LI18NUX 2000

Globalization Specification

Version 1.0 with Amendment ~~2~~3~~ DRAFT~~

Linux Internationalization Initiative (Li18nux)

1. Foreword

1.1 Scope

This document specifies interfaces and functionalities that must be supported by operating systems to run internationalized application software. This document also includes recommendations for operating systems to ease development of internationalized application software.

This specification only lists internationalization aspects of each functionality provided by the conforming operating systems.

1.2 Normative References

[POSIX.1]

ISO/IEC 9945-1:1996 Information technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language]

[POSIX.2]

ISO/IEC 9945-2:1993 Information technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities

[ISO C]

ISO/IEC 9899:1990 Programming Languages — C

ISO/IEC 9899:1990/Amd.1:1995 Programming Languages — C Amendment 1: C Integrity

[ISO C 99]

ISO/IEC 9899:1999 Programming Languages — C

[XCU5]

The Single UNIX Specification, Version 2

Commands and Utilities, Issue 5

(The Open Group CAE Specification C604)

[XBD5]

The Single UNIX Specification, Version 2

System Interface Definitions, Issue 5

(The Open Group CAE Specification C605)

[XSH5]

The Single UNIX Specification, Version 2

System Interfaces and Headers, Issue 5 (2 volumes)

(The Open Group CAE Specification C606)

[XCURSES4.2]

The Single UNIX Specification, Version 2

X/Open Curses (XCurses), Issue 4 Version 2

(The Open Group CAE Specification C610)

[ICU]

International Components for Unicode 1.6.0

`http://oss.software.ibm.com/icu/`

[ICU4J]

International Components for Unicode for Java

`http://oss.software.ibm.com/icu4j/icu4jhtml/index.html`

[Perl 5.6]

Perl 5.6 (March 23, 2000)

`http://www.perl.com/pub/n/Perl_5.6.0_is_out!`

[Java]

Java 2 Platform, Standard Edition, v1.3 API Specification

`http://java.sun.com/products/jdk/1.3/docs/api/index.html`

[X11R6]

The X Window System, Version 11, Release 6

`ftp://ftp.x.org/pub/R6.4/xc/doc/hardcopy/`

[Unicode 3.0]

The Unicode Standard, Version 3.0

The Unicode Consortium, Addison Wesley Longman, ISBN 0-201-61633-5

[ISO 10646-1]

ISO/IEC 10646-1:2000 Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane

[ISO 639]

ISO 639:1988 Code for the representation of names of languages

[ISO 3166-1]

ISO 3166-1:1997 Codes for the representation of names of countries and their subdivisions — Part 1: Country codes

[IANA-Charset-Registry]

IANA Registry of Character Sets
`http://www.isi.edu/in-notes/iana/assignments/character-sets`

[ISO 8859-1]

ISO/IEC 8859-1:1998 Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1

[ISO 8859-2]

ISO/IEC 8859-2:1999 Information technology — 8-bit single-byte coded graphic character sets — Part 2: Latin alphabet No. 2

[ISO 8859-5]

ISO/IEC 8859-5:1999 Information technology — 8-bit single-byte coded graphic character sets — Part 5: Latin/Cyrillic alphabet

[ISO 8859-7]

ISO 8859-7:1987 Information processing — 8-bit single-byte coded graphic character sets — Part 7: Latin/Greek alphabet

[ISO 8859-9]

ISO/IEC 8859-9:1999 Information technology — 8-bit single-byte coded graphic character sets — Part 9: Latin alphabet No. 5

[ISO 8859-13]

ISO/IEC 8859-13:1998 Information technology — 8-bit single-byte coded graphic character sets — Part 13: Latin alphabet No. 7

[ISO 8859-15]

ISO/IEC 8859-15:1999 Information technology — 8-bit single-byte coded graphic character sets — Part 15: Latin alphabet No. 9

1.3 Conformance

1.3.1 Conforming Environments

For conformance purposes the following environments are defined:

(1) Application Execution Environment [Obsolescence]

Application Execution Environment is a minimum operating system environment that can run internationalized application software. The functionalities defined in this environment are mandatory and shall be present on all conforming implementations.

The following sections are applied to Application Execution Environment:

3. *Base Libraries*
4. *Shells and Utilities*

(2) End User Environment

End User Environment is an operating system environment with user interface. It is assumed that End User Environment has a set of utilities for user interaction.

This environment includes all the interfaces and utilities provided by Application Execution Environment. Additional interfaces and utilities are defined for the following sub-environments:

(a) Server Environment [Obsolescence]

Server environment is an operating system environment suitable for backend server purposes. Graphical user interfaces are not required in this environment.

The following sections are applied to Server Environment:

3. *Base Libraries*
4. *Shells and Utilities*
5. *Programming Languages (with Software Development Options)*
9. *Network Servers*

(b) Desktop Environment

Desktop environment is an operating system environment suitable for end user interaction. Graphical user interface is required in this environment.

The following sections are applied to Desktop Environment:

3. *Base Libraries*
4. *Shells and Utilities*
5. *Programming Languages (with Software Development Options)*
6. *Graphical User Interface*
7. *Input Methods*
8. *Output Methods*

10. Internet Tools

If an interface or utility is defined as “supported in End User Environment”, that interface or utility shall be available in both Server and Desktop environments.

The following options can be supported in each environment:

(3) Software Development Options

If any of these options is supported, utilities, libraries and associated modules to develop internationalized software (such as compilers or interpreters) shall be provided.

In this version of the specification, the following options are available:

- C Language Development Option
- Java Language Development Option

1.3.2 Conformance Levels

Several levels are defined for conformance for each environment. These levels are defined as follows:

(1) Level 1

The level 1 is the bottom-line level of conformance. All conforming implementations shall provide this level of interfaces and utilities to conform to this specification. If level is not specified in the specification, that specification shall be considered as Level 1.

(2) Level 2

The level 2 is more advanced or extended level of conformance. Conforming implementations are encouraged to provide this level of interfaces and utilities to conform to this specification, but it is not mandatory.

2. Terminology

2.1 Definition of Terms

The following terms are used in this specification:

Implementation-defined

A value or behavior is implementation-defined when it is left to the implementation to define [and document] the corresponding requirements for correct application behavior.

May

With respect to implementations, the word “may” is to be interpreted as an optional feature that is not required in this specification but can be provided. With respect to application, the word “may” means that the feature is optional. The term “optional” has the same definition as “may”.

Shall

In this specification, the word “shall” is to be interpreted as a mandatory requirement on the implementation or on application, depending upon the context. The term “must” has the same definition as “shall”.

Should

With respect to implementations, the word “should” is to be interpreted as an implementation recommendation, but not a requirement. With respect to application, the word “should” is to be interpreted as recommended programming practice.

Supported

Certain facilities in this specification are optional. If a facility is supported, it behaves as specified by this specification.

If a facility is “supported” by an implementation, the implementation must document how to obtain and install the facility, or the facility is installed by installer of the implementation by explicitly selected by the user or implicitly installed with other system components. If an implementation “supports” a facility, the distributor of the implementation shall commit that the facility can run on the implementation.

Unspecified

When a value or behavior is unspecified, the specification defines no portability requirements for a facility on an implementation even when faced with an application that uses the facility. An application that requires specific behavior in such an instance, rather than tolerating any behavior when using that facility, is not a portable application.

Provided

Certain facilities in this specification are mandatory and implemented in all conforming implementations.

Obsolescence

The indication of that subject statement or clause will be removed from future revision of this standard.

2.2 General Terms

character

A sequence of one or more bytes representing a single graphic symbol or control code.
This term corresponds to the ISO C standard term multibyte character (multi-byte character), where a single-byte character is a special case of a multi-byte character. Unlike the usage in the ISO C standard, character here has no necessary relationship with storage space, and byte is used when storage space is discussed.
[Single UNIX Specification, Version 2]

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; see *character*.
A byte is composed of a contiguous sequence of bits, the number of which is implementation-dependent. The least significant bit is called the low-order bit; the most significant is called the high-order bit.
Note that this definition of byte deviates intentionally from the usage of byte in some international standards, where it is used as a synonym for octet (always eight bits). On a system based on the ISO/IEC 9945-2:1993 standard, a byte may be larger than eight bits so that it can be an integral portion of larger data objects that are not evenly divisible by eight bits (such as a 36-bit word that contains four 9-bit bytes).
[Single UNIX Specification, Version 2]

character set

A finite set of different characters used for the representation, organization or control of data.
[Single UNIX Specification, Version 2]

coded character set

A set of unambiguous rules that establishes a character set and the one-to-one relationship between each character of the set and its bit representation.

[Single UNIX Specification, Version 2]

codeset

The result of applying rules that map a numeric code value to each element of a character set. An element of a character set may be related to more than one numeric code value but the reverse is not true. However, for state-dependent encodings the relationship between numeric code values to elements of a character set may be further controlled by state information.

The character set may contain fewer elements than the total number of possible numeric code values; that is, some code values may be unassigned.

[Single UNIX Specification, Version 2]

internationalization

The provision within a computer program of the capability of making itself adaptable to the requirements of different native languages, local customs and coded character sets.

[Single UNIX Specification, Version 2]

globalization

A product development approach which ensures that software products are usable in the worldwide markets through a combination of internationalization and localization.

locale

The definition of the subset of a user's environment that depends on language and cultural conventions.

[Single UNIX Specification, Version 2]

localization

The process of establishing information within a computer system specific to the operation of particular native languages, local customs and coded character sets.

[Single UNIX Specification, Version 2]

local customs

The conventions of a geographical area or territory for such things as date, time and currency formats.

[Single UNIX Specification, Version 2]

portable filename character set

The set of characters from which portable filenames are constructed. For a filename to be portable across implementations conforming to this specification set and the ISO POSIX-1 standard, it must consist only of the following characters:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9 . _ -
```

The last three characters are the period, underscore and hyphen characters, respectively. The hyphen must not be used as the first character of a portable filename. Upper- and lower-case letters retain their unique identities between conforming implementations. In the case of a portable pathname, the slash character may also be used.
[Single UNIX Specification, Version 2]

file-system-safe character

Multibyte character which does not contain either 0x00 or 0x2F in any byte of its representation.

Input Method Engine

A part or a module of building block of input method which implements a language- or a script-specific logic of composing a string from one or more sequence of event or a string, which can be independent from windowing system, graphical user interface, or visual appearance.

3. Base Libraries

(1) Scope

This chapter defines runtime library interfaces required to conform to this specification. Conforming implementations shall provide the C language APIs defined by [ISO C] and [POSIX.1]. In addition to the C language interface, conforming level 2 implementations shall provide interfaces for other programming languages.

(2) Requirements

Conforming implementations shall provide the internationalization functions listed in the Table 3-1 and the headers listed in the Table 3-2. The specifications of the functions and the definitions of the headers shall conform to [POSIX.1] and [ISO C].

In addition to the functions in the Table 3-1, conforming implementations shall provide the wide character and wide string I/O functionality through **printf**/**scanf** family of functions as specified in [ISO C].

Table 3-1 C Language internationalization functions

btowc()	**fgetwc()**	**fgetws()**	**fputwc()**	**fputws()**
fwide()	**fwprintf()**	**fwscanf()**	**getwc()**	**getwchar()**
iswalnum()	**iswalpha()**	**iswcntrl()**	**iswctype()**	**iswdigit()**
iswgraph()	**iswlower()**	**iswprint()**	**iswpunct()**	**iswspace()**
iswupper()	**iswxdigit()**	**localeconv()**	**mblen()**	**mbrlen()**
mbrtowc()	**mbsinit()**	**mbsrtowcs()**	**mbstowcs()**	**mbtowc()**
putwc()	**putwchar()**	**setlocale()**	**strftime()**	**swprintf()**
swscanf()	**towctrans()**	**towlower()**	**towupper()**	**ungetwc()**
vfwprintf()	**vswprintf()**	**vwprintf()**	**wcrtomb()**	**wcscat()**
wcschr()	**wcscmp()**	**wcscoll()**	**wcscpy()**	**wcscspn()**
wcsftime()	**wcslen()**	**wcsncat()**	**wcsncmp()**	**wcsncpy()**
wcspbrk()	**wcsrchr()**	**wcsrtombs()**	**wcsspn()**	**wcsstr()**
wcstod()	**wcstok()**	**wcstol()**	**wcstombs()**	**wcstoul()**
wcsxfrm()	**wctob()**	**wctomb()**	**wctrans()**	**wctype()**
wmemchr()	**wmemcmp()**	**wmemcpy()**	**wmemmove()**	**wmemset()**
wprintf()	**wscanf()**			

Table 3-2 C language headers

<locale.h>	**<wchar.h>**	**<wctype.h>**

Note: Application programs should refer to limits in symbolic names, such as MB_CUR_MAX and MB_LEN_MAX, not the implementation-specific values directly.

Conforming level 2 implementations shall provide the following functions. The specifications of the functions shall conform to [ISO C 99].

wcstof()	**wcstold()**	**wcstoll()**	**wcstoull()**

Conforming implementations shall provide the internationalization functions listed in the Table 3-3 and headers listed in the Table 3-4. The specifications of the functions and the definitions of the headers shall conform to [XSH5].

Table 3-3 Additional C Language internationalization functions

catclose()	**catgets()**	**catopen()**
iconv()	**iconv_close()**	**iconv_open()**
nl_langinfo()	**strfmon()**	**strptime()**
wcswidth()	**wcwidth()**	

Table 3-4 Additional C language headers

<iconv.h>	**<langinfo.h>**	**<monetary.h>**	**<nl_types.h>**

Conforming implementations shall provide the message handling functions listed in Table 3-5 and headers listed in Table 3-6 which is specified in *Annex C: Publicly Available Specifications*.

Table 3-5 Additional message handling functions

gettext()	**dgettext()**	**textdomain()**	**bindtextdomain()**
dcgettext()	**ngettext()**	**dngettext()**	**dcngettext()**
bind_textdomain_codeset()			

Table 3-6 Additional message handling functions headers

<libintl.h>

Conforming level 1 implementations should support the POSIX regular expression functions listed in the Table 3-7 and the header **<regex.h>**.

The specifications of the functions and the definitions of the header should conform to [XSH5].

Table 3-7 POSIX regular expression functions

regcomp()	**regexec()**	**regerror()**	**regfree()**

Conforming implementations shall provide the application execution environment in which the internationalized applications (written by using the internationalization functions above) can behave appropriately depending on the value of environment variables, without requiring any change of the applications.

See *Annex A: Environment Variables* for the environment variables to which internationalization functions will refer.

Conforming implementations shall support the application execution environments specified in *Annex B*.

Conforming level 2 implementations shall define **_XOPEN_CURSES** version test macro and provide the internationalized curses library functions which are specified in [XCURSES4.2].

Conforming level 2 implementations shall support Java Runtime environment ([Java]), Internationalization Components for Unicode [ICU], ICU for Java [ICU4J], and Perl execution environment [Perl 5.6] including Perl interpreter and modules.

The following Perl modules are related with internationalization:

(see `http://www.perl.com/CPAN-local/modules/00modlist.long.html#Part2-ThePerl5M`)

Name	Description
I18N::	
::Charset	Character set names and aliases
::Collate	Locale based comparisons
::LangTags	compare & extract language tags (RFC1766)
::WideMulti	Wide and multibyte character string
Locale::	
::Country	ISO 3166 two letter country codes
::Date	Month/weekday names in various languages
::Langinfo	The <langinfo.h> API

::Language	ISO 639 two letter language codes
::Msgcat	Access to XPG4 message catalog functions
::PGetText	What GNU gettext does, written in pure perl
::gettext	Multilanguage messages
Unicode::	
::String	String manipulation for Unicode strings
::Map8	Convert between most 8bit encodings

(3) Implementation Examples

GNU C library version 2.2

(4) Future Direction

In the next version of this specification, conforming implementations may be required to provide POSIX regular expression functions and internationalized curses library functions.

4. Shells and Utilities

(1) Scope

This chapter defines runtime environment required to support traditional UNIX command interpreter called "shell" and other basic utilities defined in [POSIX.2].

(2) Requirements

- Shell implementation

Conforming level 1 implementations shall be able to use Portable Filename Character Set defined in [POSIX.2]. For filename globbing, conforming level 1 implementations shall provide the functionality defined in [POSIX.2], with the following exceptions: [refer to Annex F: A]

- Range expression (such as `[a-z]`) can be based on code point order instead of collating element order.
- Equivalence class expression (such as `[=a=]`) and multi-character collating element expression (such as `[.ch.]`) are optional.
- Handling of a multi-character collating element is optional.

Conforming level 2 implementations shall be able to use file-system-safe characters as arguments and filenames.

Conforming level 2 implementations shall implement the globbing functionality of the shell as defined in [POSIX.2].

Conforming implementations shall provide a shell that supports the functionalities of "Bourne shell", with internationalization capabilities defined above.

- The utilities implementation

(a) Locale

Conforming implementations shall provide the following utilities to generate and refer to locale definitions as specified in [XCU5]:

locale	**localedef**

(b) Text Editor

Conforming implementations shall provide the following utilities to edit text files encoded in the supported codesets as specified in [XCU5].

> **Note:** To *edit* text is to determine character boundaries correctly and perform operations such as insert, copy and delete characters based on the determined character boundaries. Input and output requirements are specified in *7. Input Methods* and *8. Output Methods* respectively.

ed	**ex**	**vi**

(c) Date and Time formatting

~~Conforming implementations shall provide the following utilities to display locale-specific date and time formats as specified in [XCU5]:~~ [refer to Annex F: B]

~~at~~	**~~cal~~**	**~~cpio~~**	**~~date~~**	**~~ls~~**	**~~ps~~**	**~~tar~~**	**~~Time~~**

~~In the "C" and "POSIX" locales, the date and time formats used by the utilities shall be in fixed formats for ease of parsing, for the messages can be used as input to other programs. In the other locales, the date and time formats should change depending on the current locale for end-users' ease~~This specification has no requirements on date and time formatting functionality of shells and utilities.

(d) Text Processing

Conforming implementations shall provide the following utilities to process text as specified in [XCU5].

~~comm~~	**~~diff~~**	**~~egrep~~**	**~~expand~~**	**~~fgrep~~**	**~~fold~~**
comm	**diff**	**egrep**	**expand**	**fgrep**	**fold**
~~grep~~	**~~iconv~~**	**~~join~~**	**~~more~~**	**~~mailx~~**	**~~man~~**
grep	**iconv**	**join**	**more**	**mailx**	
nm (symbol sorting order)		**od** (floating point)		**pr**	**printf**
~~sed~~	**~~sort~~**	**~~tr~~**	**~~unexpand~~**	**~~uniq~~**	**~~wc~~**
sed	**sort**	**unexpand**	**uniq**	**wc**	

The **mailx** utility can be implemented as **Mail**. The **more** utility can be implemented as **less**.

(e) Regular Expressions

On conforming level 2 implementations, utilities that process regular expressions shall support Basic Regular Expression (BRE) and Extended Regular Expression (ERE) as specified in [POSIX.2].

On conforming level 1 implementations, utilities that process regular expressions should support BRE and ERE as specified in [POSIX.2]~~. If an implementation is not able to support BRE and ERE, it may support the regular expression syntax defined in **re_comp()** of [XSH5] instead of BRE and the regular expression syntax defined in **regcmp()** of [XSH5] instead of ERE.~~, with the following exceptions: [refer to Annex F: A]

- Range expression (such as `[a-z]`) can be based on code point order instead of collating element order.
- Equivalence class expression (such as `[=a=]`) and multi-character collating element expression (such as `[.ch.]`) are optional.
- Handling of a multi-character collating element is optional.

The following utilities are relevant:

egrep	**grep**	**sed**	**awk**

(f) Filename Handling

Conforming implementations shall provide the following utilities to correctly handle filenames that use file-system-safe characters. For filename globbing, conforming level 1 implementations shall provide the functionality defined in [POSIX.2], with the following exceptions: [refer to Annex F: A]

- Range expression (such as `[a-z]`) can be based on code point order instead of collating element order.
- Equivalence class expression (such as `[=a=]`) and multi-character collating element expression (such as `[.ch.]`) are optional.
- Handling of a multi-character collating element is optional.

cpio	**find**	**ls**	**tar**

(g) General Text Editor

Conforming implementations shall support at least one text editor that can edit text encoded in UTF-8.

Note: To *edit* text is to determine character boundaries correctly and perform operations such as insert, copy and delete characters based on the determined character boundaries. Input and output requirements are specified in *7. Input Methods* and *8. Output Methods* respectively.

(h) Terminal Emulator

Conforming implementations shall support terminal emulators that can handle codesets for supported locales.

Conforming implementations should support terminal emulation for all supported locales, but an implementation may provide different terminal emulators for each locale.

(i) Message catalogs

Conforming implementations shall provide the following utilities to convert message catalog source files into message catalogs.

gencat	**msgfmt**

Conforming implementations with C Language Development Option shall provide the following utilities to create and update message catalog source files.

msgmerge	**xgettext**

(j) Message Handling

Conforming implementations shall provide the following utility to handle localized messages.

gettext

(3) Implementation Examples

Examples of level 1 implementation

GNU bash

GNU textutils

GNU shellutils

GNU fileutils

Terminal Emulators:

kterm and **kon**.

jfbterm, supporting CJK, working under frame buffer, output only.

rxvt, supporting CJK, working under X Window System.

Unicon available at:

`http://turbolinux.com.cn/TLDN/chinese/project/unicon/`

zhcon by Bluepoint Corp.:

`http://openunix.org/`

cce (Console Terminal) available at:

`http://programmer.lib.sjtu.edu.cn/cce/cce.html`

XLinux console, supporting 12 languages:

`http://www.xlinux.com.tw/`

Unicode fonts and tools for X11:

`http://www.cl.cam.ac.uk/~mgk25/ucs-fonts.html`

XFree86 4.0.1 (includes already the above):

`http://www.zepler.org/~rwb197/xterm/`

(4) Future Direction

In a future version of this specification, shell's function of handling file-system-safe characters will become mandatory.

5. Programming Languages

(1) Scope

This chapter defines the requirements for various programming languages. Only programming languages with internationalization requirements are listed here. Note that the specifications defined by this chapter shall be provided by conforming implementations if the relevant Software Development Option is supported.

(2) Requirements

Conforming level 2 implementations with Software Development Options shall support the compiler or interpreter for the following languages:

- C (if the implementation supports the C Language Development Option)
- Java (if the implementation supports the Java Language Development Option)
- Perl

Each programming language shall be internationalized as specified in the following specifications:

- C language as specified in [ISO C]
- Java language as specified in [Java]
- Perl language as specified in [Perl 5.6]

Note: See *3. Base Libraries* about runtime environment of Perl and Java languages.

(3) Implementation Examples

The following implementation examples are available for these languages:

C: GNU Compiler Collection

`http://www.gnu.org/software/gcc/gcc.html`

C: Fortran & C Package (Linux)

Fujitsu Kyushu System Engineering Limited (in Japan)

`http://www.fqs.co.jp/fort-c/`

Fujitsu C/C++ Express (Linux)

Fujitsu America Inc. (in US)

`http://www.tools.fujitsu.com/`

Perl:

`http://www.perl.com/pub/n/Perl_5.6.0_is_out!`

Java:

`http://java.sun.com/`

(4) Future Directions

None

6. Graphical User Interface

6.1 Graphic Libraries

(1) Scope

This chapter defines runtime library interfaces for graphical user interface (GUI). Conforming implementations shall provide the graphical user interface defined by the X Window System Version 11 Release 6 [X11R6].

(2) Requirements

Conforming implementations shall provide the API for following functions:

- Locale

  setlocale()
  XSupportsLocale()
  XSetLocaleModifiers()

- Internationalized Text Drawing

  XCreateFontSet() — not recommended (use **XOpenOM()**/**XCreateOC()**)
  XFreeFontSet()
  XFontsOfFontSet()
  XBaseFontNameListOfFontSet()
  XLocaleOfFontSet()
  XContextDependentDrawing()
  XExtentsOfFontSet()
  XmbTextEscapement()
  XwcTextEscapement()
  XmbTextExtents()
  XwcTextExtents()
  XmbTextPerCharExtents()
  XwcTextPerCharExtents()
  XmbDrawString()
  XwcDrawString()
  XmbDrawImageString()
  XwcDrawImageString()
  XmbDrawText()

XwcDrawText()

- X Output Methods—X11R6 Extension

XOpenOM()

XCloseOM()

XDisplayOfOM()

XLocaleOfOM()

XSetOMValues()

XGetOMValues()

XCreateOC()

XDestroyOC()

XOMOfOC()

XSetOCValues()

XGetOCValues()

- Resource Management

XrmInitialize()

XrmLocaleOfDatabase()

XrmParseCommand()

XResourceManagerString()

XScreenResourceString()

XrmGetFileDatabase()

XrmGetStringDatabase()

XrmMergeDatabases()

XrmCombineDatabase()

XrmCombineFileDatabase()

XrmGetDatabase()

XrmSetDatabase()

XrmGetResource()

XrmEnumerateDatabase()

XrmPutResource()

XrmPutStringResource()

XrmPutLineResource()

XrmPutFileDatabase()

XrmDestroyDatabase()

XrmPermStringToQuark()

XrmQGetResource()

XrmQGetSearchList()

XrmQGetSearchResource()

XrmQPutResource()

XrmQPutStringResource()

XrmQuarkToString()

XrmStringToBindingQuarkList()

XrmStringToQuark()

XrmStringToQuarkList()

XrmUniqueQuark()

- Inter-Client Communication

XmbTextListToTextProperty()

XwcTextListToTextProperty()

XmbTextPropertyToTextList()

XwcTextPropertyToTextList()

XFreeStringList()

XwcFreeStringList()

XmbSetWMProperties()

XSetWMProperties()

XSetWMName()

XSetWMIconName()

- X Input Methods—Internationalized Text Input

XOpenIM()

XCloseIM()

XDisplayOfIM()

XLocaleOfIM()

XSetIMValues()

XGetIMValues()

XCreateIC()

XVaCreateNestedList()

XDestroyIC()

XIMOfIC()

XSetICValues()

XGetICValues()

XSetICFocus()

XUnsetICFocus()

XmbResetIC()

XwcResetIC()

XFilterEvent()

XmbLookupString()

XwcLookupString()

XRegisterIMInstantiateCallback()

XUnregisterIMInstantiateCallback()

Conforming level 2 implementations shall support languages listed in *Annex B*. Conforming level 1 implementations need not to support languages that require complex text layout (the applicable languages are marked in the table in *Annex B*).

(3) Implementation Examples

The following implementation example is available for this category.

XFree86 4.0.1:
`http://www.xfree86.org/`

(4) Future Direction

None

6.2 Graphic Toolkits and X Window Servers

(1) Scope

This chapter defines the requirements for graphic toolkits supported on top of the X Window System and the X Window System servers.

(2) Requirements

- Graphic Toolkits

  There are no requirements on the Graphic Toolkits in terms of internationalization.

- X Window Servers

There is no requirement on the X Window Servers in terms of internationalization.~~Conforming implementations shall support X11R6-based X servers and font servers which support outline fonts.~~

(3) Implementation Examples

The following implementation examples are available for this category.

[Graphic Toolkits]

GTK+:

`http://www.gtk.org/`

Qt:

`http://www.troll.no/products/qt.html`

[X Window Server which supports outline fonts]

X-TrueType Server (X-TT):

`http://X-TT.dsl.gr.jp/index.html`

XFree86 4.0.1:

`http://www.xfree86.org/`

(4) Future Directions

In a future version of this specification, Unicode, BiDi (bidirectional text), and vertical writing will become requirements.

7. Input Methods

(1) Scope

This chapter defines the requirements for text input used by the X Window System and other environments. Such mechanism is needed to support non-Western languages (for example, Chinese, Japanese and Korean).

(2) Requirements

Conforming implementations shall provide means, i.e., Input Method(s) for user to input characters specified in the *Annex B: Supported locales and codesets*.

Conforming implementations shall provide X Input Method Server(s) which can connect with Input Method Engines of the supported locales. An Input Method Engine can be implemented as a separate process communicating with an X Input Method Server or can be integrated into the X Input Method Server.

Conforming implementations shall support Input Method Engines for the supported locales, that can be connected with the above Input Method Server(s). The conforming implementations shall document which Input Method Engines are supported by the above X Input Method Server(s) and how user can get and install the Engines into the conforming implementations.

The X Input Method Server(s) should have a capability to switch Input Method Engines dynamically, but a conforming implementation may provide multiple Input Method Servers per locale.

Conforming level 1 implementations should provide an X Input Method Server which supports UTF-8 encoding and allows user to input whole repertoire of [Unicode 3.0].

Conforming level 2 implementations shall provide an X Input Method Server which supports UTF-8 encoding and allows user to input whole repertoire of [Unicode 3.0].

> **Note:** User-friendly input operation is preferable, but it is acceptable to use non-user-friendly input operation, such as entering hexadecimal code points, to input not-so-frequently-used characters. Also note that the *input* requirement does not imply that the input characters are displayed correctly.

Conforming implementations may provide X Input Method Server(s) which supports locale specific character repertoire and locale specific character encodings.

Every application that has X Window System based GUI and has a capability to accept character input from users should have the interface with the above X Input Method Server(s).

Conforming implementations should provide means for user to input characters specified in the supported locale through Console and TTY device interfaces.

(3) Implementation Examples

X Input Method Server (Generic): **IIIMF**

X Input Method Servers (Japanese): **kinput2**, and **Xwnmo**.

X Input Method Servers (Chinese):

Chinput, supporting both GB and Big5

`http://turbolinux.com.cn/~justiny/project-chinput.html`

xcin, supporting both Big5 and GB

`http://xcin.linux.org.tw/`

X Input Method Servers (Korean): **ami**, **hanIM** and **byeoroo**

Chinese Console:

supports CJK and Big5 display and input with a platform-independent input server

`http://www.redflag-linux.com/news/open.htm`

`yh-3.1-opensource.tgz`

(4) Future Direction

In the next version of this specification, the recommendation of single X Input Method Server which can switch Input Method Engines dynamically will become mandatory requirement.

In the next version of this specification, the recommendation for conforming level 1 implementations regarding the X Input Method Server(s) which support UTF-8 encoding will become mandatory requirement.

8. Output Methods

(1) Scope

This chapter defines the requirements for text output used by the X Window System. Such mechanism is needed to support languages that require complex text rendering.

(2) Requirements

Conforming implementations shall provide means, i.e., Output Method(s), for user to output characters specified in the *Annex B: Supported locales and codesets*.

Conforming implementations shall provide X Output Method interface defined in X11R6 Xlib specification chapter 13 as a displaying primitive for X Window System.

Conforming level 1 implementations should provide multibyte and wide character interface which cover the following collections of UCS implementation level 1 defined in [ISO 10646-1].

Conforming level 2 implementations shall provide multibyte and wide character interface which cover the following collections of UCS implementation level 1 defined in [ISO 10646-1].

> **Note:** [ISO 10646-1] defines character blocks for subsetting purpose and are called *character collections*. Such character collections are used here to indicate minimum displayable subset.

1	BASIC LATIN	0020-007E
2	LATIN-1 SUPPLEMENT	00A0-00FF
3	LATIN EXTENDED-A	0100-017F
4	LATIN EXTENDED-B	0180-024F
5	IPA EXTENSIONS	0250-02AF
8	BASIC GREEK	0370-03CF
9	GREEK SYMBOLS AND COPTIC	03D0-03FF
10	CYRILLIC	0400-04FF
11	ARMENIAN	0530-058F
27	BASIC GEORGIAN	10D0-10FF
30	LATIN EXTENDED ADDITIONAL	1E00-1EFF
31	GREEK EXTENDED	1F00-1FFF
32	GENERAL PUNCTUATION	2000-206F (only graphical characters)
33	SUPERSCRIPTS AND SUBSCRIPTS	2070-209F
34	CURRENCY SYMBOLS	20A0-20CF
36	LETTERLIKE SYMBOLS	2100-214F

37	NUMBER FORMS	2150-218F
38	ARROWS	2190-21FF
39	MATHEMATICAL OPERATORS	2200-22FF
40	MISCELLANEOUS TECHNICAL	2300-23FF
41	CONTROL PICTURES	2400-243F
42	OPTICAL CHARACTER RECOGNITION	2440-245F
44	BOX DRAWING	2500-257F
45	BLOCK ELEMENTS	2580-259F
46	GEOMETRIC SHAPES	25A0-25FF
47	MISCELLANEOUS SYMBOLS	2600-26FF
49	CJK SYMBOLS AND PUNCTUATION	3000-303F
50	HIRAGANA	3040-309F
51	KATAKANA	30A0-30FF
52	BOPOMOFO	3100-312F
54	CJK MISCELLANEOUS	3190-319F
55	ENCLOSED CJK LETTERS AND MONTHS	3200-32FF
56	CJK COMPATIBILITY	3300-33FF
60	CJK UNIFIED IDEOGRAPHS	4E00-9FFF
62	CJK COMPATIBILITY IDEOGRAPHS	F900-FAFF
66	CJK COMPATIBILITY FORMS	FE30-FE4F
69	HALFWIDTH AND FULLWIDTH FORMS	FF00-FFEF
71	HANGUL EXTENDED	AC00-D7A3
76	YI SYLLABLES	A000-A48F
77	YI RADICALS	A490-A4CF
81	CJK UNIFIED IDEOGRAPHS EXTENSION A	3400-4DBF

Conforming implementations should provide an X Output Method which supports the encoding schemes listed in *Annex B*.

Conforming implementations shall provide a terminal emulator on the X Window System that output characters in the supported locale.

Conforming implementations should provide console or tty device interface that output characters in the supported locale.

(3) Implementation Examples

X11R6.4 Xlib, and IIIMXCF

xterm patches available at:

```
http://www.zepler.org/~rwb197/xterm/
```

(4) Future Direction

None

9. Network Servers

(1) Scope

This chapter defines the requirements for various network servers, such as file sharing servers and WWW servers.

The requirements on the following kinds of servers will be discussed in this section.

- NetBIOS over TCP/IP
- AppleTalk
- Network File System
- HTTP Server

(2) Requirements

This version of the specification has no requirements for the Network Servers.

(3) Implementation Examples

None

(4) Future Directions

In a future version of this specification, the requirements on the handling of names, e.g., filename, domain name, resource name, and user name, will be specified in this section.

10. Internet Tools

(1) Scope

This chapter defines the requirements for Internet client tools, such as WWW browsers and Mail User Agents (MUAs).

(2) Requirements

Conforming implementations shall make at least one codeset available per locale specified in *Annex B*.

The supported codeset should be in [IANA-Charset-Registry].

Conforming level 2 implementations of Web browsers and mail user agents shall be able to input and output whole repertoire of [Unicode 3.0].

Note: Character output is restricted as specified in *8. Output Methods*.

(3) Implementation Examples

The following implementation examples are available for this category.

Mozilla
`http://www.mozilla.org/`

mutt
`http://www.mutt.org/`

(4) Future Direction

None

11. Printing

(1) Scope

This chapter defines requirements related to printing, such as APIs, utilities and their behavior.

(2) Requirements

This version of the specification has no requirements for printing.

(3) Implementation Examples

None

(4) Future Direction

In a future version of this specification, requirements from the Printing subgroup of the Li18nux working group will be provided.

Annex A (Normative): Environment Variables

Conforming implementations shall provide the following environment variables that are relevant to the operation of internationalized interfaces or internationalized commands and utilities.

LANG

LC_ALL

LC_COLLATE

LC_CTYPE

LC_MESSAGES

LC_MONETARY

LC_NUMERIC

LC_TIME

NLSPATH

The usage and the semantics of these environment variables shall be the same as the description in *"6.2 Internationalisation Variables"* in [XBD5].

Annex B (Normative): Supported locales and codesets

Conforming implementations shall provide handling capability of the following locales.

C

POSIX

Conforming implementations shall support the following locales.

Note 1: The language names come from ISO 639.

Note 2: To avoid political discussion, the region/country names used here does not strictly follow ISO 3166-1.

af_ZA	Afrikaans	SOUTH AFRICA	[Support of this locale is level 2]
ar_AE	Arabic	UNITED ARAB EMIRATES	[Output method support is level 2]
ar_BH		BAHRAIN	[Output method support is level 2]
ar_DZ		ALGERIA	[Output method support is level 2]
ar_EG		EGYPT	[Output method support is level 2]
ar_IN		INDIA	[Support of this locale is level 2] [Output method support is level 2]
ar_IQ		IRAQ	[Output method support is level 2]
ar_JO		JORDAN	[Output method support is level 2]
ar_KW		KUWAIT	[Output method support is level 2]
ar_LB		LEBANON	[Output method support is level 2]
ar_LY		LIBYAN ARAB JAMAHIRIYA	[Output method support is level 2]
ar_MA		MOROCCO	[Output method support is level 2]
ar_OM		OMAN	[Output method support is level 2]
ar_QA		QATAR	[Output method support is level 2]
ar_SA		SAUDI ARABIA	[Output method support is level 2]
ar_SD		SUDAN	[Output method support is level 2]
ar_SY		SYRIAN ARAB REPUBLIC	[Output method support is level 2]
ar_TN		TUNISIA	[Output method support is level 2]
ar_YE		YEMEN	[Output method support is level 2]
as_IN	Assamese	INDIA	[Support of this locale is level 2] [Output method support is level 2]
be_BY	Byelorussian	BELARUS	
bg_BG	Bulgarian	BULGARIA	
bn_IN	Bengali	INDIA	[Support of this locale is level 2] [Output method support is level 2]

ca_ES	Catalan	SPAIN	
cs_CZ	Czech	CZECH REPUBLIC	
da_DK	Danish	DENMARK	
de_AT	German	AUSTRIA	
de_BE		BELGIUM	[Support of this locale is level 2]
de_CH		SWITZERLAND	
de_DE		GERMANY	
de_LU		LUXEMBOURG	
el_GR	Greek	GREECE	
en_AU	English	AUSTRALIA	
en_BE		BELGIUM	
en_BW		BOTSWANA	[Support of this locale is level 2]
en_CA		CANADA	
en_GB		UNITED KINGDOM	
en_HK		HONG KONG	[Support of this locale is level 2]
en_IE		IRELAND	
en_IN		INDIA	[Support of this locale is level 2]
en_NZ		NEW ZEALAND	
en_PH		PHILIPPINES	[Support of this locale is level 2]
en_SG		SINGAPORE	[Support of this locale is level 2]
en_US		UNITED STATES	
en_ZA		SOUTH AFRICA	
en_ZW		ZIMBABWE	[Support of this locale is level 2]
es_AR	Spanish	ARGENTINA	
es_BO		BOLIVIA	
es_CL		CHILE	
es_CO		COLOMBIA	
es_CR		COSTA RICA	
es_DO		DOMINICAN REPUBLIC	
es_EC		ECUADOR	
es_ES		SPAIN	
es_GT		GUATEMALA	
es_HN		HONDURAS	
es_MX		MEXICO	
es_NI		NICARAGUA	
es_PA		PANAMA	

es_PE		PERU	
es_PR		PUERTO RICO	
es_PY		PARAGUAY	
es_SV		REPUBLIC OF EL SALVADOR	
es_UY		URUGUAY	
es_VE		VENEZUELA	
et_EE	Estonian	ESTONIA	
eu_ES	Basque	SPAIN	[Support of this locale is level 2]
fa_IN	Persian	INDIA	[Support of this locale is level 2] [Output method support is level 2]
fa_IR		IRAN, ISLAMIC REPULIC OF	[Support of this locale is level 2] [Output method support is level 2]
fi_FI	Finnish	FINLAND	
fo_FO	Faroese	FAROE ISLANDS	
fr_BE	French	BELGIUM	
fr_CA		CANADA	
fr_CH		SWITZERLAND	
fr_FR		FRANCE	
fr_LU		LUXEMBOURG	
ga_IE	Irish	IRELAND	
gl_ES	Galician	SPAIN	[Support of this locale is level 2]
gu_IN	Gujarati	INDIA	[Support of this locale is level 2] [Output method support is level 2]
gv_GB	Manx Gaelic	UNITED KINGDOM	[Support of this locale is level 2]
he_IL	Hebrew	ISRAEL	[Output method support is level 2]
hi_IN	Hindi	INDIA	[Support of this locale is level 2] [Output method support is level 2]
hr_HR	Croatian	CROATIA	
hu_HU	Hungarian	HUNGARY	
id_ID	Indonesian	INDONESIA	[Support of this locale is level 2]
is_IS	Icelandic	ICELAND	
it_CH	Italian	SWITZERLAND	
it_IT		ITALY	
ja_JP	Japanese	JAPAN	
kl_GL	Greenlandic	GREENLAND	
kn_IN	Kannada	INDIA	[Support of this locale is level 2]

			[Output method support is level 2]
ko_KR	Korean	KOREA, REPUBLIC OF	
ks_IN	Kashmiri	INDIA	[Support of this locale is level 2] [Output method support is level 2]
kw_GB	Cornish	UNITED KINGDOM	[Support of this locale is level 2]
lt_LT	Lithuanian	LITHUANIA	
lv_LV	Latvian, Lettish	LATVIA	
mk_MK	Macedonian	MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	
ml_IN	Malayalam	INDIA	[Support of this locale is level 2] [Output method support is level 2]
ms_MY	Malay	MALAYSIA	[Support of this locale is level 2]
nl_BE	Dutch	BELGIUM	
nl_NL		NETHERLANDS	
no_NO	Norwegian	NORWAY	
or_IN	Oriya	INDIA	[Support of this locale is level 2] [Output method support is level 2]
pa_IN	Punjabi	INDIA	[Support of this locale is level 2] [Output method support is level 2]
pl_PL	Polish	POLAND	
ps_IN	Pashto, Pushto	INDIA	[Support of this locale is level 2] [Output method support is level 2]
pt_BR	Portuguese	BRAZIL	
pt_PT		PORTUGAL	
ro_RO	Romanian	ROMANIA	
ru_RU	Russian	RUSSIAN FEDERATION	
ru_UA		UKRAINE	[Support of this locale is level 2]
sd_IN	Sindhi	INDIA	[Support of this locale is level 2] [Output method support is level 2]
sh_YU	Serbo-Croatian	YUGOSLAVIA	
sk_SK	Slovak	SLOVAKIA	
sl_SI	Slovenian	SLOVENIA	
sq_AL	Albanian	ALBANIA	
sr_YU	Serbian	YUGOSLAVIA	
sv_FI	Swedish	FINLAND	
sv_SE		SWEDEN	

ta_IN	Tamil	INDIA	[Support of this locale is level 2] [Output method support is level 2]
te_IN	Telugu	INDIA	[Support of this locale is level 2] [Output method support is level 2]
th_TH	Thai	THAILAND	
tr_TR	Turkish	TURKEY	
uk_UA	Ukrainian	UKRAINE	
ur_IN	Urdu	INDIA	[Support of this locale is level 2] [Output method support is level 2]
vi_VN	Vietnamese	VIETNAM	
zh_CN	Chinese	CHINA	
zh_HK		HONG KONG	
zh_SG		SINGAPORE	[Support of this locale is level 2]
zh_TW		TAIWAN	

Conforming implementations shall make at least UTF-8 coded character set usable under the above locale environments. Conforming implementations also may make other coded character sets, including the following codesets, usable under some of the above locale environments.

ISO/IEC 8859-1
ISO/IEC 8859-2
ISO/IEC 8859-5
ISO/IEC 8859-7
ISO/IEC 8859-9
ISO/IEC 8859-13
ISO/IEC 8859-15

Korean EUC
Japanese EUC
Simplified Chinese EUC
Traditional Chinese EUC

If an implementation supports non UTF-8 codesets, the implementation shall support codeset conversions between the supported codesets and UTF-8 (for both directions) by **iconv** utility and iconv family functions (**iconv()**, **iconv_open()** and **iconv_close()**).

Annex C (Normative): Publicly Available Specification

C.1 gettext message handling functions

NAME

`gettext`, `dgettext`, `ngettext`, `dngettext`, `dcgettext`, `dcngettext`, `textdomain`, `bindtextdomain`, `bind_textdomain_codeset` — message handling functions

SYNOPSIS

```
#include <libintl.h>

char *gettext(const char *msgid);
char *dgettext(const char *domainname, const char *msgid);
char *ngettext(const char *msgid1, const char *msgid2, unsigned long
int n);
char *dngettext(const char *domainname, const char *msgid1, const char
*msgid2, unsigned long int n);
char *textdomain(const char *domainname);
char *bindtextdomain(const char *domainname, const char *dirname);
char  *bind_textdomain_codeset(const  char  *domainname,  const  char
*codeset);

#include <libintl.h>
#include <locale.h>

char  *dcgettext(const  char  *domainname,  const  char  *msgid,  int
category);
char *dcngettext(const char *domainname, const char *msgid1, const char
*msgid2, unsigned long int n, int category);
```

DESCRIPTION

The *gettext*(), *dgettext*(), and *dcgettext*() functions attempt to retrieve a target string based on the specified *msgid* argument within the context of a specific domain and the current locale. The length of strings returned by *gettext*(), *dgettext*(), and *dcgettext*() is undetermined until the function is called. The *msgid* argument is a null-terminated string.

The *ngettext*(), *dngettext*() and *dcngettext*() functions are equivalent to *gettext*(), *dgettext*() and *dcgettext*(), respectively, except for the handling of plural forms. The *ngettext*(), *dngettext*() and *dcngettext*() searches for the message string using the *msgid1* argument as the key, using the argument *n* to determine the plural form. If no message catalogs are found, *msgid1* is returned if *n* `== 1`, otherwise *msgid2* is returned.

The **LANGUAGE** environment variable is examined first to determine the message catalogs to be used. The value of the **LANGUAGE** environment variable is a list of locale names separated by colon (`:`) character. If the **LANGUAGE** environment variable is defined, each locale name is tried in the specified order and if a message catalog containing the requested message is found, the message is returned. If the **LANGUAGE** environment variable is defined but failed to locate a message catalog, the *msgid* string will be returned.

If the **LANGUAGE** environment variable is not defined, **LC_ALL**, **LC_*xxx*** and **LANG** environment variables are examined to locate the message catalog, following the convention used by the *setlocale*() function.

The pathname used to locate the message catalog is *dirname*/*locale*/*category*/*domainname*`.mo`, where *dirname* is the directory specified by *bindtextdomain*(), *locale* is a locale name determined by the definition of environment variables, *category* is LC_MESSAGES if *gettext*(), *ngettext*(), *dgettext*() or *dngettext*() is called, otherwise LC_*xxx* where the name is the same as the locale category name specified by the *category* argument of *dcgettext*() or *dcngettext*(). *domainname* is the name of the domain specified by *textdomain*() or the *domainname* argument of *dgettext*(), *dngettext*(), *dcgettext*() or *dcngettext*().

For *gettext*() and *ngettext*(), the domain used is set by the last valid call to *textdomain*(). If a valid call to *textdomain*() has not been made, the default domain (called `messages`) is used.

For *dgettext*(), *dngettext*(), *dcgettext*() and *dcngettext*(), the domain used is specified by the *domainname* argument. The *domainname* argument is equivalent in syntax and meaning to the *domainname* argument to *textdomain*(), except that the selection of the domain is valid only for the duration of the *dgettext*(), *dngettext*(), *dcgettext*() or *dcngettext*() function call.

The *dcgettext*() and *dcngettext*() functions require additional argument *category* for retrieving message string for other than LC_MESSAGES category. Available value for the category argument are LC_CTYPE, LC_COLLATE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC and LC_TIME (the call of *dcgettext*(*domainname*, *msgid*, LC_MESSAGES) is equivalent to *dgettext*(*domainname*, *msgid*)). Note that LC_ALL must not be used.

The *textdomain*() function sets or queries the name of the current domain of the active LC_MESSAGES locale category. The *domainname* argument is a null-terminated string that can contain only the characters allowed in legal filenames.

The *domainname* argument is the unique name of a domain on the system. If there are multiple versions of the same domain on one system, namespace collisions can be avoided by using *bindtextdomain*(). If *textdomain*() is not called, a default domain is selected. The setting of domain made by the last valid call to *textdomain*() remains valid across subsequent calls to *setlocale*(), and *gettext*().

The *domainname* argument is applied to the currently active LC_MESSAGES locale.

The current setting of the domain can be queried without affecting the current state of the domain by calling *textdomain*() with *domainname* set to the null pointer. Calling *textdomain*() with a *domainname* argument of a null string sets the domain to the default domain (`messages`).

The *bindtextdomain*() function binds the path predicate for a message domain *domainname* to the value contained in *dirname*. If *domainname* is a non-empty string and has not been bound previously, *bindtextdomain*() binds *domainname* with *dirname*.

If *domainname* is a non-empty string and has been bound previously, *bindtextdomain*() replaces the old binding with *dirname*. The *dirname* argument can be an absolute or relative pathname being resolved when *gettext*(), *ngettext*(), *dgettext*(), *dngettext*(), *dcgettext*(), or *dcngettext*() are called. If *domainname* is a null pointer or an empty string, *bindtextdomain*() returns null pointer. If *bindtextdomain*() is not called, implementation-defined default directory is used.

The *bind_textdomain_codeset*() function can be used to specify the output codeset for message catalogs for domain *domainname*. The *codeset* argument must be a valid codeset name which can be used for the *iconv_open*() function, or a null pointer.

If the *codeset* argument is the null pointer, *bind_textdomain_codeset*() returns the currently selected codeset for the domain with the name *domainname*. It returns null pointer if no codeset has yet been selected.

The *bind_textdomain_codeset*() function can be used several times. If used multiple times, with the same *domainname* argument, the later call overrides the settings made by the earlier one.

The *bind_textdomain_codeset*() function returns a pointer to a string containing the name of the selected codeset. The string is allocated internally in the function and must not be changed by the user.

RETURN VALUE

The *gettext*(), *dgettext*() and *dcgettext*() functions return the message string if the search succeeds, otherwise return the *msgid* string.

The *ngettext*(), *dngettext*() and *dcngettext*() functions return the message string if the search succeeds. If the search fails, *msgid1* is returned if *n* `== 1`, otherwise *msgid2* is returned.

The *textdomain*() function returns the currently selected domain. If it fails, null pointer will be returned.

The *bindtextdomain*() function returns the directory pathname currently bound to the domain. If it fails, null pointer will be returned.

The *bind_textdomain_codeset*() function returns the currently selected codeset name. It returns null pointer if no codeset has yet been selected.

ERRORS

The *gettext*(), *dgettext*(), *dcgettext*(), *ngettext*(), *dngettext*() and *dcngettext*() will not modify the external variable *errno*.

The *textdomain*(), *bindtextdomain*() and *bind_textdomain_codeset*() functions may fail if:

[**ENOMEM**]

Insufficient memory available.

EXAMPLES

None.

APPLICATION USAGE

Application programs shall not modify strings returned by the functions.

The *dcgettext*() function can be used, for example, to retrieve locale-specific string for time format which depends on LC_TIME category, not LC_MESSAGES category. Because the locale setting of LC_TIME and LC_MESSAGES can be different, using *gettext*() in such a case may cause unexpected result.

Specifying relative pathname to the *bindtextdomain*() function may cause trouble and should be avoided. Since the message catalogs are always searched for the directory relative to the application program's current working directory, if the program calls the *chdir*() function, the directory searched for will also be changed.

On Solaris systems, the domain names that begin with the string `SYS_` are reserved for system use. On glibc 2.2, the name `libc` is used for libc messages. Such domain names shall not be used by application programs.

FUTURE DIRECTIONS

None.

C.2 <libintl.h> header

NAME

`libintl.h` — internationalized message handling

SYNOPSIS

```
#include <libintl.h>
```

DESCRIPTION

The following are declared as functions and may also be defined as macros. Function prototypes must be provided for use with an ISO C compiler.

```
char *gettext(const char *msgid);
char *dgettext(const char *domainname, const char *msgid);
char *ngettext(const char *msgid1, const char *msgid2, unsigned long
int n);
char *dngettext(const char *domainname, const char *msgid1, const char
*msgid2, unsigned long int n);
char *textdomain(const char *domainname);
char *bindtextdomain(const char *domainname, const char *dirname);
char *bind_textdomain_codeset(const char *domainname, const char
*codeset);
char *dcgettext(const char *domainname, const char *msgid, int
category);
char *dcngettext(const char *domainname, const char *msgid1, const char
*msgid2, unsigned long int n, int category);
```

APPLICATION USAGE

None.

FUTURE DIRECTIONS

None.

C.3 msgfmt utility

NAME

`msgfmt` — create a message object from a message file

SYNOPSIS

`msgfmt` **[** *`options`* **]** *`filename.po`* `...`

DESCRIPTION

`msgfmt` creates message object files from portable object files (*filename.po*), without changing the portable object files.

The `.po` file contains messages displayed to users by system utilities or by application programs. `.po` files can be edited, and the messages in them can be rewritten in any language supported by the system.

If input file is `-`, standard input is read.

The `xgettext` utility can be used to create `.po` files from script or programs.

`msgfmt` interprets data as characters according to the current setting of the LC_CTYPE locale category.

OPTIONS

`-D` *`directory`*

`--directory=`*`directory`*

Add directory to list for input files search.

`-f`

`--use-fuzzy`

Use `fuzzy` entries in output. If this option is not specified, `fuzzy` entries are not included into the output.

`-o` *`output-file`*

`--output-file=`*`output-file`*

Specify output file name as *`output-file`*. All `domain` directives and duplicate `msgids` in the `.po` file are ignored. If *`output-file`* is `-`, output is written to standard output.

`--strict`

Direct the utility to work strictly following the UniForum/Sun implementation. Currently this only affects the naming of the output file. If this option is not given the name of the output file is the same as the domain name. If the strict UniForum mode is enabled the suffix `.mo` is added to the file name if it is not already present.

`-v`

`--verbose`

Detect and diagnose input file anomalies which might represent translation errors. The `msgid` and `msgstr` strings are studied and compared. It is considered abnormal that one string starts or ends with a newline while the other does not.
Also, if the string represents a format string used in a *printf*-like function both strings should have the same number of `%` format specifiers, with matching types. If the flag `c-format` or `possible-c-format` appears in the special comment `#,` for this entry a check is performed. For example, the check will diagnose using `%.*s` against `%s`, or `%d` against `%s`, or `%d` against `%x`. It can even handle positional parameters.

OPERANDS

The *filename.po* operands are treated as portable object files. The format of portable object files is defined in *EXTENDED DESCRIPTION*.

STDIN

The standard input is not used unless a *filename.po* operand is specified as "`-`".

INPUT FILES

Input files are text files.

ENVIRONMENT VARIABLES

LANGUAGE

Specifies one or more locale names. See *C.1 gettext message handling functions* for more information.

LANG

Specifies default locale name.

LC_ALL

Specifies locale name for all categories. If defined, overrides **LANG**, **LC_CTYPE** and **LC_MESSAGES**.

LC_CTYPE

Specifies locale name for character handling.

LC_MESSAGES

Specifies messaging locale, and if present overrides **LANG** for messages.

STDOUT

The standard output is not used unless the option-argument of the `-o` option is specified as `-`.

STDERR

The standard error is used only for diagnostic messages.

OUTPUT FILES

The format of output files are not specified in this specification.

EXTENDED DESCRIPTION

The format of portable object files (`.po` files) is defined as follows. Each `.po` file contains one or more lines, with each line containing either a comment or a statement. Comments start the line with a hash mark (#) and end with the newline character. All comments and empty lines are ignored. The format of a statement is:

`directive value`

Each *`directive`* starts at the beginning of the line and is separated from value by white space (such as one or more space or tab characters). *`value`* consists of one or more quoted strings separated by white space. If two or more strings are specified as *`value`*, they are normalized into single string using the string normalization syntax the same as the ISO C language. Use any of the following types of directives:

`domain` *`domainname`*

`msgid` *`message_identifier`*

`msgid_plural` *`untranslated_string_plural`*

`msgstr` *`message_string`*

`msgstr[`*`n`*`]` *`message_string`*

The behavior of the `domain` directive is affected by the options used. See *OPTIONS* for the behavior when the `-o` option is specified. If the `-o` option is not specified, the behavior of the `domain` directive is as follows:

- All `msgid`s from the beginning of each `.po` file to the first `domain` directive are put into a default message object file, `messages` (or `messages.mo` if `--strict` option is specified).
- When `msgfmt` encounters a `domain` *`domainname`* directive in the `.po` file, all following `msgid`s until the next `domain` directive are put into the message object file *`domainname`* (or *`domainname`*`.mo` if `--strict` option is specified).
- Duplicate `msgid`s are defined in the scope of each domain. That is, a `msgid` is considered a duplicate only if the identical `msgid` exists in the same domain.
- All duplicate `msgid`s are ignored.

The `msgid` directive specifies the value of a message identifier associated with the directive that follows it. The `msgid_plural` directive specifies the plural form message specified to the plural message handling functions *ngettext*(), *dngettext*() or *dcngettext*(). The *`message_identifier`* string identifies a target string to be used at retrieval time. Each statement containing a `msgid` directive must be followed by a statement containing a `msgstr` directive or `msgstr[`*`n`*`]` directives.

The `msgstr` directive specifies the target string associated with the `message_identifier` string declared in the immediately preceding `msgid` directive.

The `msgstr[`*n*`]` (where *n* = `0, 1, 2, ...`) directive specifies the target string to be used with plural form handling functions *ngettext*(), *dngettext*() and *dcngettext*().

Message strings can contain the escape sequences `\n` for newline, `\t` for tab, `\v` for vertical tab, `\b` for backspace, `\r` for carriage return, `\f` for formfeed, `\\` for backslash, `\"` for double quote, `\`*ddd* for octal bit pattern, and `\x`*HH* for hexadecimal bit pattern.

Comments should be in one of the following formats:

```
#  translator-comments
#. automatic-comments
#: reference...
#, flag
```

The comments that starts with `#.` and `#:` are automatically generated by `xgettext` utility. The `#:` comments indicate the location of the `msgid` string in the source files in *`filename`*`:`*`line`* format. The `#.` comments are generated when `-c` option of the `xgettext` utility is specified. These comments are informative only and silently ignored by the `msgfmt` utility.

The `#,` comments requires one or more *`flags`* separated by comma (`,`) character. The following flags can be specified:

`fuzzy`

This flag can be generated by the `msgmerge` utility or can be inserted by the translator. It shows that the `msgstr` string might not be a correct translation (anymore). Only the translator can judge if the translation requires further modification, or is acceptable as is. Once satisfied with the translation, the translator then removes this `fuzzy` flag. The `msgmerge` programs inserts this when it combined the `msgid` and `msgstr` entries after fuzzy search only.

If this flag is specified, the `msgfmt` utility will not generate the entry for the immediately following `msgid` in the output message catalog.

`c-format`

`no-c-format`

The flags are automatically added by the `xgettext` utility and they should not be added manually. The `c-format` flag indicates that the `msgid` string is used as format string by *printf*-like functions. In case the `c-format` flag is given for a string the `msgfmt` utility does some more tests to check to validity of the translation.

The `msgid` entry with empty string (`""`) is called the *header entry* and treated specially. If the message string for the header entry contains `nplurals=`*`value`*, the *value* indicates the number of plural forms. For example, if `nplurals=4`, there are 4 plural forms. If `nplurals` is defined, there should be `plural=`*`expression`* in the same line, separated by a semicolon (`;`) character. The *`expression`* is a C language expression to determine which version of `msgstr[`*`n`*`]` to be used based on the value of n, the last argument of *ngettext*(), *dngettext*() or *dcngettext*(). For example:

```
    nplurals=2; plural=n == 1 ? 0 : 1
```

indicates that there are 2 plural forms in the language; `msgstr[0]` is used if `n == 1`, otherwise `msgstr[1]` is used. Another example:

```
    nplurals=3; plural=n==1 ? 0 : n==2 ? 1 : 2
```

indicates that there are 3 plural forms in the language; `msgstr[0]` is used if `n == 1`, `msgstr[1]` is used if `n == 2`, otherwise `msgstr[2]` is used.

If the *header entry* contains `charset=`*`codeset`* string, the *`codeset`* is used to indicate the codeset to be used to encode the message strings. If the output string's codeset is different from the message string's codeset, codeset conversion from the message string's codeset to the output string's codeset will be performed upon the call of *gettext*(), *dgettext*(), *dcgettext*(), *ngettext*(), *dngettext*() and *dcngettext*(). The output string's codeset is determined by the current locale's codeset (the returned value of *nl_langinfo(CODESET)*) by default, and can be changed by the call of *bind_textdomain_codeset*().

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

APPLICATION USAGE

Neither `msgfmt` nor any *gettext*() routine imposes a limit on the total length of a message. Installing message catalogs under the C locale is pointless, since they are ignored for the sake of efficiency.

EXAMPLES

Example 1: Examples of creating message objects from message files.

In this example `module1.po` and `module2.po` are portable message objects files.

```
example% cat module1.po
# default domain "messages"
msgid "msg 1"
msgstr "msg 1 translation"
#
domain "help_domain"
msgid "help 2"
msgstr "help 2 translation"
```

```
#
domain "error_domain"
msgid "error 3"
msgstr "error 3 translation"
example% cat module2.po
# default domain "messages"
msgid "mesg 4"
msgstr "mesg 4 translation"
#
domain "error_domain"
msgid "error 5"
msgstr "error 5 translation"
#
domain "window_domain"
msgid "window 6"
msgstr "window 6 translation"
```

The following command will produce the output files, `messages`, `help_domain`, and `error_domain`.

```
example% msgfmt module1.po
```

The following command will produce the output files, `messages`, `help_domain`, `error_domain`, and `window_domain`.

```
example% msgfmt module1.po module2.po
```

The following example will produce the output file `hello.mo`.

```
example% msgfmt -o hello.mo module1.po module2.po
```

FUTURE DIRECTIONS

None.

C.4 xgettext utility

NAME

`xgettext` — extract `gettext` call strings from C programs

SYNOPSIS

`xgettext` **[** *`options`* **]** *`filename`* `...`

DESCRIPTION

The `xgettext` utility is used to automate the creation of portable message files (`.po`). A `.po` file contains copies of the C language strings that are found in ISO C source code in *`filename`* or the standard input if `-` is specified on the command line. The `.po` file can be used as input to the `msgfmt` utility, which produces a binary form of the message file that can be used by application during run-time.

`xgettext` writes `msgid` strings from *gettext*() calls in *filename* to the default output file `messages.po`. The default output file name can be changed by `-d` option. `msgid` strings in *dgettext*() calls are written to the output file *`domainname`*`.po` where *`domainname`* is the first parameter to the *dgettext*() call.

By default, `xgettext` creates a `.po` file in the current working directory, and each entry is in the same order the strings are extracted from *filenames*. When the `-p` option is specified, the `.po` file is created in the pathname directory. An existing `.po` file is overwritten.

Duplicate `msgids` are written to the `.po` file as comment lines. When the `-s` option is specified, the `.po` is sorted by the `msgid` string, and all duplicated `msgids` are removed. All `msgstr` directives in the `.po` file are empty unless the `-m` option is used.

OPTIONS

`-a`

`--extract-all`

Extract all strings, not just those found in *gettext*() and *dgettext*() calls. Only one `.po` file is created.

`-c`**[***`comment-tag`***]**

`--add-comments`**[**=*`comment-tag`***]**

The comment block beginning with *`comment-tag`* as the first token of the comment block is added to the output `.po` file as # delimited comments. For multiple domains, `xgettext` directs comments and messages to the prevailing text domain.

`-C`

`--c++`

Recognize C++ style comments.

`-d` *`default-domain`*

`--default-domain=`*`default-domain`*

Rename default output file from `messages.po` to *`default-domain`*`.po`.

The special domain name `-` means to write the output to the standard output.

`-D` *`directory`*

`--directory=`*`directory`*

Change to *`directory`* before beginning to search and scan source files. The resulting `.po` file will be written relative to the original directory, though.

`--debug`

Use the flags `c-format` and `possible-c-format` to show who was responsible for marking a message as a format string. The later form is used if the `xgettext` utility decided, the format form is used if the programmer prescribed it.

By default only the `c-format` form is used. The translator should not have to care about these details.

`-e`

`--no-escape`

Do not use C escapes in output (default).

`-E`

`--escape`

Use C escapes in output if non-ASCII characters are used.

`-f` *`file`*

`--files-from=`*`file`*

Read the names of the input files from *`file`* instead of getting them from the command line. If `-` is specified as *`file`*, the standard input is read.

`-F`

`--sort-by-file`

Sort output by file location.

`--force-po`

Always write output file even if no message is defined.

`-i`

`--indent`

Write the `.po` file using indented style.

`-j`

`--join-existing`

Join messages with existing message files. If a `.po` file does not exist, it is created. If a `.po` file does exist, new messages are appended. Any duplicate `msgids` are commented out in the resulting `.po` file. Domain directives in the existing `.po` file are ignored. Results not guaranteed if the existing message file has been edited.

`-k`**[**`*keywordspec*`**]**

`--keyword`**[**`=`*`keywordspec`***]**

> Specify additional keyword to be looked for (without *`keywordspec`* means not to use default keywords).
>
> If *`keywordspec`* is a C identifier *`id`*, `xgettext` looks for strings in the first argument of each call to the function or macro *`id`*. If *`keywordspec`* is of the form *`id`*`:`*`argnum`*, `xgettext` looks for string in the *`argnum`*th argument of the call. If *`keywordspec`* is of the form *`id`*`:`*`argnum1`*`,`*`argnum2`*, `xgettext` looks for strings in the *`argnum1`*st argument and in the *`argnum2`*nd argument of the call, and treats them as singular/plural variants for a message with plural handling.
>
> The default keywords, which are always looked for if not explicitly disabled, are `gettext`, `dgettext:2`, `dcgettext:2`, `ngettext:1,2`, `dngettext:2,3`, `dcngettext:2,3` and `gettext_noop`.

`-L` *`name`*

`--language=`*`name`*

> Recognize the specified language. Valid values are `C`, `C++`, and `PO`. Otherwise the language is guessed from file extension.

`-m`**[***`prefix`***]**

`--msgstr-prefix`**[**`=`*`prefix`***]**

> Fill in the `msgstr` with *`prefix`*. This is useful for debugging purposes. To make `msgstr` identical to `msgid`, use an empty string (`""`) for *`prefix`*.

`-M`**[***`suffix`***]**

`--msgstr-suffix`**[**`=`*`suffix`***]**

> Fill in the `msgstr` with *`suffix`*. This is useful for debugging purposes.

`-n`

`--add-location`

> Add comment lines to the output file indicating file name and line number in the source file where each extracted string is encountered (default). These lines appear before each `msgid` in the following format:
>
> `#:` *`filename`*`:`*`line`*

`--no-location`

> Do not write `#:` *`filename`*`:`*`line`* lines.

`-o` *`file`*

`--output=`*`file`*

> Write output to the specified file.

`-p` *`pathname`*

`--output-dir=`*`pathname`*

Specify the directory where the output files will be placed. This option overrides the current working directory.

`-s`

`--sort-output`

Generate output sorted by `msgids` with all duplicate `msgids` removed.

`--strict`

Write out strict UniForum conforming PO file.

`-T`

`--trigraphs`

Understand ISO C trigraphs for input.

`-w` *`number`*

`--width=`*`number`*

Limit the output lines to *`number`* columns.

`-x` *`exclude-file`*

`--exclude-file=`*`exclude-file`*

Specify a `.po` file that contains a list of `msgids` that are not to be extracted from the input files. The format of *`exclude-file`* is identical to the `.po` file. However, only the `msgid` directive line in *`exclude-file`* is used. All other lines are simply ignored. The `-x` option can only be used with the `-a` option.

OPERANDS

The operands are pathnames to the C or C++ language source files.

STDIN

The standard input is not used unless a *filename* operand is specified as `-`.

INPUT FILES

The input files are text files.

ENVIRONMENT VARIABLES

LANGUAGE

Specifies one or more locale names. See *C.1 gettext message handling functions* for more information.

LANG

Specifies default locale name.

LC_ALL

Specifies locale name for all categories. If defined, overrides **LANG**, **LC_CTYPE** and **LC_MESSAGES**.

LC_CTYPE

Specifies locale name for character handling.

LC_MESSAGES

Specifies messaging locale, and if present overrides **LANG** for messages.

STDOUT

The standard output is not used unless the option-argument to the `-o` option is specified as `-`.

STDERR

The standard error is used only for diagnostic messages.

OUTPUT FILES

The output files are text files.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

APPLICATION USAGE

`xgettext` is not able to extract cast strings, for example ISO C casts of literal strings to `(const char *)`. This is unnecessary anyway, since the prototypes in `<libintl.h>` already specify this type.

EXAMPLES

None.

FUTURE DIRECTIONS

None.

C.5 msgmerge utility

NAME

`msgmerge` — merge two portable object files

SYNOPSIS

`msgmerge` **[** *`options`* **]** *`def.po ref.po`*

DESCRIPTION

The `msgmerge` utility merges two UniForum style `.po` files together. The *`def.po`* file is an existing PO file with the old translations which will be taken over to the newly created file as long as they still match; comments will be preserved, but extract comments and file positions will be discarded.

The *`ref.po`* file is the last created PO file (generally by `xgettext`), any translations or comments in the file will be discarded, however dot comments (`#.` comments) and file positions (`#:` comments) will be preserved. Where an exact match cannot be found, fuzzy matching is used to produce better results. The results are written to the standard output unless an output file is specified.

OPTIONS

`-D` *`directory`*

`--directory=`*`directory`*

Change to *`directory`* before beginning to search and scan source files. The resulting `.po` file will be written relative to the original directory, though.

`-e`

`--no-escape`

Do not use C escapes in output (default).

`-E`

`--escape`

Use C escapes in output if non-ASCII characters are used.

`--force-po`

Always write output file even if no message is defined.

`-i`

`--indent`

Write the `.po` file using indented style.

`-o` *`file`*

`--output-file=`*`file`*

Write output to the specified file.

`--add-location`

Add comment lines to the output file indicating file name and line number in the source file where each extracted string is encountered (default). These lines appear before each `msgid` in the following format:

```
#: filename:line ...
```

`--no-location`

Do not write `#:` *`filename`*`:`*`line`* lines.

`--strict`

Write out strict UniForum conforming PO file.

`-w` *`number`*

`--width=`*`number`*

Limit the output lines to *`number`* columns.

OPERANDS

The following operands are supported:

`def.po`

The *`def.po`* operand is a pathname of the message portable object file that may have translated text.

`ref.po`

The *`ref.po`* operand is a pathname of the message portable object file newly generated by the `xgettext` utility with modified program source files. This file may contain newly introduced message strings or modified message strings, and the `msgmerge` utility will detect such changes and merge the changes to *`def.po`*.

STDIN

The standard input is not used unless *`def.po`* or *`ref.po`* operand is specified as `-`.

INPUT FILES

The input files are text files.

ENVIRONMENT VARIABLES

LANGUAGE

Specifies one or more locale names. See *C.1 gettext message handling functions* for more information.

LANG

Specifies default locale name.

LC_ALL

Specifies locale name for all categories. If defined, overrides **LANG**, **LC_CTYPE** and **LC_MESSAGES**.

LC_CTYPE

Specifies locale name for character handling.

LC_MESSAGES

Specifies messaging locale, and if present overrides **LANG** for messages.

STDOUT

The standard output is used to write merged result unless `-o` option is specified.

STDERR

The standard error is used only for diagnostic messages.

OUTPUT FILES

The output files are text files.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

APPLICATION USAGE

None.

EXAMPLES

None.

FUTURE DIRECTIONS

None.

C.6 gettext utility

NAME

`gettext` — retrieve text string from message database

SYNOPSIS

`gettext` **[** *`options`* **]** **[** *`textdomain`* **]** *`msgid`*

`gettext -s` **[** *`options`* **]** *`msgid`* `...`

DESCRIPTION

The `gettext` utility retrieves a translated text string corresponding to string *msgid* from a message object generated with `msgfmt` utility.

The message object name is derived from the optional argument *textdomain* if present, otherwise from the **TEXTDOMAIN** environment. If no domain is specified, or if a corresponding string cannot be found, `gettext` prints *msgid*.

Ordinarily *gettext* looks for its message object in *dirname*/*lang*/`LC_MESSAGES` where *dirname* is the implementation-defined default directory and *lang* is the locale name. If present, the **TEXTDOMAINDIR** environment variable replaces the *dirname*.

This utility interprets C escape sequences such as `\t` for tab. Use `\\` to print a backslash. To produce a message on a line of its own, either put a `\n` at the end of *msgid*, or use this command in conjunction with *printf* utility.

When used with the `-s` option the utility behaves like the `echo` utility. But it does not simply copy its arguments to standard output. Instead those messages found in the selected catalog are translated.

OPTIONS

`-d` *`domainname`*

`--domain=`*`domainname`*

Retrieve translated messages from *domainname*.

`-e`

Enable expansion of some escape sequences.

`-n`

Suppress trailing newline.

OPERANDS

The following operands are supported:

textdomain

A domain name used to retrieve the messages.

msgid

A key to retrieve the localized message.

STDIN

Standard input is not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

LANGUAGE

Specifies one or more locale names. See *C.1 gettext message handling functions* for more information.

LANG

Specifies locale name.

LC_MESSAGES

Specifies messaging locale, and if present overrides **LANG** for messages.

TEXTDOMAIN

Specifies the text domain name, which is identical to the message object filename without `.mo` suffix.

TEXTDOMAINDIR

Specifies the pathname to the message database, and if present replaces the implementation-defined default directory.

STDOUT

All messages are written to the standard output.

STDERR

The standard error is used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

APPLICATION USAGE

None.

EXAMPLES

None.

FUTURE DIRECTIONS

None.

Annex D (Informative): Base Components

(This Annex is informative only and does not contain any requirements to the conforming implementations.)

Scope

One of the goals of LI18NUX 2000 specification is maximize internationalized application portability and interpretability among the conforming implementations. However, application may depend on functions and utilities that are not described in the main sections of LI18NUX 2000. Therefore, this annex describes recommended functions and utilities that all of conforming implementation should provide.

Since scope of the LI18NUX 2000 is limited within internationalization functionality, and the functions and utilities included in this annex may not necessarily be internationalized, this annex is provided just as informative, and does not contain any requirement for conformity.

Future version of this annex may/will refer to Linux Standard Base (LSB) specification when the LSB specification becomes available.

A) Conforming implementations are assumed to provide the following interfaces and utilities besides internationalized interfaces and utilities in chapters 3-11.

** System Interfaces:

Conforming implementations are assumed to provide the C functions and headers which are defined in [POSIX.1].

** Commands and Utilities:

[A] **ar**, **at**, **arch**, **arp**

[B] **basename**, **batch**, **bzip2**, **bunzip2**, **bzip2recover**

[C] **cat**, **cd**, **chgrp**, **chmod**, **chown**, **cmp**, **col**, **comm**, **compress**, **cp**, **cpio**, **csplit**, **cut**, **chroot**

[D] **date**, **dd**, **df**, **diff**, **dirname**, **du**, **diff3**, **domainname**

[E] **echo**, **expand**, **expr**

[F] **false**, **file**, **fuser**, **ftp**

[G] **getopts**, **gzip**, **gunzip**, **getconf**

[H] **head**, **hostname**, **hash**

[I] **id**, **ipcrm**, **ipcs**, **ifconfig**, **imake**

[J] **join**

[K] **kill**, **killall**

[L] **ln**, **logger**, **logname**, **ls**, **ldd**

[M] **make**, **mkdir**, **mkfifo**, **mv**, **mount**, **m4**, **mailx**, **mkswap**, **mkfs**

[N] **nice**, **nl**, **nohup**, **netstat**, **nslookup**, **newgrp**, **nm**

[O] **od**

[P] **paste**, **patch**, **pathchk**, **printf**, **ps**, **pwd**, **ping**

[R] **read**, **renice**, **rm**, **rmdir**, **reboot**

[S] **sleep**, **split**, **strings**, **strip**, **sum**, **shar**, **su**, **shutdown**

[T] **tail**, **tar**, **tee**, **test**, **time**, **touch**, **tr**, **true**, **tty**, **type**, **telnet**, **talk**, **tput**, **tsort**

[U] **umask**, **uname**, **uncompress**, **unexpand**, **uniq**, **uudecode**, **uuencode**, **umount**

[W] **wait**, **wc**, **who**

[X] **xargs**

[Z] **zcat**

B) Furthermore, conforming implementations should support the following utilities and protocols.

** Commands and Utilities:

[A] **alias**

[B] **bc**, **bg**

[C] **cal**, **crontab**, **clear**, **cancel**, **cflow**, **cksum**, **command**, **ctags**

[D] **fc**

[E] **env**

[F] **fg**

[J] **jobs**

[L] **lex**, **lpr**, **lpq**, **lprm**, **lpc**, **less**

[M] **more**, **mesg**

[P] **passwd**, **pr**

[S] **stty**

[T] **tclsh**

[U] **unalias**, **ulimit**

[W] **wish**, **write**

[Y] **yacc**

** Protocols:

Conforming implementations are assumed to support the protocols which are defined in the following RFC specifications (`http://www.rfc-editor.org/`):

- ICMP (Internet Control Message Protocol): RFCs 792 and 950

- SMTP (Simple Mail Transfer Protocol): RFCs 821, 822, 1123 and 2045-2049

- FTP (File Transfer Protocol): RFCs 959, 2228 and 2640
- TELNET: RFCs 854, 855, 856, 857, 858, 859, 860 and 861
- DNS (Domain Naming System): RFCs 974, 1034, 1035, 1101, 1183, 1706, 1982, 1995, 1996, 2136, 2137, 2181, 2308 and 2535
- LPD (Line Printer Daemon Protocol): RFC 1179
- POP3 (Post Office Protocol - Version 3): RFCs 1939, 1957 and 2449

Annex E (Informative): Informative References

(This Annex is informative only and does not contain any requirements to the conforming implementations.)

[XNS5.2]

The Single UNIX Specification, Version 2
Networking Services, Issue 5.2
(The Open Group CAE Specification C808)

[Unicode Normalization]

Unicode Technical Report #15: Unicode Normalization Forms, Revision 18.0
`http://www.unicode.org/unicode/reports/tr15/tr15-18.html`
(included in "The Unicode Standard, Version 3.0")

[Line Breaking Properties]

Unicode Technical Report #14: Line Breaking Properties, Version 6.0
`http://www.unicode.org/unicode/reports/tr14/tr14-6.html`

[Unicode Newline Guidelines]

Unicode Technical Report #13: Unicode Newline Guidelines, Version 5.0
`http://www.unicode.org/unicode/reports/tr13/tr13-5.html`

[East Asian Width]

Unicode Technical Report #11: East Asian Width, Version 5.0
`http://www.unicode.org/unicode/reports/tr11/tr11-5.html`

[Bidirectional Algorithm]

Unicode Technical Report #9: The Bidirectional Algorithm, Version 6.0
`http://www.unicode.org/unicode/reports/tr9/tr9-6.html`

[ISO 639-2]

ISO 639-2:1998 Codes for the representation of names of languages — Part 2: Alpha-3 code

[ISO 3166-2]

ISO 3166-2:1998 Codes for the representation of names of countries and their subdivisions — Part 2: Country subdivision code

[ISO 3166-3]

ISO 3166-3:1999 Codes for the representation of names of countries and their subdivisions — Part 3: Code for formerly used names of countries

[ISO 2022]

ISO/IEC 2022:1994 Information technology — Character code structure and extension techniques

ISO/IEC 2022:1994/Cor 1:1999

[ISO 6429]

ISO/IEC 6429:1992 Information technology — Control functions for coded character sets

[ISO 646]

ISO/IEC 646:1991 Information technology — ISO 7-bit coded character set for information interchange

[ISO 6937]

ISO/IEC 6937:1994 Information technology — Coded graphic character set for text communication — Latin alphabet

[ISO 8859-3]

ISO/IEC 8859-3:1999 Information technology — 8-bit single-byte coded graphic character sets — Part 3: Latin alphabet No. 3

[ISO 8859-4]

ISO/IEC 8859-4:1998 Information technology — 8-bit single-byte coded graphic character sets — Part 4: Latin alphabet No. 4

[ISO 8859-6]

ISO/IEC 8859-6:1999 Information technology — 8-bit single-byte coded graphic character sets — Part 6: Latin/Arabic alphabet

[ISO 8859-8]

ISO/IEC 8859-8:1999 Information technology — 8-bit single-byte coded graphic character sets — Part 8: Latin/Hebrew alphabet

[ISO 8859-10]

ISO/IEC 8859-10:1998 Information technology — 8-bit single-byte coded graphic character sets — Part 10: Latin alphabet No. 6

[ISO 8859-14]

ISO/IEC 8859-14:1998 Information technology — 8-bit single-byte coded graphic character sets — Part 14: Latin alphabet No. 8 (Celtic)

[Tcl/Tk 8.3]

Tcl/Tk 8.3 (February 10, 2000)
`http://dev.scriptics.com/software/tcltk/8.3.html`

[PPP-I18N]

RFC 2484 PPP LCP Internationalization Configuration Option. G. Zorn. January 1999. (Format: TXT=8330 bytes) (Updates RFC2284, RFC1994, RFC1570) (Status: PROPOSED STANDARD)

[IETF-Charset]

RFC 2277 IETF Policy on Character Sets and Languages. H. Alvestrand. January 1998. (Format: TXT=16622 bytes) (Also BCP0018) (Status: BEST CURRENT PRACTICE)

[IANA-Charset]

RFC 2278 IANA Charset Registration Procedures. N. Freed, J. Postel. January 1998. (Format: TXT=18881 bytes) (Also BCP0019) (Status: BEST CURRENT PRACTICE)

[MIME-Parameter]

RFC 2231 MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations. N. Freed, K. Moore. November 1997. (Format: TXT=19280 bytes) (Obsoletes RFC2184) (Updates RFC2045, RFC2047 RFC2183) (Status: PROPOSED STANDARD)

[RFC 2130]

RFC 2130 The Report of the IAB Character Set Workshop held 29 February - 1 March, 1996. C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin, P. Svanberg. April 1997. (Format: TXT=63443 bytes) (Status: INFORMATIONAL)

[HTML 4.01]

HTML 4.01 Specification
24 December 1999. Dave Raggett, Arnaud Le Hors, Ian Jacobs
`http://www.w3.org/TR/html401`

This specification is the latest version of HTML 4. It supersedes the HTML 4.0 Recommendation first published as HTML 4.0 on 18 December 1997 and revised as HTML 4.0 on 24 April 1998.

[MIME]

RFC 2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. N. Freed & N. Borenstein. November 1996. (Format: TXT=72932 bytes) (Obsoletes RFC1521, RFC1522, RFC1590)
(Updated by RFC2184, RFC2231) (Status: DRAFT STANDARD)

RFC 2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. N. Freed & N. Borenstein. November 1996. (Format: TXT=105854 bytes) (Obsoletes RFC1521, RFC1522, RFC1590) (Updated by RFC2646) (Status: DRAFT STANDARD)

RFC 2047 MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text. K. Moore. November 1996. (Format: TXT=33262 bytes) (Obsoletes RFC1521, RFC1522, RFC1590)
(Updated by RFC2184, RFC2231) (Status: DRAFT STANDARD)

RFC 2048 Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures. N. Freed, J. Klensin & J. Postel. November 1996. (Format: TXT=45033 bytes) (Obsoletes RFC1521, RFC1522, RFC1590) (Also BCP0013) (Status: BEST CURRENT PRACTICE)

RFC 2049 Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples. N. Freed & N. Borenstein. November 1996. (Format: TXT=51207 bytes) (Obsoletes RFC1521, RFC1522, RFC1590) (Status: DRAFT STANDARD)

[LANG-TAG]

RFC 1766 Tags for the Identification of Languages. H. Alvestrand. March 1995. (Format: TXT=16966 bytes) (Status: PROPOSED STANDARD)

[MIME-BIDI]

RFC 1556 Handling of Bi-directional Texts in MIME. H. Nussbacher. December 1993. (Format: TXT=5602 bytes) (Status: INFORMATIONAL)

[Unicode-Language-tag]

RFC 2482 Language Tagging in Unicode Plain Text. K. Whistler, G. Adams.
January 1999. (Format: TXT=27800 bytes) (Status: INFORMATIONAL)

[UTF-16]

RFC 2781 UTF-16, an encoding of ISO 10646. P. Hoffman, F. Yergeau.
February 2000. (Format: TXT=29870 bytes) (Status: INFORMATIONAL)

[KOI8-U]

RFC 2319 Ukrainian Character Set KOI8-U. KOI8-U Working Group. April 1998. (Format: TXT=18042 bytes) (Status: INFORMATIONAL)

[UTF-8]

RFC 2279 UTF-8, a transformation format of ISO 10646. F. Yergeau. January 1998. (Format: TXT=21634 bytes) (Obsoletes RFC2044) (Status: DRAFT STANDARD)

[ISO-2022-JP-1]

RFC 2237 Japanese Character Encoding for Internet Messages. K. Tamaru. November 1997. (Format: TXT=11628 bytes) (Status: INFORMATIONAL)

[UTF-7]

RFC 2152 UTF-7 A Mail-Safe Transformation Format of Unicode. D. Goldsmith, M. Davis. May 1997. (Format: TXT=28065 bytes) (Obsoletes RFC1642) (Status: INFORMATIONAL)

[ISO-8859-7]

RFC 1947 Greek Character Encoding for Electronic Mail Messages. D. Spinellis. May 1996. (Format: TXT=14428 bytes) (Status: INFORMATIONAL)

[ISO-2022-CN]

RFC 1922 Chinese Character Encoding for Internet Messages. HF. Zhu, DY. Hu, ZG. Wang, TC. Kao, WCH. Chang & M. Crispin. March 1996. (Format: TXT=50995 bytes) (Status: INFORMATIONAL)

[HZ-GB-2312]

RFC 1842 ASCII Printable Characters-Based Chinese Character Encoding for Internet Messages. Y. Wei, Y. Zhang, J. Li, J. Ding, Y. Jiang. August 1995. (Format: TXT=24143 bytes) (Status: INFORMATIONAL)

[HZ]

RFC 1843 HZ - A Data Format for Exchanging Files of Arbitrarily Mixed Chinese and ASCII characters. F. Lee. August 1995. (Format: TXT=8787 bytes) (Status: INFORMATIONAL)

[ISO-10646-J-1]

RFC 1815 Character Sets ISO-10646 and ISO-10646-J-1. M. Ohta. July 1995. (Format: TXT=11823 bytes) (Status: INFORMATIONAL)

[ISO-2022-KR]

RFC 1557 Korean Character Encoding for Internet Messages. U. Choi, K. Chon, H. Park. December 1993. (Format: TXT=8736 bytes) (Status: INFORMATIONAL)

[ISO-2022-JP-2]

RFC 1554 ISO-2022-JP-2: Multilingual Extension of ISO-2022-JP. M. Ohta, K. Handa. December 1993. (Format: TXT=11449 bytes) (Status: INFORMATIONAL)

[ISO-8859-8]

RFC 1555 Hebrew Character Encoding for Internet Messages. H. Nussbacher, Y. Bourvine. December 1993. (Format: TXT=9273 bytes) (Status: INFORMATIONAL)

[KOI8-R]

RFC 1489 Registration of a Cyrillic Character Set. A. Chernov. July 1993.
(Format: TXT=7798 bytes) (Status: INFORMATIONAL)

[ISO-2022-JP]

RFC 1468 Japanese Character Encoding for Internet Messages. J. Murai, M. Crispin, E. van der Poel. June 1993. (Format: TXT=10970 bytes) (Status: INFORMATIONAL)

[VISCII]

RFC 1456 Conventions for Encoding the Vietnamese Language VISCII: VIetnamese Standard Code for Information Interchange VIQR: VIetnamese Quoted-Readable Specification. Vietnamese Standardization Working Group. May 1993. (Format: TXT=14775 bytes) (Status: INFORMATIONAL)

[WWW-Charmodel]

Character Model for the World Wide Web
29 November 1999, Martin J. Dürst, François Yergeau
`http://www.w3.org/TR/1999/WD-charmod-19991129`

[Unicode-Markup]

Unicode in XML and other Markup Languages
28 September 1999, Martin Dürst, Mark Davis, Hideki Hiura, Asmus Freytag
`http://www.w3.org/TR/1999/WD-unicode-xml-19990928`

[MAIL-I18N]

Using International Characters in Internet Mail
prepared by Internet Mail Consortium

Internet Mail Consortium Report: MAIL-I18N

IMCR-010, August 1, 1998

`http://www.imc.org/mail-i18n.html`

[PLS]

Portable Layout Services: Context-dependent and Directional Text

The Open Group CAE Specification C616

[DISS2]

Distributed Internationalisation Services, Version 2

The Open Group Snapshot S308

[DIF]

Distributed Internationalisation Framework

The Open Group Snapshot S503

[Mozilla L10N-Spec]

Localization Engineering Check List

`http://www.mozilla.org/projects/intl/l10n_eng_chklist.html`

International Browser Character Coding Menu UI/UE Specifications

`http://www.mozilla.org/projects/intl/uidocs/browsercharmenu.html`

International Editor UI/UE Specifications

`http://www.mozilla.org/projects/intl/uidocs/editorcharmenu.html`

[ICMP]

RFC 0792 Internet Control Message Protocol. J. Postel. Sep-01-1981.

(Format: TXT=30404 bytes) (Obsoletes RFC0777) (Updated by RFC0950)

(Also STD0005) (Status: STANDARD)

RFC 0950 Internet Standard Subnetting Procedure. J.C. Mogul, J. Postel. Aug-01-1985. (Format: TXT=37985 bytes) (Updates RFC0792) (Also STD0005) (Status: STANDARD)

[SMTP]

RFC 0821 Simple Mail Transfer Protocol. J. Postel. Aug-01-1982. (Format: TXT=124482 bytes) (Obsoletes RFC0788) (Also STD0010) (Status: STANDARD)

[RFC 822]

RFC 0822 Standard for the format of ARPA Internet text messages. D. Crocker. Aug-13-1982. (Format: TXT=109200 bytes) (Obsoletes RFC0733) (Updated by RFC1123, RFC1138, RFC1148, RFC1327, RFC2156) (Also STD0011) (Status: STANDARD)

[TELNET]

RFC 0854 Telnet Protocol Specification. J. Postel, J.K. Reynolds.
May-01-1983. (Format: TXT=39371 bytes) (Obsoletes RFC0764, NIC 18639)
(Also STD 0008) (Status: STANDARD)

RFC 0855 Telnet Option Specifications. J. Postel, J.K. Reynolds. May-01-1983. (Format: TXT=6218 bytes) (Obsoletes NIC 18640) (Also STD0008) (Status: STANDARD)

RFC 0856 Telnet Binary Transmission. J. Postel, J.K. Reynolds. May-01-1983. (Format: TXT=9192 bytes) (Obsoletes NIC 15389) (Also STD0027) (Status: STANDARD)

RFC 0857 Telnet Echo Option. J. Postel, J.K. Reynolds. May-01-1983.
(Format: TXT=11143 bytes) (Obsoletes NIC 15390) (Also STD0028)
(Status: STANDARD)

RFC 0858 Telnet Suppress Go Ahead Option. J. Postel, J.K. Reynolds. May-01-1983. (Format: TXT=3825 bytes) (Obsoletes NIC 15392) (Also STD0029) (Status: STANDARD)

RFC 0859 Telnet Status Option. J. Postel, J.K. Reynolds. May-01-1983.
(Format: TXT=4443 bytes) (Obsoletes RFC0651) (Also STD0030) (Status: STANDARD)

RFC 0860 Telnet Timing Mark Option. J. Postel, J.K. Reynolds. May-01-1983.
(Format: TXT=8108 bytes) (Obsoletes NIC 16238) (Also STD0031)
(Status: STANDARD)

RFC 0861 Telnet Extended Options: List Option. J. Postel, J.K. Reynolds. May-01-1983. (Format: TXT=3181 bytes) (Obsoletes NIC 16239) (Also STD0032) (Status: STANDARD)

[TELNET-CHARSET]

RFC 2066 TELNET CHARSET Option. R. Gellens. January 1997. (Format:
TXT=26088 bytes) (Status: EXPERIMENTAL)

[FTP]

RFC 0959 File Transfer Protocol. J. Postel, J.K. Reynolds. Oct-01-1985. (Format: TXT=151249 bytes) (Obsoletes RFC0765) (Updated by RFC2228, RFC2640) (Also STD0009) (Status: STANDARD)

[DNS]

RFC 0974 Mail routing and the domain system. C. Partridge. Jan-01-1986.
(Format: TXT=18581 bytes) (Also STD0014) (Status: STANDARD)

RFC 1034 Domain names - concepts and facilities. P.V. Mockapetris. Nov-01-1987. (Format: TXT=129180 bytes) (Obsoletes RFC0973, RFC0882, RFC0883) (Obsoleted by RFC1065, RFC2308) (Updated by RFC1101, RFC1183, RFC1348, RFC1876, RFC1982, RFC2065, RFC2181, RFC2308, RFC2535) (Also STD0013) (Status: STANDARD)

RFC 1035 Domain names - implementation and specification. P.V. Mockapetris. Nov-01-1987. (Format: TXT=125626 bytes) (Obsoletes RFC0973, RFC0882, RFC0883) (Updated by RFC1101, RFC1183, RFC1348, RFC1876, RFC1982, RFC1995, RFC1996, RFC2065, RFC2181, RFC2136, RFC2137, RFC2308, RFC2535) (Also STD0013) (Status: STANDARD)

[LPD]

RFC 1179 Line printer daemon protocol. L. McLaughlin. Aug-01-1990.
(Format: TXT=24324 bytes) (Status: INFORMATIONAL)

[POP3]

RFC 1939 Post Office Protocol - Version 3. J. Myers & M. Rose. May 1996. (Format: TXT=47018 bytes) (Obsoletes RFC1725) (Updated by RFC1957, RFC2449) (Also STD0053) (Status: STANDARD)

[*** POP3 Extensions ***]

RFC 1957 Some Observations on Implementations of the Post Office Protocol (POP3). R. Nelson. June 1996. (Format: TXT=2325 bytes) (Updates RFC1939) (Status: INFORMATIONAL)

RFC 2449 POP3 Extension Mechanism. R. Gellens, C. Newman, L. Lundblade.
November 1998. (Format: TXT=36017 bytes) (Updates RFC1939) (Status:
PROPOSED STANDARD)

[*** RFC 822 Extensions ***]

RFC 1123 Requirements for Internet hosts - application and support. R.T. Braden. Oct-01-1989. (Format: TXT=245503 bytes) (Updates RFC0822) (Updated by RFC2181) (Also STD0003) (Status: STANDARD)

RFC 1138 Mapping between X.400(1988) / ISO 10021 and RFC 822. S.E. Kille. Dec-01-1989. (Format: TXT=191029 bytes) (Obsoleted by RFC1327, RFC1495, RFC2156) (Updates RFC0822, RFC0987, RFC1026) (Updated by RFC1148) (Status: EXPERIMENTAL)

RFC 1148 Mapping between X.400(1988) / ISO 10021 and RFC 822. S.E. Kille. Mar-01-1990. (Format: TXT=194292 bytes) (Obsoleted by RFC1327, RFC1495, RFC2156) (Updates RFC0822, RFC0987, RFC1026, RFC1138) (Status: EXPERIMENTAL)

RFC 1327 Mapping between X.400(1988) / ISO 10021 and RFC 822. S. Hardcastle-Kille. May 1992. (Format: TXT=228598 bytes) (Obsoletes RFC987, RFC1026, RFC1138, RFC1148) (Obsoleted by RFC1495, RFC2156) (Updates RFC0822, RFC0822) (Status: PROPOSED STANDARD)

RFC 2156 MIXER (Mime Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME. S. Kille. January 1998. (Format: TXT=280385 bytes) (Obsoletes RFC0987, RFC1026, RFC1138, RFC1148, RFC1327, RFC1495) (Updates RFC0822) (Status: PROPOSED STANDARD)

[*** FTP Extensions ***]

RFC 2228 FTP Security Extensions. M. Horowitz, S. Lunt. October 1997. (Format: TXT=58733 bytes) (Updates RFC0959) (Status: PROPOSED STANDARD)

[FTP-I18N]

RFC 2640 Internationalization of the File Transfer Protocol. B. Curtin. July 1999. (Format: TXT=57204 bytes) (Updates 959) (Status: PROPOSED STANDARD)

[*** DNS Extensions ***]

RFC 1101 DNS encoding of network names and other types. P.V. Mockapetris. Apr-01-1989. (Format: TXT=28677 bytes) (Updates RFC1034, RFC1035) (Status: UNKNOWN)

RFC 1183 New DNS RR Definitions. C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris. Oct-01-1990. (Format: TXT=23788 bytes) (Updates RFC1034, RFC1035) (Status: EXPERIMENTAL)

RFC 1348 DNS NSAP RRs. B. Manning. July 1992. (Format: TXT=6871 bytes) (Obsoleted by RFC1637) (Updates RFC1034, RFC1035) (Updated by RFC1637) (Status: EXPERIMENTAL)

RFC 1637 DNS NSAP Resource Records. B. Manning, R. Colella. June 1994. (Format: TXT=21768 bytes) (Obsoletes RFC1348) (Obsoleted by RFC1706) (Updates RFC1348) (Status: EXPERIMENTAL)

RFC 1706 DNS NSAP Resource Records. B. Manning, R. Colella. October 1994. (Format: TXT=19721 bytes) (Obsoletes RFC1637) (Status: INFORMATIONAL)

RFC 1876 A Means for Expressing Location Information in the Domain Name System. C. Davis, P. Vixie, T. Goodwin, I. Dickinson. January 1996. (Format: TXT=29631 bytes) (Updates RFC1034, RFC1035) (Status: EXPERIMENTAL)

RFC 1982 Serial Number Arithmetic. R. Elz & R. Bush. August 1996. (Format: TXT=14440 bytes) (Updates RFC1034, RFC1035) (Status: PROPOSED STANDARD)

RFC 1995 Incremental Zone Transfer in DNS. M. Ohta. August 1996. (Format: TXT=16810 bytes) (Updates RFC1035) (Status: PROPOSED STANDARD)

RFC 1996 A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY). P. Vixie. August 1996. (Format: TXT=15247 bytes) (Updates RFC1035) (Status: PROPOSED STANDARD)

RFC 2065 Domain Name System Security Extensions. D. Eastlake, 3rd, C. Kaufman. January 1997. (Format: TXT=97718 bytes) (Obsoleted by RFC2535) (Updates RFC1034, RFC1035) (Status: PROPOSED STANDARD)

RFC 2181 Clarifications to the DNS Specification. R. Elz, R. Bush. July 1997. (Format: TXT=36989 bytes) (Updates RFC1034, RFC1035, RFC1123) (Updated by RFC2535) (Status: PROPOSED STANDARD)

RFC 2136 Dynamic Updates in the Domain Name System (DNS UPDATE). P. Vixie, Ed., S. Thomson, Y. Rekhter, J. Bound. April 1997. (Format: TXT=56354 bytes) (Updates RFC1035) (Status: PROPOSED STANDARD)

RFC 2137 Secure Domain Name System Dynamic Update. D. Eastlake. April 1997. (Format: TXT=24824 bytes) (Updates RFC1035) (Status: PROPOSED STANDARD)

RFC 2308 Negative Caching of DNS Queries (DNS NCACHE). M. Andrews. March 1998. (Format: TXT=41428 bytes) (Obsoletes RFC1034) (Updates RFC1034, RFC1035) (Status: PROPOSED STANDARD)

RFC 2535 Domain Name System Security Extensions. D. Eastlake. March 1999.
(Format: TXT=110958 bytes) (Updates RFC2181, RFC1035, RFC1034)
(Status: PROPOSED STANDARD)

Annex F (Informative): Rationale for exceptions

(This Annex is informative only and does not contain any requirements to the conforming implementations.)

[A]

Impossible to fix due to shortcoming of Standard API

[B]

External specifications referenced in the LI18NUX specification are unstable or evolving.