1   country_car 4.11
character 3.1.2   country_isbn 4.11
character, graphic 4.3.1   country_name 4.11
character, special 4.3.1   country_num 4.11
character representation 4.1.1   country_post 4.11
character, native digit 4.3.1   cultural convention 3.1.5
character, hexadecimal digit 4.3.1   currency_symbol 4.5
character, multibyte 4.1.1   d_fmt 4.7
character, decimal constant 4.1.1   d_t_fmt 4.7
character, hexadecimal constant 4.1.1   date field descriptors 4.7.1
character, space 4.3.1   date 4.2
character, octal constant 4.1.1   day 4.7
character, control 4.3.1   decimal_point 4.6
character, blank 4.3.1   default_missing 4.3.2
character, digit 4.3.1   definitions 3.1
character, punctuation 4.3.1   digit 4.3.1
character, printable 3.1.10   ellipses 4.3, 4.4.1, 5.1
character class 3.1.9   ellipses, absolute 4.3, 4.4.1
character, coded 3.1.3   ellipses, symbolic 4.3, 4.4.1, 5.1
Character set rationale B.2   email 4.2
charset 5, 4.1.4.4, 3.1.7   equivalence class 3.1.16
code_set_name 5.1   era 4.7
coded character 3.1.3   era_d_fmt 4.7
cntrl 4.3.1   era_year 4.7
desc set name 5.1   escape_char 4.1.4.2, 5.1.6
desc set rationale B.2   esqseq 5.1
euro B.1.3 LC_TELEPHONE rationale B.1.11
extended regular expression 4.8 LC_TIME 4.7
fax 4.2 LC_TIME rationale B.1.6
FDCC-set, definition 4.1 LC_XLITERATE 4.9
FDCC-set 4f LC_XLITERATE rationale B.1.8
FDCC-set 3.1.6 LC_X__ 4
FDCC-set rationale B.1 line continuation 4.1.4
first_weekday 4.7 lower 4.3.1
first_workday 4.7 map 4.3.1
frac_digits 4.5 mb_cur_max 5.1
graph 4.3.1 mb_cur_min 5.1
graphic characters 4.3.1 messages 4.8
grouping 4.6 modified date field descriptors 4.7.2
height 4.9 mon 4.7
include 4.3.2 mon_decimal_point 4.5
include 5.1 mon_grouping 4.5
include 4.3.2.2 mon_thousands_sep 4.5
int_curr_symbol 4.5 monetary 4.5
int_frac_digits 4.5 multicharacter collating element 3.1.14
int_n_cs_precedes 4.5 n_cs_precedes 4.5
int_n_sep_by_space 4.5 n_sep_by_space 4.5
int_n_sign_posn 4.5 n_sign_posn 4.5
int_p_cs_precedes 4.5 name_formatting 4.10
int_p_sep_by_space 4.5 name_fmt 4.10
int_p_sign_posn 4.5 name_gen 4.10
int_prefix 4.12 name_miss 4.10
int_select 4.12 name_mr 4.10
keywords 4.1 name_mrs 4.10
lang_ab 4.11 name_ms 4.10
lang_lib 4.11 negative_sign 4.5
lang_name 4.11 noexpr 4.8
lang_term 4.11 notations 3.2
language 4.2 numeric 4.6
LC_ADDRESS 4.11 operands 4.1
LC_ADDRESS rationale B.1.10 order_end 4.4.9, 4.4
LC_COLLATE 4.4 order_start 4.4, 4.4.8
LC_COLLATE rationale B.1.3 outdigit 4.3.1
LC_CTYPE 4.3 p_cs_precedes 4.5
LC_CTYPE rationale B.1.2 p_sep_by_space 4.5
LC_IDENTIFICATION 4.2 p_sign_posn 4.5
LC_IDENTIFICATION rationale B.1.1 portable character set 3.2.4
LC_MEASUREMENT D.3.2 positive_sign 4.5
LC_MESSAGES 4.8 POSIX 1
LC_MESSAGES rationale B.1.7 POSIX differences 4.4
LC_MONETARY 4.5 POSIX conformance 4.2
LC_MONETARY rationale B.1.4 postal addresses 4.11
LC_NAME 4.10 postal_fmt 4.11
LC_NAME rationale B.1.9 pre-category statements 4.1.4
LC_NUMERIC 4.6 print 4.3.1
LC_NUMERIC rationale B.1.5 printable character 3.1.10
LC_PAPER D.3.1 punct 4.3.1
LC_TELEPHONE 4.12 punctuation characters 4.3.1
rere define 4.3.2
references 2
reorder-after 4.4, 4.4.10
reorder-after rationale B.1.2.1
reorder-end 4.4, 4.4.11
reorder-section-after 4.4, 4.4.13
reorder-section-end 4.4, 4.4.14
repertoire rationale B.3
repertoire 6
repertoiremap 6, 3.1.8, 5.1, 4.1.4.3
revision 4.2
scope 1
section 4.4, 4.4.4, 4.4.12
source 4.2
space 4.3.1
special characters 4.3.1
symbol-equivalence 4.4, 4.4.7
symbolic ellipses 4.3, 5.1
symbolic name 4.1.1
syntax format 3.2.1
t_fmt 4.7
t_fmt_ampm 4.7
tel 4.2
tel_domFmt 4.12
tel_intFmt 4.12
telephone numbers 4.12
territory 4.2
text file 3.1.4
thousandsSep 4.6
timezone 4.7
title 4.2
tolower 4.3.1	
toupper 4.3.1
translitIgnore 4.9
transliteration 4.9
transliteration statements 4.9.1
upper 4.3.1
validFrom 4.5
validTo 4.5
visible glyph portable characters 3.2.4
week 4.7
white space 3.1.11
width 4.9
xdigit 4.3.1
yesexpr 4.8
BIBLIOGRAPHY

The following specifications are considered relevant to this Technical Report, in addition to the normative references.

CEPT, CEPT-MAILCODE, Country code for mail.

ISO 646, Information technology - ISO 7-bit coded character set for information interchange.

ISO/IEC 9899, Information technology - Programming language C.

ISO/IEC 14977, Information technology - Syntactic metalanguage - Extended BNF.

