HP-UX 9.x - 11i Internationalization Features White Paper

Edition 1

Manufacturing Part Number: 5971-2270
E0601

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs, in their present form or with alterations, is expressly prohibited.

Copyright Notices

Publication History

The manual publication date and part number indicate its current edition. The publication date will change when a new edition is released. The manual part number will change when extensive changes are made.

New editions of this manual will incorporate all material updated since the previous edition. For the latest version, see the HP-UX 11i Documentation section on the Web at:

http://docs.hp.com

First Edition: June 2001

Typographic Conventions

This document uses these typographic conventions:

Italics	Indicates a parameter or argument that you must replace with the actual value, such as *ServerName*. Also indicates a manpage reference.
Bold	Indicates words defined for the first time. Also indicates a default value for a flag.
`constant width`	C source code, information that the system displays, file names, and locale names appear in constant width type.

Contents

1. NLS Fileset Organization

Fileset Restructuring For NLS Components [10.0] 20
New NLS Directory Layout [10.0] 22
System V.4 Filesystem Changes [10.0] 23
Alternatives 23
ASE (Asian System Environment) Delivery Restructuring [10.01] 24

2. Encoding Characters

Extended UNIX Codes Support In Commands [10.0, 10.10] 28
Extended UNIX Codes Support In Libraries [10.0] 33
EUC Locales [10.0] 34
Disk Space Requirements 34
Migrating HP-UX 9.x Data 34
Commands 35
dtterm(1) 35
eucset(1) 35
Pacific Rim Encoding Methods [10.0] 36
Traditional Chinese 4-Byte EUC 36
Performance 36
Chinese Code For Data Communication 37
Japanese 3-Byte EUC 37
Invalid Shift-JIS (SJIS) Ranges 38
Euro Support 39
Euro (ISO 8859-15 locales) [11.0 patch, 11i] 39
CDE Support 41
X Windows Support 41
Libraries 41
Codeset Converters 41
LaserJet Printers 41
Euro - ISO 10646/Unicode Support [11i] 42
Commands 44
libc 44
Codeset Converters 45
Impact 46
Unicode Character Set [11.0 Patch, 11i] 47
Unicode Euro Enhancement 49
Size Requirement 49
Performance 50

Contents

Streams PTY Driver [11i] 50
Converting Between Encodings 51
Enhancements to iconv() Converters [10.0] 51
New iconv() Converters 51
iconv Name Aliasing 51
Customizing iconv Tables and Algorithms 52
Table Specific Changes 52
Miscellaneous iconv Changes 53
Unmapped Character Conversion 54
Invertible Conversions 54
Enhancements to iconv(1) and iconv(3) [10.10] 55
Enhancements to iconv(1) and iconv(3) [10.20] 56
Corrected Character Mappings to iconv(1) and iconv(3C) [11.0 patch, 11i] 57
Correction for Simplified Chinese 57
Correction for Traditional Chinese 58
Correction for Japanese 61
Correction for Korean 62

3. Locales

Changes to locales [10.0] 64
New Directory Structure For Locales 64
Locale Source Files 64
Disk Space Requirements 64
Build Change For Archive Internationalized Applications 64
Rationale 65
Alternatives 65
Compatibility 66
Disk Space Requirement 66
ISO Locale Names Supported 66
Alternatives 70
Performance 70
Compatibility 70
Modifications To Locales 71
Summary of Changes 71
Other Locale Changes 74
Changed options for locale(1) 75
Compatibility 75
Character Maps (charmaps) Supported 76

Disk Space Requirements . . . 78
Installing a Default Codeset for the C Locale . . . 78
Alternatives . . . 78
Performance . . . 79
Changes to locales [10.10] . . . 79
Changes to locales [10.20] . . . 80
Changes to locales [10.30] . . . 82
New ISO 8859-15 and UTF-8 locales [11.0 patch, 11i] . . . 82
Changed locale and localedef Commands [11i Version 1.5] . . . 83
Directory Structure Changes . . . 83
Changes to locale Command . . . 84
Changes to localedef Command . . . 84
Compatibility . . . 84
Building Locales for HP-UX 11.x . . . 85
Requirements . . . 85
Compiler . . . 85
Source Files . . . 85
Kernel Configuration (needed for UTF8 locales only) . . . 85
The three locale flavors . . . 86
32-bit locales . . . 86
3.2 64-bit locales . . . 86
3.3 Compatibility 32-bit locales . . . 86
Alternatives . . . 86

4. Asian System Environment (ASE)

ASE - Changes [10.30] . . . 88
Features . . . 88
ASE Common . . . 88
JSE . . . 88
KSE . . . 89
SSE . . . 89
Summary of Changes . . . 89
ASE Common . . . 89
JSE . . . 90
KSE . . . 90
Impact . . . 90
JSE . . . 90
KSE . . . 90

Contents

Compatibility 90
JSE 90
Alternatives 91
JSE 91
KSE 91
ASE - Changes [11.0] 92
JSE (The Japanese System Environment) 94
KSE (Korean System Environment) 95
SSE (Simplified-Chinese System Environment) 95
TSE (Traditional-Chinese System Environment) 95
Enhanced Print Capabilities in ASE [11.0 patch, 11i] 96
Changes Common to all ASEs 96
Japanese System Environment (JSE) 96
Korean System Environment (KSE) 98
Simplified-Chinese System Environment (SSE) 98
Traditional-Chinese System Environment (TSE) 98
ASE - Changes [11i and 11i Version 1.5] 99
New Features 99
Changed Feature 106
Deleted Features 106
Troubleshooting Information 107
Software Availability in Native Languages 109
Command and Library Support [not applicable for 11i] 109

5. Conformance [10.0]

X/Open Portability Guide (XPG4) 112
New / Modified Functions 112
iconv(3C) 112
iconv_close(3C) 112
iconv_open(3C) 112
localedef(1M) 112
strfmon(3C) 115
strptime(3C) 115
Program Messaging 116
LC_MESSAGES Support for Messaging 116
Affirmative/Negative Responses (YESEXPR / NOEXPR) 117
POSIX 1003.1 118
Summary of Change 118

Comparison of POSIX and XPG 118
Performance 119
Alternatives/compatibility 119
Potential Future Directions 119

6. Commands and Libraries

New Commands [10.0, 10.01] 122
dmpxlt(1) 122
genxlt(1) 122
Changed Commands [10.0] 123
adjust(1) 123
gencat(1) 123
nljust(1) 123
nroff(1) 124
sed(1) 124
sort(1) 124
tr(1) 124
Changed Commands [10.01] 126
grep(1), egrep(1), fgrep(1) 126
Changed Commands [10.10] 127
cal(1) 127
eucset(1) 127
nl(1) 128
sh-posix(1) 128
Changed Commands [10.20] 129
findmsg(1) 129
Impact 129
Performance 129
Changed Commands [11.0] 130
spell(1) 130
Status 130
tar(1) 130
Status 130
Alternative 130
Functions and Interfaces [11.0] 131
Internationalization- and Localization-related Changes 131
Obsolete Commands And Interfaces 133
Obsolete Core Commands and libc [10.0] 133

Contents

Compatibility . . . 135
Obsolete HP-UX Proprietary Multi-byte nl_tools_16(3C) Interfaces [10.0] . . . 136
Compatibility . . . 137
Alternatives . . . 137
Obsolete HP-UX Proprietary Multi-byte nl_tools_16(3C) Interfaces [10.30] . . . 137

7. Graphical User Interfaces

HP Common Desktop Environment (CDE) & Motif . . . 140
HP CDE 1.0 [10.10] . . . 140
Printing and Building Help Volumes in Multibyte Locales . . . 140
Accessing Font Aliases for Remote CDE Sessions . . . 140
Changing Languages Between Sessions in CDE . . . 141
HP CDE 2.1 & Motif [11.0] . . . 143
HP CDE 2.1 . . . 143
X/Motif Libraries . . . 143
HP VUE 3.0 . . . 144
Vuelogin 'langSetup' Resource [10.10] . . . 144
Support for New Locales in VUE [10.10] . . . 144
Image Help When Using Localized Environments with HP VUE [10.20] . . . 145
Text Help When Using Roman8 Locales with HP VUE [10.20] . . . 145
X11R4/Motif 1.0/1.1 Application Font Usage [10.20] . . . 145
X Window . . . 147
Functions Marked for Obsolescence [10.0] . . . 147
Changes in Keyboard Functionality [10.0] . . . 148
Rendering International Characters [10.0] . . . 148
Terminal Emulators [10.0] . . . 150
X11 New Bitmap Fonts [10.10] . . . 150
X11 FontServer [10.10] . . . 150
Font Manipulation . . . 151
Charset Encoding Extension . . . 151
Glyph Caching and Scalable Aliases . . . 151
New Rasterizer . . . 151
New Fonts . . . 151
Command Changes . . . 152
X11 FontServer [10.20] . . . 152
TrueType Fonts . . . 152
Administering Charsets for True Type Fonts . . . 152
Wildcard Aliases . . . 153

Compatibility [10.30] 153
X Window System (X11 R6) Run-Time Libraries on Workstations [11.0 patch, 11i] 153

8. Miscellaneous Modifications

Configuration Files [10.0] 156
NLS libc Interfaces [10.0] 157
catopen(3C) 157
getlocale(3C) 157
Impact 158
setlocale(3C) 158
Alternatives 158
nl_langinfo(3C) 158
Multibyte Support Extensions 160
Shells [10.0] 161
Differences between the Bourne and POSIX Shells 161
Recompilation Note [10.0] 161
SD-UX partial Internationalization and Localization [10.20] 161
Archive Internationalized Applications [10.30] 162
Example of sh using CCOPTS and LDOPTS 162
Example Using cc Line Only 162
elm 2.4 [10.30] 163
MIME support (RFC 1521) 163
JIS Support for Japanese email Messages 163
GeoCustoms [10.30] 164
Compatibility 164
Performance 164
Alternatives 164
Obsolescence 164
New supported USB keyboards [10.20 patch, 11.0 patch, 11i] 165
Multibyte Support Extension and Unix98 Support [11i] 166
Stream Orientation 166
Restartable APIs and the Conversion State 167
How to Get MSE/Unix98 Behavior 167
New Interfaces 168
btowc 168
fwide 168
fwprintf, swprintf, wprintf 168

Contents

fwscanf, swscanf, wscanf 168
mbrlen 168
mbrtowc 168
mbsinit 168
mbsrtowcs 168
towctrans 169
vfwprintf, vswprintf, vwprintf 169
wcrtomb 169
wcsrtombs 169
wcsstr 169
wctob 169
wctrans 169
wmemchr, wmemcmp, wmemcpy, wmemmove, wmemset 170
Modified interfaces 170
fprintf, printf, snprintf, sprintf, fscanf, scanf, sscanf 170
fputwc, putwc, putwchar 170
freopen 170
wcschr, wcsrchr 170
Potential Modifications to NLS as of HP-UX 11i 171

A. Bibliography

Glossary

About This Guide

The purpose of this document is to present the changes made to the Native Language Support features in the various releases of HP-UX beginning with release 10.0

Organization

The following table illustrates which changes to Native Language Support features described in this manual have been introduced in which HP-UX release.

All subheadings under listed headings are also included unless otherwise indicated.

Table 1 Document Reference

HP-UX Release	Section
10.0	“Fileset Restructuring For NLS Components [10.0]” on page 20
	“New NLS Directory Layout [10.0]” on page 22
	“System V.4 Filesystem Changes [10.0]” on page 23
	“Extended UNIX Codes Support In Commands [10.0, 10.10]” on page 28
	“Extended UNIX Codes Support In Libraries [10.0]” on page 33
	“EUC Locales [10.0]” on page 34
	“Pacific Rim Encoding Methods [10.0]” on page 36
	“Enhancements to iconv() Converters [10.0]” on page 51
	“Changes to locales [10.0]” on page 64
	“Conformance [10.0]” on page 111
	“New Commands [10.0, 10.01]” on page 122
	“Changed Commands [10.0]” on page 123
	“Obsolete Core Commands and libc [10.0]” on page 133
	“Obsolete HP-UX Proprietary Multi-byte nl_tools_16(3C) Interfaces [10.0]” on page 136
	“Functions Marked for Obsolescence [10.0]” on page 147
	“Changes in Keyboard Functionality [10.0]” on page 148
	“Rendering International Characters [10.0]” on page 148
	“Terminal Emulators [10.0]” on page 150
	“Configuration Files [10.0]” on page 156
	“NLS libc Interfaces [10.0]” on page 157
	“Shells [10.0]” on page 161
	“Recompilation Note [10.0]” on page 161

Table 1 Document Reference

HP-UX Release	Section
10.01	“ASE (Asian System Environment) Delivery Restructuring [10.01]” on page 24
	“New Commands [10.0, 10.01]” on page 122
	“Changed Commands [10.01]” on page 126
10.10	“Extended UNIX Codes Support In Commands [10.0, 10.10]” on page 28
	“Enhancements to iconv(1) and iconv(3) [10.10]” on page 55
	“Changes to locales [10.10]” on page 79
	“Changed Commands [10.10]” on page 127
	“HP CDE 1.0 [10.10]” on page 140
	“Vuelogin 'langSetup' Resource [10.10]” on page 144
	“Support for New Locales in VUE [10.10]” on page 144
	“X11 New Bitmap Fonts [10.10]” on page 150
	“X11 FontServer [10.10]” on page 150
10.20	“Enhancements to iconv(1) and iconv(3) [10.20]” on page 56
	“Changes to locales [10.20]” on page 80
	“Changed locale and localedef Commands [11i Version 1.5]” on page 83
	“Changed Commands [10.20]” on page 129
	“Image Help When Using Localized Environments with HP VUE [10.20]” on page 145
	“Text Help When Using Roman8 Locales with HP VUE [10.20]” on page 145
	“X11R4/Motif 1.0/1.1 Application Font Usage [10.20]” on page 145
	“X11 FontServer [10.20]” on page 152
	“SD-UX partial Internationalization and Localization [10.20]” on page 161
	“New supported USB keyboards [10.20 patch, 11.0 patch, 11i]” on page 165

Table 1 Document Reference

HP-UX Release	Section
10.30	“Changes to locales [10.30]” on page 82
	“ASE - Changes [10.30]” on page 88
	“Obsolete HP-UX Proprietary Multi-byte nl_tools_16(3C) Interfaces [10.30]” on page 137
	“Compatibility [10.30]” on page 153
	“Archive Internationalized Applications [10.30]” on page 162
	“elm 2.4 [10.30]” on page 163
	“GeoCustoms [10.30]” on page 164
11.0	“Euro (ISO 8859-15 locales) [11.0 patch, 11i]” on page 39
	“Unicode Character Set [11.0 Patch, 11i]” on page 47
	“Corrected Character Mappings to iconv(1) and iconv(3C) [11.0 patch, 11i]” on page 57
	“ASE - Changes [11.0]” on page 92
	“Enhanced Print Capabilities in ASE [11.0 patch, 11i]” on page 96
	“Changed Commands [11.0]” on page 130
	“Functions and Interfaces [11.0]” on page 131
	“HP CDE 2.1 & Motif [11.0]” on page 143
	“X Window System (X11 R6) Run-Time Libraries on Workstations [11.0 patch, 11i]” on page 153
	“New supported USB keyboards [10.20 patch, 11.0 patch, 11i]” on page 165

Table 1 Document Reference

HP-UX Release	Section
11i	“Euro (ISO 8859-15 locales) [11.0 patch, 11i]” on page 39
	“Euro - ISO 10646/Unicode Support [11i]” on page 42
	“Unicode Character Set [11.0 Patch, 11i]” on page 47
	“Corrected Character Mappings to iconv(1) and iconv(3C) [11.0 patch, 11i]” on page 57
	“Enhanced Print Capabilities in ASE [11.0 patch, 11i]” on page 96
	“ASE - Changes [11i and 11i Version 1.5]” on page 99
	“X Window System (X11 R6) Run-Time Libraries on Workstations [11.0 patch, 11i]” on page 153
	“New supported USB keyboards [10.20 patch, 11.0 patch, 11i]” on page 165
	“Multibyte Support Extension and Unix98 Support [11i]” on page 166
	“Potential Modifications to NLS as of HP-UX 11i” on page 171
11i Version 1.5	“ASE - Changes [11i and 11i Version 1.5]” on page 99
	“Changed locale and localedef Commands [11i Version 1.5]” on page 83

Suggested Reference

This document is to be used in conjunction with:

Title	Programming for the World: A Guide to Internationalization
Author	Sandra Martin O'Donnell
Publisher	Prentice Hall
Copyright	1994 by PTR Prentice Hall
ISBN Number	0-13-722190-8

1 NLS Fileset Organization

Fileset Restructuring For NLS Components [10.0]

The HP-UX 9.x NLS-CORE fileset has been subdivided into several products/filesets. The ability to select a specific component will save disk space.

Table 1-1 HP-UX 10.0 Fileset

Product/fileset	Description	Details	Audience
International/*language*	language specific locales and methods[a], iconv tables and methods		all users of NLS
OS-Core /CMDS-MIN	core non-language specific locales and methods, iconv tables and methods		all users
OS-Core /CMDS-AUX	NLS commands and auxiliary non-language specific iconv tables	iconv, forder, locale, nljust	all users of NLS
OS-Core /NLS-AUX	NLS commands and sources for the customization of locales, iconv tables, and message catalogs	dmpxlt, dumpmsg, gencat, genxlt, localedef input sources, charmap input sources	localizers and advanced NLS users who desire customization
OS-Core / UX-*abbrev_lang-abbrev_codeset*-MSG	localization message catalogs per language and codeset	*.cat message catalogs	users who want localized messages (if they exist for the desired language and codeset)

Table 1-1 HP-UX 10.0 Fileset

Product/fileset	Description	Details	Audience
`ProgSupport/ PROG-AUX`	NLS commands for the modification of C code source files to use NLS messaging, `portNLS`	`findmsg`, `findstr`, `insertmsg`, `libportnls.a`	programmers who wish to internationalize their sources or programmers who require compatibility with MPE

a. A method is an algorithm used to process specific locales and codesets.

New NLS Directory Layout [10.0]

Due to the growing number of various files within the NLS directory, the NLS files have been restructured and organized.

Table 1-2 NLS Directory Layout

Pathname	Contents
/usr/lib/nls/config	locale config file with new format
/usr/lib/nls/iconv/config.iconv	iconv config file
/usr/lib/nls/iconv/methods	iconv method shared objects
/usr/lib/nls/iconv/tables	iconv tables
/usr/lib/nls/loc/charmaps	charmap source files
/usr/lib/nls/loc/locales	locale shared objects
/usr/lib/nls/loc/methods	locale method shared objects
/usr/lib/nls/loc/src	locale source files
/usr/lib/nls/msg	message catalogs for core HP-UX products[a]
/usr/share/man	man pages for core HP-UX products[b]
/opt/*application*/lib/nls/msg/*locale*	message catalogs for core HP-UX products only per locale

a. Message catalogs for optional products should be placed in /opt/*product*/lib/nls/msg.
b. man pages for optional products should be placed in /opt/*product*/man.

System V.4 Filesystem Changes [10.0]

System V.4 filesystem layout specifies that optional applications are placed in `/opt/`*`product`*. Thus, the message catalogs for the optional products can be placed in `/opt/`*`product`*`/lib/nls/msg/`*`10.0_locale_name`*`/*`. In order to allow message catalogs to be found without making the user modify `NLSPATH` for each optional application used, the application can append the new location of the message catalogs to the user's `NLSPATH`. Here are some simple steps that products can take to modify the user's `NLSPATH`:

- use `getenv(3C)` to retrieve the user's existing `NLSPATH`
- save user's original `NLSPATH`
- append the new message catalog path to `NLSPATH`
- use `putenv(3C)` to install the modified `NLSPATH`
- open the message catalog with `catopen(3C)`
- restore the user's original `NLSPATH`

Alternatives

The application could recommend that the user add the new location of the message catalogs to the `NLSPATH` if the user wants localized messages. However, putting the burden on all users should be avoided as users may have multiple applications in `/opt`, which would greatly increase the number of paths in `NLSPATH`.

ASE (Asian System Environment) Delivery Restructuring [10.01]

For 10.0, the Asian Operating Environments consisted of the combination of a localized bundle from the core media and an ASE bundle from separate pieces of media. For 10.01, new bundles have been developed that integrate the two previous bundles into a single bundle that allows you to install all necessary components via a single bundle. This allows Asian versions of HP-UX to have the same level of operating system integration as European versions had at 10.0. These new structures apply to both the S700 and S800 platforms. The new bundles are:

Series 700:

```
HPUXJpnDT700  Japanese HP-UX Desktop Environment (with JSE)
HPUXJpnRT700  Japanese HP-UX Runtime Environment (with JSE)
HPUXKorDT700  Korean HP-UX Desktop Environment (with KSE)
HPUXKorRT700  Korean HP-UX Runtime Environment (with KSE)
HPUXSChDT700  Simplified Chinese HP-UX Desktop Environment (with SSE)
HPUXSChRT700  Simplified Chinese HP-UX Runtime Environment (with SSE)
HPUXTChDT700  Traditional Chinese HP-UX Desktop Environment (with TSE)
HPUXTChRT700  Traditional Chinese HP-UX Runtime Environment (with TSE)
```

Series 800:

```
HPUXJpnRT800  Japanese HP-UX Runtime without Graphics (with JSE)
HPUXJpnGS800  Japanese HP-UX Runtime with Graphics (with JSE)
HPUXKorRT800  Korean HP-UX Runtime without Graphics (with KSE)
HPUXKorGS800  Korean HP-UX Runtime with Graphics (with KSE)
HPUXSChRT800  Simplified Chinese HP-UX without Graphics (with SSE)
HPUXSChGS800  Simplified Chinese HP-UX with Graphics (with SSE)
HPUXTChRT800  Traditional Chinese HP-UX without Graphics (with TSE)
HPUXTChGS800  Traditional Chinese HP-UX with Graphics (with TSE)
```

The structure of bundles on media will also change from 10.0. At 10.0, the media was structured as follows:

- S700 CORE MEDIA -- All European language Operating Environments and “Asian Operating Environment Bases” for S700 (Included with ABA, ABJ, AB0, AB1, and AB2 media options)

- S800 CORE MEDIA -- All European language Operating Environments and “Asian Operating Environment Bases” for S800 (Included with ABA, ABJ, AB0, AB1, and AB2 media options)
- JSE MEDIA -- Japanese System Environment (JSE) bundles for both S700 and S800 platforms (Included with ABJ media option only)
- KSE MEDIA -- Korean System Environment bundles (KSE) for both S700 and S800 platforms (Included with AB1 media option only)
- SSE MEDIA -- Both Simplified and Traditional Chinese System Environment bundles (SSE/TSE) for S700 and S800 platforms (Included with AB0 and AB2 media options only)

At 10.01 the Asian bundles are delivered as follows:

- S700 CORE MEDIA -- All European language Operating Environments for S700 and all core Asian localized filesets, but no Asian Operating Environment bundles (Included with ABA media option only)
- S700 JSE MEDIA -- All European language Operating Environments for S700, all core Asian localized filesets, and Japanese Operating Environment bundles complete with JSE (Included with ABJ media option only)
- S700 KSE MEDIA -- All European language Operating Environments for S700, all core Asian localized filesets, and Korean Operating Environment bundles complete with KSE (Included with AB1 media option only)
- S700 SSE MEDIA -- All European language Operating Environments for S700, all core Asian localized filesets, and Simplified and Traditional Operating Environment bundles complete with SSE/TSE (Included with AB0 and AB2 media options only)
- S800 CORE MEDIA -- All European language Operating Environments for S800 and all core Asian localized filesets, but no Asian Operating Environment bundles (Included with ABA media option only)
- S800 JSE MEDIA -- All European language Operating Environments for S800, all core Asian localized filesets, and Japanese Operating Environment bundles complete with JSE (Included with ABJ media option only)

- S800 KSE MEDIA -- All European language Operating Environments for S800, all core Asian localized filesets, and Korean Operating Environment bundles complete with KSE (Included with ABK media option only)
- S800 SSE MEDIA -- All European language Operating Environments for S800, all core Asian localized filesets, and Simplified and Traditional Operating Environment bundles complete with SSE/TSE (Included with AB0 and AB2 media options only)

2 Encoding Characters

Extended UNIX Codes Support In Commands [10.0, 10.10]

For alphabets of more than 256 characters, such as Traditional Chinese or Kanji, multi-byte character codes are required. For such languages, several encoding schemes have been defined. Encoding schemes provide a set of rules for parsing a byte stream into a group of coded characters.

The Extended UNIX Code (EUC) is an encoding scheme defined by AT&T USO Pacific, Ltd. which is used for data processing and storage. It is a de facto standard encoding scheme for Asian multi-byte codesets.

The following commands will support full 4-byte EUC:

Table 2-1 Commands Supporting Full 4-Byte EUC

Command Type	Commands Modified For Release	
	10.0	**10.10**
File / Text Manipulation	awk(1) cat(1) cp(1) cut(1) diff(1) ed(1) edit(1) egrep(1) ex(1) expr(1) fgrep(1) find(1) join(1) ln(1) ls(1) mkdir(1) more(1) mv(1) mvdir(1M) page(1) paste(1) rm(1) rmdir(1) sed(1) sort(1) tr(1) uniq(1) vedit(1) vi(1) view(1)	bdiff(1) comm(1) compress(1) csplit(1) deroff(1) diff3(1) diffmk(1) dircmp(1) expand(1) fold(1) head(1) pr(1) split(1) tail(1) tee(1) uncompress(1) unexpand(1) wc(1) xargs(1) zcat(1)

Table 2-1 Commands Supporting Full 4-Byte EUC

Command Type	Commands Modified For Release	
	10.0	**10.10**
File System		chgrp(1) chmod(1) chown(1) df(1M) file(1) fsck(1M) fsdb(1M) link(1M) touch(1) unlink(1M)
File Management		fbackup(1M) frecover(1M)
Document Formatting	adjust(1) col(1) man(1) neqn(1) nroff(1) tbl(1)	
Communication / Networking	eucset(1) mailx(1)	uupick(1) uuto(1) uux(1) write(1)
Conversion	dmpxlt(1) genxlt(1) iconv(1)	
Message Handling	dumpmsg(1) gencat(1)	findmsg(1) findstr(1) insertmsg(1)

Table 2-1 Commands Supporting Full 4-Byte EUC

Command Type	Commands Modified For Release	
	10.0	**10.10**
Program Development		asa(1) tsort(1)
Source Control		admin(1) delta(1) get(1) prs(1) rmchg(1) sact(1) sccs(1) unget(1) val(1) what(1)
Shells[a]	sh-posix(1)	csh(1)
System Administration		getconf(1) quota(1) quotacheck(1M) quotaoff(1M) quotaon(1M) repquota(1M)
Libraries		libxcurses.1

Table 2-1 Commands Supporting Full 4-Byte EUC

Command Type	Commands Modified For Release	
	10.0	**10.10**
Miscellaneous	echo(1) locale(1) localedef(1) make(1) od(1) pax(1) xd(1)	at(1) basename(1) batch(1) calendar(1) crontab(1) date(1) env(1) getopt(1) mesg(1) nice(1) nohup(1) pathchk(1) pwd(1) tabs(1) tput(1) which(1) who(1)
Special Testing[b]	cmp(1) cu(1) lp(1) mt(1) tar(1) tcio(1)	

a. Commands such as `alias(1)`, `cd(1)`, `test(1)`, `unalias(1)` etc. will be supported as a built-in command of the shells.
b. These commands do not require modifications in order to support 4-byte EUC data because they do not manipulate data. However, these commands have been tested with 4-byte EUC data.

Extended UNIX Codes Support In Libraries [10.0]

The standard C library (libc) interfaces will support full 4-byte EUC. Applications which require full 4-byte EUC support must be recompiled on HP-UX 10.0.

EUC Locales [10.0]

In order to support the 4-byte EUC, the following locales are provided:

Table 2-2 Locales for 4-byte EUC Support

Locale Name	Language	Country	Codeset
`ja_JP.eucJP`	Japanese	Japan	AJEC[a] CNS[b]
`zh_TW.eucTW`	Traditional Chinese	Taiwan	11643-1992

a. Advanced Japanese EUC Code (includes JIS X0212 characters).
b. Chinese National Standard

Refer to "New NLS Directory Layout [10.0]" on page 22 for the location of the new locale files.

Disk Space Requirements

The disk space requirements of `ja_JP.eucJP` is approximately 50 Kbytes, and for `zh_TW.eucTW` it is approximately 500 Kbytes. Since most customer systems typically do not require more than one or two locales, the size of these locales should not present a problem.

Migrating HP-UX 9.x Data

The `iconv(1)` and `iconv(3C)` commands can be used to migrate existing HP-UX 9.x data in Japanese or Traditional Chinese codesets to `ja_JP.eucJP` and `zh_TW.eucTW`. Refer to "Enhancements to iconv() Converters [10.0]" on page 51 for the conversions that can be used.

Commands

dtterm(1)

The command `dtterm(1)` must be used instead of the `hpterm(1X)` terminal emulator when using EUC codesets. The `hpterm(1X)` command should be used for 2-byte codesets. Single-byte locales are supported in both `dtterm(1)` and `hpterm(1X)` with the exception of Thai, Arabic, and Hebrew which are not supported in either command.

eucset(1)

This command is new for HP-UX 10.0. It will permit a user to set and get EUC code widths for `ldterm(7)`, the streams line driver. In HP-UX 10.0, users will not need to use `eucset(1)` to set the EUC code widths since `dtterm(1)` will automatically call `eucset(1)`. Users may still wish to inquire about the settings of their EUC code widths.

Pacific Rim Encoding Methods [10.0]

Traditional Chinese 4-Byte EUC

Traditional Chinese EUC will be supported via the locale `zh_TW.eucTW`, which will contain characters from plane numbers 1 - 4 of Chinese National Standard (CNS) 11643-1992.

Table 2-3 Traditional Chinese EUC

Code Set	EUC Representation	Related Character Set	Number of Characters
CS0	0XXXXXXX	ASCII	128
CS1	1XXXXXXX 1XXXXXXX	plane 1 CNS	6,085
CS2	SS2[a] 1XXXXXXX[b]	plane 2 CNS	7,650
	1XXXXXXX 1XXXXXXX	plane 3 CNS	6,148
		plane 4 CNS	7,298
		plane 12 CNS User Definable Character (UDC)	N/A
CS3	not used		

a. SS2 equals x8E.
b. Plane numbers 2 - 16 are encoded here as xA2 - xB0.

Performance

If performance is an issue and the support of planes 3 and 4 are not desired, then it is possible to specify a different font file in `dtterm(1)`.

In order to load only planes 1, 2 and 12 (not planes 3 and 4):

```
$LANG=zh_TW.eucTW
$/usr/vue/bin/dtterm -fn "sung18n*:"
```

In order to load planes 1, 2, 3, 4 and 12:

```
$LANG=zh_TW.eucTW
$/usr/vue/bin/dtterm -fn "sungx18n*:"
```

Plane 12 will have galley patterns initially. New fonts can be created with ASX (Asian System Extension) User Definable Character (UDC) commands later.

Chinese Code For Data Communication

The commands and libc interfaces that support 4-byte EUC will not be able to handle Chinese Code for Data Communication (CCDC) characters whose second byte falls in the range x21 - x3F. As always, these characters cannot be used as filenames.

NOTE CCDC was chinese-t in HP-UX 9.x.

Japanese 3-Byte EUC

Japanese EUC will be supported via the locale `ja_JP.eucJP`. The locale standard Japanese National Profile (JNP) for POSIX has been developed by SC22/POSIX Working Group of Information Technology Standards Commission of Japan (ITSCJ), Information Processing Society of Japan (IPSJ) in an attempt to define Japanese environments. The encoding scheme is Advanced Japanese EUC Code (AJEC), which has been agreed upon by OSF, UNIX International (UI), and UNIX System Laboratories Pacific (USLP).

Table 2-4 Japanese EUC

Code Set	EUC Representation			Related Character Set	Number of Characters
CS0	0XXXXXXX			ASCII	128
CS1	1XXXXXXX	1XXXXXXX		JIS X0208-1983 (Kanji)	6,900
CS2	SS2[a]	1XXXXXXX		JIS X0201 (Katakana)	128
CS3	SS3[b]	1XXXXXXX	1XXXXXXX	JIS X0212-1990 (more Kanji characters)	6,100

a. SS2 equals x8E.
b. SS3 equals x8F.

If performance is an issue and the support of JIS X0212 fonts is not desired, then please refer to the ASX documentation on using the font names or aliases to specify the font set without JIS X0212 glyphs.

Invalid Shift-JIS (SJIS) Ranges

Although the SJIS ranges as defined by PRO JP-TG have not changed from HP-UX 9.x to HP-UX 10.0, the libc interfaces and HP-proprietary (now obsolete) multi-byte macros and interfaces in HP-UX 9.x did not return a failure case for data in invalid SJIS ranges. However, in HP-UX 10.0, failures will be reported for data in invalid SJIS ranges.

Data entered via ASX will be in valid SJIS ranges only, since ASX did not permit users to enter data in invalid SJIS ranges. However, if data was entered for testing purposes only strict attention to the valid SJIS ranges may not have been followed and the test data may have been created in invalid SJIS ranges, which will result in failures with libc interfaces in HP-UX 10.0.

The SJIS ranges permitted in HP-UX 9.x and HP-UX 10.0 are listed below:

Table 2-5 SJIS Ranges

Byte	**HP-UX 9.x SJIS Ranges**		**HP-UX 10.0 SJIS Ranges**		
single-byte	0x00 - 0x7f	0xa1 - 0xdf	0x00 - 0x7f	0xa0 - 0xdf	0xfd - 0xff
first of 2	0x80 - 0xa0	0xe0 - 0xff	0x81 - 0x9f	0xe0 - 0xfc	
second of 2	0x21-0xff (excluding 0x7f)		0x40-0xfc (excluding 0x7f)		

Euro Support

Euro (ISO 8859-15 locales) [11.0 patch, 11i]

Euro support is provided via locale support for the ISO 8859-15 character set. ISO 8859-15 is a newly ratified character set that differs from ISO 8859-1 in that it supports eight new characters. Specific enhancements are provided to allow Euro display, input, and processing capabilities.

Fourteen new locales have been created based on ISO 8859-15:

Table 2-6 New ISO 8859-15 locales

Locale	Language (Country)
C.iso885915	"C"
da_DK.iso885915@euro	Danish (Denmark)
de_DE.iso885915@euro	German (Germany)
en_GB.iso885915@euro	English (Great Britain)
es_ES.iso885915@euro	Spanish (Spain)
fi_FI.iso885915@euro	Finnish (Finland)
fr_CA.iso885915	French (Canada)
fr_FR.iso885915@euro	French (France)
is_IS.iso885915@euro	Icelandic (Iceland)
it_IT.iso885915@euro	Italian (Italy)
nl_NL.iso885915@euro	Dutch (The Netherlands)
no_NO.iso885915@euro	Norwegian (Norway)
pt_PT.iso885915@euro	Portuguese (Portugal)
sv_SE.iso885915@euro	Swedish (Sweden)

Source files for supported European locales are also being supplied.

Applications must elect to enable ISO 8859-15 support, by setting the LANG environment variable to the desired locale.

ISO 8859-15 support is part of HP-UX and is available to all platforms. ISO 8859-15 support is not automatically turned on for any application. No special configuration is required and there are no compatibility issues involved with the addition of this new feature.

Locales are installed, based on which current language filesets are already installed on a target system.

The LC_MONETARY environment variable will be set to the euro for all locales listed above except C.iso885915 and fr_CA.iso885915. Standard euro formatting rules will apply to ALL locales where this environment variable is set to the euro. As a result, users may encounter a change to the decimal and thousands separators for the currency, whereas decimal and thousands separators outside the monetary area stay the same as in previous locales.

For example in the French locale, the thousands separator is a space and the decimal point is a comma. However, the international standard for the thousands separator for the euro currency is a period. So, a user who has the LC_MONETARY locale category set to fr_FR.iso885915@euro will see the following behavior:

- The number one thousand five hundred and fifty and a half, outside the monetary area will be displayed as 1 550,50
- One thousand five hundred and fifty euro and 50 cents will be displayed as EUR 1.550,50.

The LC_MONETARY value can be changed by users to their national currency unit.

ISO 8859-15 support is not automatically provided in any application. Applications which use the Euro symbol must elect to enable ISO 8859-15 support, by way of setting the LANG environment variable to the desired locale. Users enable ISO 8859-15 automatically in some locales when logging in through the CDE.

For more information, please see:

http://software.hp.com/products/EURO/index.html

CDE Support

New functionality was introduced in the CDE product to support input and display of the Euro symbol. (These changes are for both the workstation and the server.)

X Windows Support

New functionality was added to Xlib to support input and display of the Euro symbol. This was done by adding internal support for the ISO8859-15 character set (as well as support of UTF8 on 11.0). When an Xlib application is started, Xlib internals determine if the locale is set to an ISO8859-15 character set. If it is, Xlib will perform character lookups using the eight new symbols present in the ISO8859-15 character set. Currently, only applications linked with X11R6 (X-Windows version X11 Release 6) will support the ISO8859-15 character set. Older X11 versions are not currently supported.

Libraries

The libc and xlib libraries support the Euro symbol.

Codeset Converters

New iconv tables exist to support conversion from/to ISO 8859-15 and ISO 8859-1, ucs2, and utf8. The additional disk space in HP-UX 11.0 is 6.42MB. No additional memory is required.

LaserJet Printers

An important aspect of the euro support is printing the new symbol on LaserJet printers using existing standard `lp(1)` model files.

The ISO8859-15 font set is resident on the HP 4500 Color LaserJet Printer, which contains the Euro symbol at position A4 (hexadecimal). Your data file must contain this code to print the Euro symbol.

A new utility will be provided to download the fonts to the printer RAM. These fonts will then reside in the printer's RAM until the next power cycle.

Use the `lp` option `-ocs9N` (or `-oscs9N`) to select the ISO 8859-15 character set as the primary (or secondary) character set. For example:

`lp -d`*`printer_name`* `-ocs9N -o`*`other_options print_filename`*

NOTE The case is significant. Be sure to use an upper case "N".

Euro - ISO 10646/Unicode Support [11i]

HP-UX 11i provides system level support for the Unicode 2.1/ISO 10646 character set. Hewlett-Plackard's support for Unicode provides a basis of enabling heterogeneous interoperability for all geographic areas.

ISO 10646 is an industry standard for defining a single encoding which uniquely encodes all the characters of the modern world. Unicode 2.1 is the companion specification to ISO 10646. Unicode specification at revision 2.1 includes the Euro symbol at 0X20AC codepoint.

Euro support to input, store, retrieve, display and print the Euro symbol has been added for this release. In addition to the base functionalities, HP-UX 11i is providing the following new functionalities:

- Dual currency support using `@euro` modifier.
- UTF-8 (Universal Transformation Format - 8 Bit) performance tuning.
- Euro display and processing capabilities for Asian UTF-8 locales.
- Additional converter tables.

Specific enhancements are provided to `locales`, `localedef`, `libc`, `Xlib` and `iconv` converter tables to achieve those new functionalities.

A subset of existing European locales has been modified to support dual currency to meet euro standard monetary formatting.

The following table gives the list of euro locales being supplied which support dual currency:

Table 2-7 Supplied utf8 locales supporting dual currency

Locale	Language/Country
de_DE.utf8	German (Germany)
es_ES.utf8	Spanish (Spain)
fr_FR.utf8	French (France)
it_IT.utf8	Italian (Italy)
sv_SE.utf8	Swedish (Sweden)

The following table gives the list of locale sources being supplied which include dual currency support:

Table 2-8 Supplied utf8 locale sources supporting dual currency

Locale	Language/Country
da_DK.utf8	Danish (Denmark)
de_DE.utf8	German (Germany)
el_GR.utf8	Greek (Greece)
en_GB.utf8	English (Great Britain)
es_ES.utf8	Spanish (Spain)
i_FI.utf8	Finnish (Finland)
fr_FR.utf8	French (France)
is_IS.utf8	Icelandic (Iceland)
it_IT.utf8	Italian (Italy)
nl_NL.utf8	Dutch (The Netherlands)
no_NO.utf8	Norwegian (Norway)
pt_PT.utf8	Portuguese (Portugal)
sv_SE.utf8	Swedish (Sweden)

To build these locales, refer to the *localedef* (1M) manpage.

When the `LANG` and/or `LC_*` environment variables are set to a euro supported locale, the national monetary formatting rules are used. The `LC_MONETARY` environment variable should be set to the euro supported locale name with `@euro` modifier to use/access euro monetary formatting rules.

For example, to specify the Euro as the currency for French, the following should be set:

```
LANG=fr_FR.utf8
LC_MONETARY=fr_FR.utf8@euro
```

Similarly, to specify French francs the following should be set:

```
LANG=fr_FR.utf8
```

To access the monetary unit and the related monetary formatting rules programmatically, the programmer needs to toggle between the alternate monetary units via `setlocale(3C)` calls:

```
/* Handle euro in strfmon(), ... */
setlocale(LC_MONETARY, "fr_FR.utf8@euro");

...
/* Handle French francs in strfmon(), ... */
setlocale(LC_MONETARY, "fr_FR.utf8");
```

When the `LC_MONETARY` environment variable is set to euro, the formatting in monetary category will use euro standard formatting rules whereas other categories will still use local convention in formatting. As a result, users may encounter a change to the decimal and thousandths separators for the currency, whereas decimal and thousandths separators outside the monetary area, like in numeric numbers, remain as per local conventions.

For example, in the French locale the thousandths separator is a space and the decimal point is a comma. However, the international standard for the thousandths separator for the euro currency is a period. So, a user that has the `LC_MONETARY` locale category set to `fr_FR.utf8@euro` will see the following behavior:

- The number “One thousand five hundred and fifty and a half” outside the monetary area will be displayed as 1 550,50.
- The monetary number “One thousand five hundred and fifty euro and 50 cents” will be displayed as EUR 1.550,50

Commands

The *localedef* (1M) command has been enhanced to handle the `@euro` modifier in order to build dual currency locale(s).

The *lp* (1) model scripts for the dual currency locales have been enhanced to print the euro character.

libc

Standard libc supports `@euro` dual currency.

Codeset Converters

New `iconv` converter tables exist to support conversion from/to utf8, ucs2 and iso885915, PC codepages and IBM's euro enabled codepages.

New `iconv` converter tables are available to support conversion from utf8, ucs2, and iso885915 to IBM's euro enabled codepages and PC codepages:

Table 2-9 utf8 and IBM's codepages (EBCDIC)

utf8 <-> cp1140	utf8 <-> cp1141	utf8 <-> cp1142	utf8 <-> cp1143
utf8 <-> cp1144	utf8 <-> cp1145	utf8 <-> cp1146	utf8 <-> cp1147
utf8 <-> cp1148	utf8 <-> cp1149		

Table 2-10 ucs2 and IBM's codepages (EBCDIC)

ucs2 <-> cp1140	ucs2 <-> cp1141	ucs2 <-> cp1142	ucs2 <-> cp1143
ucs2 <-> cp1144	ucs2 <-> cp1145	ucs2 <-> cp1146	ucs2 <-> cp1147
ucs2 <-> cp1148	ucs2 <-> cp1149		

Table 2-11 iso885915 and IBM's codepages (EBCDIC)

iso885915 <-> cp1140	iso885915 <-> cp1141	iso885915 <-> cp1142	iso885915 <-> cp1143
iso885915 <-> cp1144	iso885915 <-> cp1145	iso885915 <-> cp1146	iso885915 <-> cp1147
iso885915 <-> cp1148	iso885915 <-> cp1149		

Table 2-12 utf8 and PC codepages (EBCDIC)

utf8 <-> cp437	utf8 <-> cp737	utf8 <-> cp775	utf8 <-> cp850
utf8 <-> cp852	utf8 <-> cp855	utf8 <-> cp857	utf8 <-> cp1860
utf8 <-> cp861	utf8 <-> cp862	utf8 <-> cp863	utf8 <-> cp864
utf8 <-> cp865	utf8 <-> cp866	utf8 <-> cp869	utf8 <-> cp874
utf8 <-> cp1250	utf8 <-> cp1251	utf8 <-> cp1252	utf8 <-> cp1253
utf8 <-> cp1254	utf8 <-> cp1255	utf8 <-> cp1256	utf8 <-> cp1257

Table 2-12 utf8 and PC codepages (EBCDIC)

utf8 <-> cp1258			

Table 2-13 ucs2 and PC codepages (EBCDIC)

ucs2 <-> cp437	ucs2 <-> cp737	ucs2 <-> cp775	ucs2 <-> cp850
ucs2 <-> cp852	ucs2 <-> cp855	ucs2 <-> cp857	ucs2 <-> cp1860
ucs2 <-> cp861	ucs2 <-> cp862	ucs2 <-> cp863	ucs2 <-> cp864
ucs2 <-> cp865	ucs2 <-> cp866	ucs2 <-> cp869	ucs2 <-> cp874
ucs2 <-> cp1250	ucs2 <-> cp1251	ucs2 <-> cp1252	ucs2 <-> cp1253
ucs2 <-> cp1254	ucs2 <-> cp1255	ucs2 <-> cp1256	ucs2 <-> cp1257
ucs2 <-> cp1258			

Impact

To use euro monetary formatting rules, the `LC_MONETARY` environment variable must be set to the euro supported locale name with the `@euro` modifier appended to it.

The size requirement for locale sources and binaries is 20.1MB, while the converter tables size requirement is 191KB.

There are no compatibility issues involved with the addition of these features.

Applications using UTF-8 locales should see improved collation performance as compared with UTF-8 locales delivered in the previous releases.

Unicode Character Set [11.0 Patch, 11i]

HP-UX provides system level support for the Unicode 2.1/ISO 10646 character set. Hewlett-Packard's support for Unicode provides a basis of enabling heterogeneous interoperability for all locales.

ISO 10646 is an industry standard for defining a single encoding which uniquely encodes all the world's characters. Unicode 2.1 is the companion specification to ISO 10646, Unicode support conforms with existing X/Open (OpenGroup), POSIX, ISO C and other relevant UNIX-based standards.

HP-UX 11.0 supports Unicode/ISO 10646 by utilizing the UTF-8 (Universal Transformation Format - 8) representation for persistent storage. UTF-8 is an industry recognized 8-bit multibyte format representation for Unicode. This representation allows for successful data transmission over 8-bit networking protocols as well as for safe storage and retrieval within a historically byte-oriented operating system such as HP-UX.

For internal processing, HP-UX utilizes the four-octet (32-bit) canonical form specified in ISO 10646. This support allows parity with HP-UX's current `wchar_t` implementation which has been based on a 32-bit representation.

Full systems level support is provided for all locales provided in the release.

For more information on the Unicode features of Asian System Environment, see `/usr/share/doc/ASX-UTF8`.

A select subset of locale binaries have been provided for 32-bit application processing:

Table 2-14 Base utf8 locales for 32-bit application processing

Base:	
`C.utf8`	C UTF-8
`univ.utf8`	universal

Table 2-15 European utf8 locales for 32-bit application processing

European:	
fr_CA.utf8	French Canadian
fr_FR.utf8	French
de_DE.utf8	German
it_IT.utf8	Italian
es_ES.utf8	Spanish
sv_SE.utf8	Swedish

Table 2-16 Asian utf8 locales for 32-bit application processing

Asian:	
ja_JP.utf8	Japanese
ko_KR.utf8	Korean
zh_CN.utf8	Simplified Chinese
zh_HK.utf8	Traditional Chinese (Hong Kong)
zh_TW.utf8	Traditional Chinese

To enable Unicode support in applications, set the environment variable to a desired `utf8` locale.

Locales are installed based on the current language filesets already installed on the target system. For example, if the system uses the International.German the German Unicode locale (`de_DE.utf8`) is installed.

Source files for ALL supported locales (34 total) have also been supplied for 64 or 32-bit applications.

To build Unicode locales use the `localedef` command. Refer to the *localedef* (1M) manpage. Systems must have the kernel parameters `MAXDSIZ, MAXTSIZ,` and `SHMMAX` set to at least 100 MB to ensure adequate swap space allowance for successful `localedef` compilation of these locales.

Unicode Euro Enhancement

This release provides expanded Unicode support to align the character repertoire with the ISO 8859-15 locales that are being provided for Euro support. This will ensure full interoperability with the newly added support for the ISO 8859-15 codeset.

Specific enhancements are provided to allow Euro display and input capabilities though Xlib and new fonts.

Size Requirement

Unicode support requires the following additional disk space requirements:

Base Unicode offering (installed on all systems): Approximately 10 MB.

Table 2-17 Unicode European locales and localized files

French & French Canadian	8.4 MB
German	4.2 MB
Italian	4.2 MB
Spanish	4.2 MB
Swedish	4.2 MB

Table 2-18 Unicode Asian locales and localized files

Japanese	3.4 MB
Korean	2.4 MB
Simplified Chinese	2.5 MB
Hong Kong	1.7 MB
Traditional Chinese	4.2 MB

Performance

Applications using Unicode support should see comparable performance as observed with other multibyte codesets. For those applications moving from a single-byte codeset to Unicode, some performance impact will be observed for some types of character based operations.

Streams PTY Driver [11i]

UTF-8 is supported on the Streams PTY driver's line discipline (`LDTERM`) module. The user does not interact with the Streams PTY driver directly; it runs underneath the dtterm window. The Streams PTY driver is responsible for providing a UTF-8 communication channel while `dtterm` is responsible for processing the UTF-8 code and displaying the characters on the screen.

Refer to *eucset* (1), *ldterm* (7) and the *lp* (1) model script for details.

Converting Between Encodings

Enhancements to iconv() Converters [10.0]

New iconv() Converters

The following classes of new `iconv(1)` and `iconv(3C)` codeset conversions are new or enhanced:

1. Traditional Chinese EUC
2. Japanese EUC
3. IBM EBCDIC Japanese
4. IBM EBCDIC PC-Cyrillic
5. IBM EBCDIC PC-Latin-2
6. IBM EBCDIC Traditional Chinese
7. IBM PC code page US
8. IBM PC code page Western Europe
9. IBM code page Western Europe
10. SJIS UDC[1], VDC[2] for HP, mainframe, and PC

For a detailed list of iconv codeset conversions supported, please see the file:

`/usr/lib/nls/iconv/config.iconv`

iconv Name Aliasing

Since iconv table names are cryptic, an iconv alias file will permit the creation of easy-to-use iconv names. For more information on the iconv aliases, please see the comments in the file:

`/usr/lib/nls/iconv/config.iconv`

1. User Defined (or Definable) Code (UDC)
2. Vendor Defined (or Definable) Code (VDC)

Compatibility This aliasing mechanism is HP-specific and is not portable across vendor platforms. The aliases supported are subject to obsolescence/change from release to release.

Customizing iconv Tables and Algorithms

There is a growing need to customize iconv conversion tables. The utilities **genxlt(1)** and **dmpxlt(1)** are new and will enable iconv table customization. **dmpxlt(1)** is used to dump the existing iconv tables into a readable format that can be modified. The command **genxlt(1)** is used to compile the modified iconv table into a format that is usable by **iconv(1)** and **iconv(3C)**.

The ability to customize iconv algorithmic conversions is also possible. This task involves creating a shared object using some pre-defined data structures and entry points. Needed are two conversion tables which map characters to and from one codeset to another, and a method library that may operate on that data. The document containing these instructions is entitled *10.0 Internationalization White Paper* and can be obtained from your support representative.

Disk Space Requirements The total size of both commands is approximately 16 Kbytes.

Table Specific Changes

JIS X0208 Conversion Tables The Japanese SJIS (and UJIS) <-> IBM EBCDIC iconv conversion tables have been updated from JIS X0208-1978 to adhere to JIS X0208-1990. Since the 1978 revision, 2 new standards have been defined: JIS X0208-1983, which added 75 characters and JIS X0208-1990, which added 2 characters.

SJIS and EBCDIC gaiji SJIS tables have been updated to include IBM UDC (gaiji) characters.

SJIS and eucJP table The SJIS<->eucJP conversion is now a table conversion as opposed to an algorithmic conversion as it was in HP-UX 9.x. This change was made to allow users to easily customize the tables with `genxlt(1)` and `dmpxlt(1)`.

Korean table Modifications were made to fix codepoints (including 2-byte punctuation symbols) that were mapped to the same codepoint or to 0xffff. Many KS2 characters, which were previously mapped to 0xffff in HP-UX 9.x are now mapped to characters in EBCDIC (IBM933).

Traditional Chinese EBCDIC IBM UDC (gaiji) characters were added to the big5/EBCDIC tables.

Alternatives The utilities `dmpxlt(1)` and `genxlt(1)` are new and can be used to enable the creation or modification of iconv conversion tables.

Compatibility The changes are upwards compatible.

Miscellaneous iconv Changes

1. JIS->SJIS and JIS->eucJP support more JIS ESCAPE sequences for HP-UX 10.0 than HP-UX 9.x.

Table 2-19 JIS ESCAPE sequences

ESCAPE Sequence	9.0	10.0	Description
ESC(J	Y	Y	JIS X0201 ROMAN8
ESC$B	Y	Y	JIS X0208-1983 Kanji
ESC$C	Y	N	
ESC(B	N	Y	JIS ASCII
ESC(H	N	Y	IS X0208 ROMAN8
ESC$@	N	Y	JIS X0208-1978 Kanji
ESC$(@	N	Y	JIS X0208-1978 Kanji
ESC$(B	N	Y	JIS X0208-1983 Kanji
ESC&@ESC$B	N	Y	JIS X0208-1990 Kanji
ESC&@ESC$(B	N	Y	JIS X0208-1990 Kanji
ESC(I	N	Y	JIS X0201 Katakana
ESC$(D	N	Y	JIS X0212-1990 Kanji 3 byte

2. New Simplified Chinese tables: hp15CN<->chinese-s_e

Unmapped Character Conversion

Error handling now exists for invalid character conversions. If an invalid conversion character is detected, the following will occur:

- If a galley[1] character is specified in the iconv table, the galley character will be output.
- If a galley character is not specified in the iconv table, the conversion will stop.

See **genxlt(1)** for more information regarding the usage of the galley character.

Invertible Conversions

HP-UX provides conversions based on readability rather than invertibility. A readable conversion is one that maps characters into other characters that look very similar; it is easily converted back to the original characters once a readable conversion has been used. An invertible conversion is one that maps characters into other characters such that it is possible to convert back to the original characters after an invertible conversion had been used.

If invertible conversions are required, the existing readable conversion tables can be modified to make them invertible tables with `dmpxlt(1)` and `genxlt(1)`.

Here are some general instructions that explain how to create an invertible conversion for two codesets, which are the same size and are not 1-to-1 mapped:

1. sort unmapped characters in both codesets by value
2. map those characters according to sequence orders

For example:

Roman8 <==> ISO 8859-1:

There are 12 unmapped characters in the conversion between HP-Roman8 and ISO 8859-1. The invertible conversion rule is:

1. A galley character is a default character that is output if a conversion doesn't exist

Table 2-20 Invertible conversion rule for conversion between HP-Roman8 and ISO 8859-1

Roman8 (hexvalueanddescription)		ISO 8859/1 (hex value and description)	
0xA9	Accent grave	0xA6	Broken bar
0xAA	Circumflex accent	0xA9	Copyright sign
0xAC	Tilde accent	0xAC	Not sign
0xAF	Italian Lira	0xAD	Soft hyphen
0xB0	Overline	0xAE	Registered trade mark
0xBE	Dutch guilder	0xAF	Macron
0xEB	S caron	0xB2	Superscript two
0xEC	s caron	0xB3	Superscript three
0xEE	Y umlaut	0xB8	Cedilla
0xF6	Long dash	0xB9	Superscript one
0xFC	Solid box	0xD7	Multiplication sign
0xFF	(undefined)	0xF7	Division sign

Enhancements to iconv(1) and iconv(3) [10.10]

- Now conforms to XPG4 specifications.
- Increased codeset conversion support.
- Warning messages will no longer be printed.
- New codeset naming convention, but old convention still supported.
- Shift-in characters not added while converting to IBM or JIS codesets, if an ASCII character is the first character of the record.

See the *README for NLS in 10.01* and the `iconv(1)` manpage for details.

Enhancements to iconv(1) and iconv(3) [10.20]

10646 conversions for UCS forms: UCS-2 and UTF-8

iconv(1) and **iconv(3)** can now support the conversion of characters between some of the HP-supported code sets and various forms of the ISO 10646 code set, specifically the UCS forms, UCS-2 and UTF-8. This feature will enable HP-UX to exchange and interoperate with other systems which support ISO 10646 characters.

WARNING **Do not store UCS-2 data in files, unexpected results may occur. Only UTF-8 data may be safely stored in files for processing.**

WARNING **This is only a conversion feature to be used to facilitate input/output with other systems and for code-internal purposes. None of the rest of HP-UX (commands, utilities, services, etc.) have been converted to support UCS-2 or UTF-8 at this time. Only iconv supports this feature currently.**

The following conversions are supported:

- UTF-8, UCS-2 <-->roman8, iso8859-1 for West European languages
- UTF-8, UCS-2 <-->SJIS, eucJP for Japanese
- UTF-8, UCS-2 <-->eucKR for Korean
- UTF-8, UCS-2 <-->hp15CN for Simplified Chinese
- UTF-8, UCS-2 <-->roc15, big5, eucTW for Traditional Chinese
- UTF-8, UCS-2 <-->thai for Thai
- UTF-8, UCS-2 <-->iso8859-2 for East European
- UTF-8, UCS-2 <-->iso8859-5 for Cyrillic
- UTF-8, UCS-2 <-->iso8859-6 for Arabic
- UTF-8, UCS-2 <-->iso8859-7 for Greek
- UTF-8, UCS-2 <-->iso8859-8 for Hebrew
- UTF-8, UCS-2 <-->iso8859-9 for Turkish
- UTF-8 <--> UCS-2

Additional tables and methods have been added to support this conversion capability.

Corrected Character Mappings to *iconv*(1) and *iconv*(3C) [11.0 patch, 11i]

This release contains defect fixes for incorrect character mappings. The corrections concern the Simplified Chinese, Traditional Chinese, Japanese, and Korean characters of HP-UX.

Corrected character converter mappings allow for improved interoperability when sending or receiving converted character data to/from Unicode-aware systems.

Correction for Simplified Chinese

A patch corrects an incorrect character mapping that occurs when converting between hp15CN and Unicode (UCS2)/UTF-8 for Simplified Chinese.

Specifically, the Simplified Chinese character “Double Vertical Line” mapped incorrectly when converting between hp15CN and UCS2/UTF-8. This character was being mapped to the “Parallel To” character, which is a different character.

The following table summarizes the change applied to `iconv` tables:

Table 2-21 Changes in iconv tables for Simplified Chinese

hp15CN	incorrect UCS2	correct UCS2	Character Name
0xA1CE	-	0x2225	Parallel To
0xA1AC	0x2225	0x2016	Double Vertical Line

The `hp15CN=ucs2` and `ucs2=hp15CN` `iconv` converter tables are affected. These tables are shared by both UCS2 and UTF-8 conversions.

No compatibility problems are anticipated. However, if compatibility concerns arise with regard to persistent data stored either in Unicode (UCS2) or UTF-8 on an HP-UX system, it is possible to generate a simple conversion script to search for each occurrence of an incorrect value in either UCS2 or UTF-8 and convert it to the correct value, based on the following mapping:

Table 2-22 Mapping between old and new Unicode characters for Simplified Chinese

Old UCS2	UCS2	Old UTF-8	UTF-8	Char Name
0x2225	0x2016	0xe288a5	0xe28096	Double Vertical Line

Correction for Traditional Chinese

A patch corrects several incorrect character mappings that occur when converting between Big-5/EUC and Unicode (UCS2)/UTF-8 for Traditional Chinese.

In the case of Big-5 to/from UCS2/UTF-8, the “Ideographic Space” character was absent in the Unicode conversion table mapping:

Table 2-23 Changes in iconv tables for big5/Unicode

big5	incorrect UCS2	correct UCS2	Char Name
0xA140	-	0x3000	Ideographic Space

The following table summarizes the changes applied for conversions between eucTW and UCS2:

Table 2-24 Changes in iconv tables for eucTW/Unicode

eucTW	incorrect UCS2	correct UCS2	Character Name
0xa1a6	0x30fb	0x2022	Bullet
0xa1b7	0x2014	0x2013	EN Dash
0xa1b9	0x2013	0x2014	EM Dash
0xa1b6	0xfe31	0xff5c	Fullwidth Vertical Line
0xa1b8	0xfe32	0xfe31	Presentation form Vertical EN Dash
0xa1ea	0x2032	0x2035	Reversed Prime

Table 2-24 Changes in iconv tables for eucTW/Unicode

eucTW	incorrect UCS2	correct UCS2	Character Name
0xa1eb	0x2035	0x2032	Prime
0xa2b9	0x2264	0x2266	Less-than over equal to
0xa2ba	0x2265	0x2267	Greater-than over equal to
0xa2c2	0xfe66	0xfe65	Small Greater-Than
0xa2c3	0xfe65	0xfe66	Small Equals Sign
0xa2de	0xff5c	0x2223	Divides
0xa2e1	0xfe67	0xff0f	Full-width Solidus
0xa2e4	0xffe5	0x00a5	Yen Sign
0xa2e6	0xffe0	0x00a2	Cent Sign
0xa2e7	0xffe1	0x00a3	Pound Sign

`iconv` conversions between eucTW and UCS2 or UTF-8 may be affected.

Big-5 conversions with UCS2/UTF-8 are not directly impacted as only a missing table entry has been added.

`eucTW=ucs2`, `ucs2=eucTW`, `big5=ucs2` and `ucs2=big5` `iconv` converter tables are affected. These tables are shared by both UCS2 and UTF-8 conversions.

No compatibility problems are anticipated. However, if compatibility concerns arise with regard to persistent data stored either in Unicode (UCS2) or UTF-8 on an HP-UX system, it is possible to generate a simple conversion script to search for each occurrence of an incorrect value in either UCS2 or UTF-8 and convert it to the correct value, based on the following mappings:

Table 2-25 Mapping between old and new Unicode characters for Traditional Chinese

Old UCS2	UCS2	Old UTF-8	UTF-8	Char Name
0x30fb	0x2022	0xe383bb	0xe280a2	Bullet
0x2014	0x2013	0xe28094	0xe28093	EN Dash
0x2013	0x2014	0xe28093	0xe28094	EM Dash
0xfe31	0xff5c	0xefb8b1	0xefbd9c	Fullwidth Vertical Line
0xfe32	0xfe31	0xefb8b2	0xefb8b1	Presentation form Vertical EN Dash
0x2032	0x2035	0xe280b2	0xe280b5	Reversed Prime
0x2035	0x2032	0xe280b5	0xe280b2	Prime
0x2264	0x2266	0xe289a4	0xe289a6	Less-than over equal to
0x2265	0x2267	0xe289a5	0xe289a7	Greater-than over equal to
0xfe66	0xfe65	0xefb9a6	0xefb9a5	Small Greater-Than
0xfe65	0xfe66	0xefb9a5	0xefb9a6	Small Equals Sign
0xff5c	0x2223	0xefbd9c	0xe288a3	Divides
0xfe67	0xff0f	0xefb9a7	0xefbc8f	Full-width Solidus
0xffe5	0x00a5	0xefbfa5	0xc2a5	Yen Sign
0xffe0	0x00a2	0xefbfa0	0xc2a2	Cent Sign
0xffe1	0x00a3	0xefbfa1	0xc2a3	Pound Sign

Correction for Japanese

A patch corrects four incorrect Japanese character mappings that occur between Shift-JIS/EUC and Unicode (UCS2)/UTF-8.

The following table summarizes the changes applied:

Table 2-26 Changes in iconv tables for Japanese

sjis	eucJP	incorrect UCS2	correct UCS2	Character Name
0x8150	0xA1B1	0xFFE3	0x203E	Overline
0x815C	0xA1BD	0x2015	0x2014	Em Dash
0x818F	0xA1EF	0xFFE5	0x00A5	Yen Sign
n/a	0x8FA2B7	0x02DC	0xFF5E	Full-width Tilde

Affected `iconv` conversions are conversions between sjis and UCS2 or UTF-8 as well as conversions between eucJP and UCS2 or UTF-8.

`sjis=ucs2`, `ucs2=sjis`, `eucJP=ucs2` and `ucs2=eucJP` are the affected `iconv` conversion tables. These tables are shared by both UCS2 and UTF-8 conversions.

No compatibility problems are anticipated. However, if compatibility concerns arise with regard to persistent data stored either in Unicode (UCS2) or UTF-8 on an HP-UX system, it is possible to generate a simple conversion script to search for each occurrence of an incorrect value in either UCS2 or UTF-8 and convert it to the correct value, based on the following mappings:

Table 2-27 Mapping between old and new Unicode characters for Japanese

Old UCS2	UCS2	Old UTF-8	UTF-8	Char Name
0xFFE3	0x203E	0xefbfa3	0xe280be	Overline
0x2015	0x2014	0xe28095	0xe28094	Em Dash
0xFFE5	0x00A5	0xefbfa5	0xc2a5	Yen Sign
0x02DC	0xFF5E	0xcb9c	0xefbd9e	Full-width Tilde

Correction for Korean

A patch provides a defect fix to address standards non-conformance for Korean Unicode (UCS2)/UTF-8 character mappings.

The currently supplied Korean `iconv` converter tables do not conform to the Unicode 2.1 and ISO 10646 (with 1997 amendments) standards in addition to the Korean national standard, KSC-5700. The current mappings are considered obsolete by all noted standards organizations.

The enhancement provides a set of standards-conformant `iconv` converter tables for converting between eucKR and Unicode/UTF-8. Specifically, the obsolete region of 0x3d2e - 0x4dff has been re-mapped to the 0xac00 - 0xd7ff region specified in Unicode 2.1 for Hangul.

Without this modification, it is impossible to share data with any other system which is standards conformant in adhering to the Unicode 2.1/ISO 10646/KSC-5700 standards.

Affected `iconv` conversions are any conversions between eucKR and UCS2 or UTF-8.

The `iconv` conversion tables affected by this modification are `eucKR=ucs2` and `ucs2=eucKR`. These tables are shared by both UCS2 and UTF-8 conversions.

No compatibility problems are anticipated. However, if compatibility concerns arise with regard to persistent data stored either in Unicode (UCS2) or UTF-8 on an HP-UX system, it is recommended that the previously installed `ucs2=eucKR` table be saved and renamed prior to installation of this fix. Persistent data can then be converted back to eucKR using this old table and then reconverted to the correct Unicode/UTF-8 representation.

3 Locales

Changes to locales [10.0]

New Directory Structure For Locales

HP-UX 9.x locale files were stored in a hierarchical directory structure:

`/usr/lib/nls/`*`language`*`/`*`territory`*`/`*`codeset`*

All HP-UX 10.0 locales will be stored in a flat directory structure:

`/usr/lib/nls/loc/locales`

The locale filenames are in conformance to the ISO 639-defined two character abbreviations for language names, and ISO 3166-defined two character abbreviations for country names. The template for the filenames is:

ISO 639 language abbreviation[*_ISO 3166 country*][*.codeset name*][@*modifier*]

Locale Source Files

The `-d` flag of the `localedef(1M)` command will no longer be supported. Instead, the `localedef(1M)` input source files will be provided for customer locale customization. The source files will be located in:

`/usr/lib/nls/loc/src/`*`locale name`*`.src`

Disk Space Requirements

Fileset groupings are based on language name, so there is a fine granularity for selecting the locales desired. The average size of a non-Asian locale source file will be approximately 17 - 20 Kbytes. The average size average size of an Asian locale may be as much as 60 Kbytes. Most customer squire at most one or two locales.

Build Change For Archive Internationalized Applications

WARNING **It is not recommended to build internationalized applications using an archive libc, but if this is still desired, there are build changes for ALL archive programs that use NLS.**

Any application that calls **setlocale(3C)** or **iconv(3C)** and is compiled archive will not be a complete archive, and will now contain position independent code and data, which are loaded at runtime. Two calls in `libdld.sl` are used to load the position independent code and data must be loaded as shared; however, the rest of the libraries can be archived in the executable.

The compile and load options for applications built with `-Wl,-a,archive` are as follows. Please be aware that the `-Wl,-a,archive` link option is a positionally dependent option and should occur at the beginning of the `cc(1)` line in order to compile the program.

- Example with `sh(1)` using `CCOPTS` and `LDOPTS`:

  HP-UX 9.x compile:

```
export CCOPTS="${CCOPTS:-} your_options"
export LDOPTS="${LDOPTS:-} your_options"
cc -Wl,-a,archive -o executable source_file
```

  HP-UX 10.x compile:

```
export CCOPTS="${CCOPTS:-} | your_options -Wl,-E -l:libdld.sl"
export LDOPTS="${LDOPTS:-} your_options -E -l:libdld.sl"
cc -Wl,-a,archive -o executable source_file
```

- Example using `cc(1)` line only:

  HP-UX 9.x compile:

```
cc -Wl,-a,archive -o executable source_file
```

  HP-UX 10.x compile:

```
cc -Wl,-a,archive -Wl,-E -l:libdld.sl -o executable source_file
```

Rationale

Locales are now stored as shared objects that are dynamically loaded at run time. This strategy will allow customization of libc interfaces to handle new or existing codesets and removes locale-specific code from `libc`.

This customization is a complex task. If you need to do this, please contact your support representative and ask for the *10.0 Internationalization White Paper*.

Alternatives

The application can be linked shared which is the default option.

Compatibility

It is not recommended that an application have dependencies on both archive and shared libraries. Library calls may be resolved in unexpected ways, especially when objects comprising the executable were created on different releases of HP-UX.

For further information on mixing shared and archive libraries, please refer to *Programming on HP-UX*.

Disk Space Requirement

With the exception of the EUC locales which average approximately 350 Kbytes per locale, the size of most other locales is about 37 Kbytes. Most customer systems typically need one or two locales.

ISO Locale Names Supported

The HP-UX locales have been renamed in HP-UX 10.0 in order to conform to ISO standards. The HP-UX 10.0 locale names use the standard ISO 639 language name and the ISO 3166 territory name.

The HP-UX 9.x locale names will still work on HP-UX 10.0. For compatibility purposes, the new 10.0 locale names are linked to the 9.x HP-proprietary locale names. The 9.x locale names have been obsoleted in that no further enhancements will be done to those locale names.

Examples of linked locale names:

`/opt/`*`application`*`/lib/nls/msg/japanese/euc`

is linked to

`/opt/`*`application`*`/lib/nls/msg/ja_JP.eucJP`

and

`/opt/`*`application`*`/lib/nls/msg/french`

is linked to

`/opt/`*`application`*`/lib/nls/msg/fr_FR.roman8`

The following list is alphabetized using the HP-UX 10.0 locale names. The naming conventions used for the locales conform to ISO 639 for the language element, and ISO 3166 for the territory. There is no standard convention for the codeset element.

Table 3-1 New ISO Locale Names

Old 9.x Locale Name	New HP-UX 10.0 Locale Name	Description (language, territory, codeset)
american	en_US.roman8	English, United States, roman8
american.iso88591	en_US.iso88591	English, United States, ISO 8859/1
arabic	ar_SA.arabic8	Arabic, Saudi Arabia, arabic8
arabic.iso88596	ar_SA.iso88596	Arabic, Saudi Arabia, ISO 8859/6
arabic-w	ar_DZ.arabic8	Arabic, Algeria, arabic8
bulgarian	bg_BG.iso88595	Bulgarian, Bulgaria, ISO 8859/5
C	C	Computer default - same as POSIX
c-french	fr_CA.roman8	French, Canada, roman8
c-french.iso88591	fr_CA.iso88591	French, Canadian, ISO 8859/1
chinese-s	zh_CN.hp15CN	Simplified Chinese, China, GB 2312-80
chinese-t	zh_TW.ccdc	Traditional Chinese, Taiwan, CCDC
chinese-t.big5	zh_TW.big5	Traditional Chinese, Taiwan, BIG5
none[a]	zh_TW.eucTW	Traditional Chinese, Taiwan, CNS 11643-92
czech	cs_CZ.iso88592	Czech, Czech Republic, ISO 8859/2
danish	da_DK.roman8	Danish, Denmark, roman8
danish.iso88591	da_DK.iso88591	Danish, Denmark, ISO 8859/1
dutch	nl_NL.roman8	Dutch, Netherlands, roman8
dutch.iso88591	nl_NL.iso88591	Dutch, Netherlands, ISO 8859/1
english	en_GB.roman8	English, United Kingdom, roman8
english.iso88591	en_GB.iso88591	English, United Kingdom, ISO 8859/1
finnish	fi_FI.roman8	Finnish, Finland, roman8
finnish.iso88591	fi_FI.iso88591	Finnish, Finland, ISO 8859/1

Table 3-1 New ISO Locale Names

Old 9.x Locale Name	New HP-UX 10.0 Locale Name	Description (language, territory, codeset)
french	fr_FR.roman8	French, France, roman8
french.iso88591	fr_FR.iso88591	French, France, ISO 8859/1
german	de_DE.roman8	German, Germany, roman8
german.iso88591	de_DE.iso88591	German, Germany, ISO 8859/1
greek[b]	el_GR.greek8	Greek, Greece, greek8
greek.iso88597	el_GR.iso88597	Greek, Greece, ISO 8859/7
hebrew[c]	iw_IL.hebrew8	Hebrew, Israel, hebrew8
hebrew.iso88598	iw_IL.iso88598	Hebrew, Israel, ISO 8859/8
hungarian	hu_HU.iso88592	Hungarian, Hungary, ISO 8859/2
icelandic	is_IS.roman8	Icelandic, Iceland, roman8
icelandic.iso88591	is_IS.iso88591	Icelandic, Iceland, ISO 8859/1
italian	it_IT.roman8	Italian, Italy, roman8
italian.iso88591	it_IT.iso88591	Italian, Italy, ISO 8859/1
japanese	ja_JP.SJIS	Japanese, Japan, SJIS
japanese.euc	ja_JP.eucJP	Japanese, Japan, AJEC
katakana	ja_JP.kana8	Japanese, Japan, kana8 - to be OBSOLETED
korean	ko_KR.eucKR	Korean, Korea, KS C5601
n-computer	none	obsolete in 10.0 - use C or POSIX instead
norwegian	no_NO.roman8	Norwegian, Norway, roman8
norwegian.iso88591	no_NO.iso88591	Norwegian, Norway, ISO 8859/1
polish	pl_PL.iso88592	Polish, Poland, ISO 8859/2
portuguese	pt_PT.roman8	Portuguese, Portugal, roman8

Table 3-1 New ISO Locale Names

Old 9.x Locale Name	New HP-UX 10.0 Locale Name	Description (language, territory, codeset)
portuguese.iso88591	pt_PT.iso88591	Portuguese, Portugal, ISO 8859/1
POSIX	POSIX	specified by POSIX - contains only POSIX portable characters
rumanian	ro_RO.iso88592	Rumanian, Romania, ISO 8859/2
russian	ru_RU.iso88595	Russian, Russian Federation, ISO 8859/5
serbocroatian	hr_HR.iso88592	Croatian, Croatia, ISO 8859/2
none[d]	sk_SK.iso88592	Slovakian, Slovakia, ISO 8859/2
slovene	sl_SI.iso88592	Slovenian, Slovenia, ISO 8859/2
spanish	es_ES.roman8	Spanish, Spain, roman8
spanish.iso88591	es_ES.iso88591	Spanish, Spain, ISO 8859/1
swedish.iso88591	sv_SE.iso88591	Swedish, Sweden, ISO 8859/1
swedish	sv_SE.roman8	Swedish, Sweden, roman8
thai	th_TH.tis620	Thai, Thailand, tis620
turkish[e]	tr_TR.turkish8	Turkish, Turkey, turkish8
turkish.iso88599	tr_TR.iso88599	Turkish, Turkey, ISO 8859/9

a. zh_TW.eucTW is new in HP-UX 10.0.
b. In HP-UX 10.0, the greek locale name is internally mapped to el_GR.iso88597 rather than el_GR.greek8. See “Modifications To Locales” on page 71 for more details.
c. In HP-UX 10.0, the hebrew locale name is internally mapped to iw_IL.iso88598 rather than iw_IL.hebrew8. See “Modifications To Locales” on page 71 for more details.
d. sk_SK.iso88592 is new in HP-UX 10.0.
e. In HP-UX 10.0, the turkish locale name is internally mapped to tr_TR.iso88599 rather than tr_TR.turkish8. See “Modifications To Locales” on page 71 for more details.

Alternatives

If you need to use locale names other than the ones listed in Table 3-1, “New ISO Locale Names” on page 67, you can:

- Add a personal alias in you `$HOME/.profile` or `$HOME/.cshrc` files.
- Add a global alias for all users in the `/etc/profile` or `/etc/csh.login` files.
- Create a symbolic link to your desired locale name. If you do this, you may need to link the directory containing your local translations. This includes HP-UX core and optional products.

Performance

The use of HP-UX 10.0 locale names will result in slightly better performance.

Compatibility

The mapping between HP-UX 9.x locale names and the standard ISO names can be found in the file:

`/usr/lib/nls/config`

During this period of transition to the new ISO locale names, products, scripts, and users may not have fully converted to the new ISO locale names. Therefore, it is recommended that links should be created from directories containing local translations with the new HP-UX 10.0 locale names to the old HP-UX 9.x locale names. These links will ensure that products and scripts continue to operate for products, scripts and users continuing to use the old 9.x locale names, and it will also satisfy those using the new ISO locale names.

Example for HP-UX core products with Swedish ROMAN8 translations:

```
ln /opt/lib/nls/msg/sv_SE.roman8 /usr/lib/nls/swedish
```

Example for optional products with Japanese EUC translations:

```
ln /opt/product/lib/nls/ja_JP.eucJP /opt/product/lib/nls/japanese/euc
```

Modifications To Locales

Summary of Changes

The 9.x Locale Names will be obsoleted.

Table 3-2 Summary Of Modifications To Locales

HP-UX 10.0 Locale Names	HP-UX 9.x Locale Names	Comments
ALL locales	N/A	Only the folded collating sequence will be provided in the locale source files. A folded collating sequence is made up of the uppercase and lowercase characters intermixed. An unfolded collating sequence is made up of all the uppercase characters followed by the lowercase characters.
		Folded collating sequence provides the greatest degree of compatibility across vendors platforms. The X/Open locale registry provides only folded collation in its locales.
		The format of `grouping` and `mon_grouping` have been modified to adhere to XPG4.
All multi-byte locales	N/A	Character classification (`ctype`) information is now provided for multi-byte characters. For detailed information regarding specific characters, see the locale source files in `/usr/lib/nls/loc/src`.

Table 3-2 Summary Of Modifications To Locales

HP-UX 10.0 Locale Names	HP-UX 9.x Locale Names	Comments
cs_CZ.iso88592	czech	The czech locale has undergone the following modifications: lower case letters collate first Characters 0xc8, 0xe8 | 0xd8, 0xf8 | 0xa9, xb9 | 0xae, 0xbe must have their own primary sequence numbers. Combination 'cH' will not represent one char in the collation The currency sign has been changed from K<C-caron>S to <K><C-caron> The international currency sign has been changed from CSK to CZK
de_DE.iso88591 de_DE.roman8	german.iso88591 german	The international currency sign has been changed from DDM to DEM.
el_GR.greek8 iw_IL.hebrew8 tr_TR.turkish8	greek hebrew turkish	In 9.x, the codesets of the greek, hebrew, and turkish locales were respectively greek8, hebrew8, and turkish8; however, they will now be based on the respective ISO codesets: ISO 8859/7, ISO 8859/8, and ISO 8859/9. If backwards compatibility with the 9.x greek (greek8), hebrew (hebrew8), and turkish (turkish8) locales is desired, the locale names el_GR.greek8, iw_IL.hebrew8, and tr_TR.turkish8 should be used.
hr_HR.iso88592	serbocroation	The international currency symbol changed from DIN to HRD.

Table 3-2 Summary Of Modifications To Locales

HP-UX 10.0 Locale Names	HP-UX 9.x Locale Names	Comments
ja_JP.eucJP	japanese.euc	Locale sources were replaced with the UI-OSF Japanese locale with the UI-OSF AJEC[a] charmap. There were many changes in the locale and some of these changes include: removed era_year removed era dates prior to 1927 removed LC_TIME time unit descriptors removed YESSTR / NOSTR collation sequences were modified to include o only valid characters in the binary sort order added HP UDC/VDC characters added bkinsoku and ekinsoku characters the date format follows the format of the Western year
ja_JP.SJIS	japanese	Locale sources were replaced with the UI-OSF Japanese locale with the PRO JL-TG[b] SJIS charmap. The rest of the comments are the same as ja_JP.eucJP described above.
ko_KR.eucKR	korean	Era information from the LC_TIME category has been removed from the locale.
nl_NL.iso88591 nl_NL.roman8	dutch.iso88591 dutch	The time format has been changed to adhere to standards. The character that separates hours, minutes, and seconds will be changed from "." to ":". Example, 9.x time format %H.%M.%S will be changed to 10.0 %H:%M:%S.
si_SI.iso88592	slovene	The international currency symbol has changed from YUD to SIT. The currency symbol has changed from DIN to SIT.

Table 3-2 Summary Of Modifications To Locales

HP-UX 10.0 Locale Names	HP-UX 9.x Locale Names	Comments
th_TH.tis620	thai	Thai sorting has been provided by using the POSIX locale model to fit the special Thai collating sequence into the locale. More than 240 many-to-one characters have been defined and the primary weight has been set to IGNORE for tone mark characters. With this locale definition, strcoll(3C) and sort(1) can handle the Thai sorting correctly.
zh_TW.big5 zh_TW.ccdc	chinese-t.big5 chinese-t	era has been changed due to XPG4 to use %EC and %Ey instead of %N.

a. AJEC - Advanced Japanese EUC Code
b. PRO (Precision RISC Organization) JL-TG will define and establish a standard Japanese Localization environment in order to increase ISV support and widely promote PA-RISC based systems into the Japanese engineering and business market. In defining the above standard, PRO will pursue conformance with international and de-facto standards.

See “localedef(1M)” on page 112 for locale source file and keyword changes.

Other Locale Changes

1. en_GB.iso88591 and en_GB.roman8 - abbreviated month (ABMON) values for month 6 and 7 (of 12):

   HP-UX 9.0: June, July

   HP-UX 10.0: Jun, Jul

   Example of HP-UX 9.0 date: Thu June 9 12:17:17 PDT 1994

   Example of HP-UX 10.0 date: Thu Jun 9 12:17:17 PDT 1994

   This change is visible via nl_langinfo(3C), date(1), ...

2. zh_TW.ccdc and zh_TW.big5 - date time format (D_T_FMT):

   HP-UX 9.0: %E%b ...

   HP-UX 10.0: %EY%b ...

This change is visible via `nl_langinfo(3C)`, ...

Impact: locale sensitive date/time results may be different.

Alternatives: customers can modify locale with `localedef(1M)`.

3. HP-UX 8.0 locales are no longer provided. No international support for HP-UX 8.0 archive applications.

   Impact: internationalized applications that call `setlocale(3C)` will not run in internationalized mode. The C locale behavior will be used.

   Alternatives: copy HP-UX 8.0 locales on to the HP-UX 10.0 system.

4. 2-1 characters such as the sharp-S and SS in locales such as `de_DE.roman8` and `de_DE.iso88591` now collate the same.

   Impact: sorting algorithms for 2-1 characters now conform to XPG4 and local conventions for handling 2-1 characters. Sorting results may be different in HP-UX 10.0. This behavior may be exhibited in the interfaces and utilities such as: `strcoll(3C)`, `strxfrm(3C)`, `wcscoll(3C)`, `wcsxfrm(3C)`, `sort(1)`.

   Alternatives: if you prefer the non-XPG4 behavior, modify the locale to change the sort order so that the 2-1 characters will collate in a different order.

Changed options for locale(1)

`-a`	Lists all available public locales.
`-k`	Displays the name of the `charmap` file if `charmap` is specified. Supports new keywords in the `LC_TIME` category.
`-c`	Supports the `CHARMAP` category.

Compatibility

If compatibility with the HP-specific 9.x locale is desired, the 9.x `/usr/old/usr/bin/localedef(1M)` can be used to dump the HP-UX 9.x locale. These changes can then be applied to the sources of the HP-UX 10.0 locales. There are a number of differences in format with the 10.0 locale, so it is recommended that the 10.0 locale is used as the basis for customizing with data from the 9.x locale.

Character Maps (charmaps) Supported

In the interest of adhering to the latest conventions and standards, locale source files will contain charmap symbolic names. Literals will no longer be accepted by **localdef(1M)**.

The following charmaps will be provided in `/usr/lib/nls/loc/charmaps` for use in modifying or creating a locale with `localedef -f` in order to resolve the charmap symbolic names used in the locale source files:

Table 3-3 Charmaps Supported

Charmap File	Description
`SJIS.cm`	Implementation of the Japanese Shift-JIS encoding method using ISO 646:1991 IRV, JIS X0201:1976, and JIS X0208:1990.
`arabic8.cm`	arabic8 is an HP 8-bit codeset comprised of ASCII and a superset of ASMO 449.
`big5.cm`	big5 is an implementation of the big5 de facto encoding standard in Taiwan for Traditional Chinese and contains 13,052 characters from CISCII (Chinese Industrial Standard Code for Information Interchange: 1986), with 1,700 code values being reserved for user-defined characters.
`ccdc.cm`	ccdc is based on the Chinese Code for Data Communications standard published in 1984 by the Taiwan Telegraph Bureau. It defines 16,384 Traditional Chinese characters into 2 levels which include chinese characters, special symbols, and user-defined characters.
`hp15CN.cm`	hp15CN is an HP encoding method for Simplified Chinese which implements the Chinese National Standard GB 2312-1980. This includes common Chinese characters which are sorted phonetically (Level 1), other Chinese characters which are sorted according to radical and number of strokes (Level 2), as well as special symbols and space for additional user-defined characters.
`eucJP.cm`	Implementation of the EUC (Extended UNIX Codes) encoding method for the Japanese JIS standard with ISO 646:1991 IRV assigned to CS0, JIS X0208:1990 assigned to CS1, JIS X0201:1976 assigned to CS2, and JIS x0212:1990 assigned to CS3.
`eucTW.cm`	Implementation of the EUC (Extended UNIX Codes) encoding method for Traditional Chinese with ISO 646:1991 IRV assigned to CS0, CNS 11643:1992 plan 1 assigned to CS1, CNS 11643:1993 planes 2-16 assigned to CS2.

Table 3-3 Charmaps Supported

Charmap File	Description
eucKR.cm	eucKR is an encoding method which implements the Korean National Standard KS C5601-1987. Includes Hangul and Hanja characters as well as special symbols and space for additional user-defined Hangul and Hanja characters.
greek8.cm	greek8 is an HP 8-bit code-set that is comprised of ASCII and characters defined in ECMA-118 Latin/Greek. However, it is not identical to ECMA-118, as different code locations are defined for some symbols. (to be obsoleted)
hebrew8.cm	hebrew8 is an HP 8-bit code-set that is comprised of ASCII and characters defined in ECMA-121. However, it is not identical to ECMA-121, as different code locations are defined for some symbols. (to be obsoleted)
iso88591.cm	ISO8859-1 Latin-1 1987
iso88592.cm	ISO8859-2 Latin-2 1987
iso88595.cm	ISO8859-5 Cyrillic Latin 1988
iso88596.cm	ISO8859-6 Arabic Latin 1987
iso88597.cm	ISO8859-7 Greek Latin1987
iso88598.cm	ISO8859-8 Hebrew Latin1988
iso88599.cm	ISO8859-9 Latin-5 1989
kana8.cm	kana8 is an HP 8-bit code-set for Japanese comprised of JASCII and one-byte Katakana. (to be obsoleted)
roman8.cm	roman8 is an HP 8-bit code-set comprised of ASCII and subset of ECMA-94 Latin 1.
tis620.cm	tis620 is based on the standard specified by the Thai Industrial Standard Institute (TISI) 620-2533
turkish8.cm	turkish8 is an HP 8-bit code-set that is comprised of ASCII and Turkish characters. It is different than ECMA-94 Latin 3 or ECMA-128 Latin 5. (to be obsoleted)

Disk Space Requirements

A customer typically needs only a few charmaps on the system. The average size of a non-Asian charmap is 5 Kbytes and the average size of Asian charmaps is 16 Kbytes.

Installing a Default Codeset for the C Locale

In the standard C/POSIX locale, the default codeset returned is roman8. Users may wish to continue using the C/POSIX locale, but may wish to use a default codeset other than roman8. Applications may take advantage of the codeset as returned via `nl_langinfo` (`CODESET`). For example, VUE (X-windows) uses the codeset in the locale to set up the correct fonts. Thus, if you wish to continue using the C or POSIX locale and also need a codeset value for the C/POSIX locale, you can simply create a link from the C or POSIX locale that you have created to `/usr/lib/nls/loc/locales/C` or `/usr/lib/nls/loc/locales/POSIX`.

WARNING **Please note that the C/POSIX locale will contain only POSIX portable characters and will not provide any international support.**

A commonly used codeset is ISO 8859/1, so a `C.iso88591` locale configured with the codeset value iso88591 has been provided. To use the `C/POSIX` locale with either of these codesets, the super user can install it with `ln(1)`. Here are some examples:

To install the C locale with the ISO 8859/1 codeset:

```
ln /usr/lib/nls/loc/locales/C.iso88591 /usr/lib/nls/loc/locales/C
```

To install the POSIX locale, which is identical to the C locale, with the ISO 8859/1 codeset:

```
ln /usr/lib/nls/loc/locales/C.iso88591 /usr/lib/nls/loc/locales/POSIX
```

Alternatives

To create a C locale with a codeset other than ISO 8859/1 or ROMAN8, then you can use `localedef(1M)` to compile a new `C` locale with the `/usr/lib/nls/loc/src/C.src` and the desired charmap located in `/usr/lib/nls/loc/charmaps`.

Performance

The performance of the C/POSIX locale built into `libc` is slightly faster than using a `C/POSIX` locale installed in `/usr/lib/nls/loc/locales`.

Changes to locales [10.10]

The locale `cs_CZ.iso88592` has been changed. `LC_TIME` `abday` abbreviation for Thursday was incorrect and has been changed from C(acute)t to C(caron)t.

The impact of this change: the abbreviated name for Thursday is different. Applications and commands that retrieve the `ABDAY_5` from the locale will now display a different value for abbreviated Thursday. Applications that use **nl_langinfo**(`ABDAY_5`) to search for the incorrect name or that expect the incorrect name may no longer find a match.

Changes to locales [10.20]

Locale Enhancements:

1. `isprint(3C)` may return true, and `wcwidth(3C)` may return a positive value for the affected codepoints
2. applications that depend on `isprint(3C)` and `wcwidth(3C)` may be affected. For example, `dtterm(1)` will now be able to display these codepoints.

   - `ja_JP.eucJP`

     — CS3 UDC were added to the ctype print class

     <J0101>;...;<J0194>;\
     <J0201>;...;<J0214>;\
     <J0226>;...;<J0233>;\
     <J0237>;...;<J0274>;\
     <J0282>;...;<J0294>;\
     <J0301>;...;<J0394>;\
     <J0401>;...;<J0494>;\
     <J0501>;...;<J0594>;\
     <J0601>;...;<J0664>;\
     <J0670>;\
     <J0672>;\
     <J0675>;\
     <J0677>;...;<J0680>;\
     <J0693>;\
     <J0694>;\
     <J0701>;...;<J0733>;\
     <J0747>;...;<J0781>;\
     <J0801>;...;<J0894>;\
     <J0903>;\
     <J0905>;\
     <J0907>;\

<J0910>;\
<J0914>;\
<J0917>;...;<J0932>;\
<J0949>;...;<J0994>;\
<J1025>;\
<J1088>;...;<J1094>;\
<J1128>;\
<J1136>;\
<J1188>;...;<J1194>;\
<J7801>;...;<J7894>;\
<J1201>;...;<J1294>;\
<J1301>;...;<J1394>;\
<J1401>;...;<J1494>;\
<J1501>;...;<J1594>;\
<J7768>;...;<J7794>;\
<J7801>;...;<J7894>

— NEC VDC CS1 row 13 were added to the ctype print class
<J1301>;...;<J1394>;\

— Additional UDC characters were added to the ctype print class
<J8768>;...;<8794>;\

- `zh_CN.hp15CN`
  — <gb0487>;...;<gb0586>;\
  <gb0827>;...;<gb0832>;\
  — UDC:
  “0xfb3f” - “0xfb7e”
  “0xfc21” - “0xfc7e”
  “0xfd21” - “0xfd7e”
  “0xfe21” - “0xfe7e”
- `ja_JP.SJIS`
  — NEC VDC CS1 row 13 were added to the ctype print class
  <J1301>;...;<J1394>;\
  — additional UDC were added to the ctype print class rows 95-104

Changes to locales [10.30]

1. The NLS methods and locale libraries, and the **iconv(3C)** method libraries have been versioned.
2. Locales created by **localedef(1M)** will not be linked with libc. You will be impacted if:
   a. You have archive internationalized applications and
   b. If you have not been following the instructions to build archived internationalized applications.

New ISO 8859-15 and UTF-8 locales [11.0 patch, 11i]

Fourteen new locales have been created for Euro support based on ISO 8859-15. Refer to Table 2-6 on page 39.

Thirteen new locales have been created for Unicode (and Euro) support based on UTF-8. Refer to Table 2-14, “Base utf8 locales for 32-bit application processing” on page 47, Table 2-15, “European utf8 locales for 32-bit application processing” on page 48 and Table 2-16, “Asian utf8 locales for 32-bit application processing” on page 48.

Thirteen new locale sources are being supplied which include dual currency support based on UTF-8. Refer to Table 2-8, “Supplied utf8 locale sources supporting dual currency” on page 43.

Changed locale and localedef Commands [11i Version 1.5]

To enable smooth migration from PA to IPF, both PA and IPF locale libraries and iconv methods need to exist on an Itanium-based system.

Directory Structure Changes

The current directory structure on 64-bit PA systems is:

```
/usr/lib/nls/loc/locales.1           # PA32 versioned 10.20 locales
/usr/lib/nls/loc/locales.2           # PA32 versioned 11.0 locales
/usr/lib/nls/loc/locales             # Link to /usr/lib/nls/loc/locales.2
/usr/lib/nls/loc/methods.1           # PA32 versioned 10.20 locale method libraries
/usr/lib/nls/loc/methods.2           # PA32 versioned 11.0 locale method libraries
/usr/lib/nls/loc/methods             # Link to /usr/lib/nls/loc/methods.2
/usr/lib/nls/iconv/methods.1         # PA32 versioned 10.20 iconv method libraries
/usr/lib/nls/iconv/methods.2         # PA32 versioned 11.0 iconv method libraries
/usr/lib/nls/iconv/methods           # Link to /usr/lib/nls/iconv/methods.2
/usr/lib/nls/loc/pa20_64/locales.2   # PA64 locales
/usr/lib/nls/loc/pa20_64/locales     # Link to /usr/lib/nls/loc/pa20_64/locales.2
/usr/lib/nls/loc/pa20_64/methods.2   # PA64 locale method libraries
/usr/lib/nls/loc/pa20_64/methods     # Link to /usr/lib/nls/loc/pa20_64/methods.2
/usr/lib/nls/iconv/pa20_64/methods.2 # PA64 iconv method libraries
/usr/lib/nls/iconv/pa20_64/methods   # Link to /usr/lib/nls/iconv/pa20_64/methods.2
```

New directories have been created for 32-bit and 64-bit IPF locale libraries, which co-exist with PA libraries. The new directories are:

```
/usr/lib/nls/loc/hpux32/locales.1   # IA-64 native 32 bit locales
/usr/lib/nls/loc/hpux32/locales     # Link to /usr/lib/nls/loc/hpux32/locales.1
/usr/lib/nls/loc/hpux32/methods.1   # IA-64 native 32 bit method libraries
/usr/lib/nls/loc/hpux32/methods     # Link to /usr/lib/nls/loc/hpux32/methods.1
/usr/lib/nls/iconv/hpux32/methods.1 # IA-64 native 32 bit iconv method libraries
/usr/lib/nls/iconv/hpux32/methods   # Link to /usr/lib/nls/iconv/hpux32/methods.1

/usr/lib/nls/loc/hpux64/locales.1   # IA-64 native 64 bit locales
/usr/lib/nls/loc/hpux64/locales     # Link to /usr/lib/nls/loc/hpux64/locales.1
/usr/lib/nls/loc/hpux64/methods.1   # IA-64 native 64 bit method libraries
/usr/lib/nls/loc/hpux64/methods     # Link to /usr/lib/nls/loc/hpux64/methods.1
/usr/lib/nls/iconv/hpux64/methods.1 # IA-64 native 64 bit iconv method libraries
/usr/lib/nls/iconv/hpux64/methods   # Link to /usr/lib/nls/iconv/hpux64/methods.1
```

Changes to locale Command

Locales are listed according to the underlying processor type if the `-a` option of the `locale` command is specified. The `-a` option is used as follows:

`-a` list 32-bit IPF locales

`-a 32` list 32-bit IPF locales

`-a 64` list 64-bit IPF locales

`-pa32` list 32-bit PA locale libraries in `/usr/lib/nls/loc/locales.2`

`-pa64` list 64-bit IPF locale libraries in `/usr/lib/nls/loc/pa20_64/locales.2`

A new option, `-A` lists all the locale libraries irrespective of the flavors and versions. This list includes locales in the directories

```
/usr/lib/nls/loc/locales.2
/usr/lib/nls/loc/pa20_64/locales.2
/usr/lib/nls/loc/hpux32/locales.1
/usr/lib/nls/loc/hpux64/locales.1
```

Changes to localedef Command

Beginning in 11.0, `localedef` builds both 32-bit and 64-bit locale libraries on 32-bit and 64-bit systems for PA. On Itanium-based systems, both 32-bit and 64-bit libraries will be built for IA only. The generated IA locales are installed as described in “Directory Structure Changes” on page 83.

Compatibility

The default behavior of the `locale` command on IPF systems lists IA locales instead of PA locales. PA applications that list PA locales will see different results on IPF and PA systems.

The `localedef` command will not be able to generate PA locales on an Itanium-based system. If you need PA locales, build them on a PA system and move them to the Itanium-based system.

Building Locales for HP-UX 11.x

HP-UX 11.x systems can have up to three different flavors of the same locale:

- 32-bit
- 64-bit
- HP-UX 10.20 compatibility 32-bit

The following sections present build requirements, briefly describe each flavor, and give the locale building procedure. The locale `zh_CN.utf8` is used as an example.

Requirements

Compiler

Building locales on HP-UX requires a C compiler named "HP C/ANSI C Developer's Bundle for HP-UX". Run the `swlist` command to verify that you have this compiler installed.Normally, this package is NOT shipped with HP-UX systems.

Source Files

All needed files are normally shipped with the system at paths shown below:

charmaps	`/usr/lib/nls/loc/charmaps/utf8.cm`
method files	`/usr/lib/nls/loc/locales/univ.utf8.m`
locale source	`/usr/lib/nls/loc/src/zh_CN.utf8.src`

Kernel Configuration (needed for UTF8 locales only)

Building Unicode/UTF8 locales requires the following kernel parameters to be increased from 67MB to 100MB:

- MAXDSIZ
- MAXTSIZ
- SHMMAX

The three locale flavors

Using the build syntax shown below, locales are built in the local directory. After a locale is built, copy it to the path shown below.

32-bit locales

- Located at `/usr/lib/nls/loc/locales.2/zh_CN.utf8`
- Used by 32-bit HP-UX 11.x applications
- To build, run the following command on a 32-bit machine:

```
localedef -C '+ESlit' -Q -L -x -Pcnf utf8.cm -m univ.utf8.m -i \
  zh_CN.utf8.src zh_CN.utf8 > zh_CN.out 2> zh_CN.err
```

3.2 64-bit locales

- This flavor is found on 64-bit systems only.
- Located at `/usr/lib/nls/loc/pa20_64/locales.2/zh_CN.utf8`
- Used by 64-bit HP-UX 11.x applications.
- To build, run the following command on a 64-bit machine. This produces 32 and 64-bit locales. The 64-bit locale gets created under the `./pa20_64` directory.

```
localedef -C '+ESlit' -Q -L -x -Pcnf utf8.cm -m univ.utf8.m -i \
  zh_CN.utf8.src zh_CN.utf8 > zh_CN.out 2> zh_CN.err
```

3.3 Compatibility 32-bit locales

- Located at `/usr/lib/nls/loc/locales.1/zh_CN.utf8`.
- Used by HP-UX 10.20 applications.
- To build, run the following command on a 10.20 machine then copy the locale binary to the 11.x system:

```
localedef -C '+ESlit' -Q -L -x -Pcnf utf8.cm -m univ.utf8.m -i \
  zh_CN.utf8.src zh_CN.utf8 > zh_CN.out 2> zh_CN.err
```

Alternatives

- There are options to build locales and get them installed immediately in the default locations. Read the localedef manual on this option.
- Locales under `/usr/lib/nls/loc/locales.1` can be copied from a 10.20 machine.

4 Asian System Environment (ASE)

ASE is the system for these four Asian countries:

JSE	Japanese System Environment
KSE	Korean System Environment
SSE	Simplified-Chinese System Environment
TSE	Traditional-Chinese System Environment

ASE - Changes [10.30]

Features

ASE Common

- Starbase fonts are included in both Server and Workstation systems. They were once deleted from Server systems at A.02.00, but are now again included in this release.

JSE

- Japanese TrueType outline font can be used when you buy and install the optional TrueType font sold by the Japanese font vendor Ricoh. The supported TrueType product of Ricoh is described in the JSE Release Notes or JSE User's Guide (available online or hardcopy).
- FA/FM fonts are included in both Server and Workstation systems. They were once deleted from Server systems at A.02.00, but are now again included in this release. Note that FA/FM fonts are planned to be removed in the next release.
- The following printers are added to LIPS printer support:

Table 4-1 New printers added to LIPS printer support

Printer	lp model	DPS model
Canon LBP-830	LIPS4	LBP-830
Canon LBP-450	LIPS4	LBP-450

- The following printer is added to PS printer support only for HP Distributed Print Service (HPDPS). It supports notification of failure by bi-directional communication.

Table 4-2 New printer added to PS printer support for HPDPS

Printer	lp model	DPS model
Fuji Xerox DP300	N/A	DP300

- JSE supports printing via X Print Server to LIPS, PostScript, and PCL printers.

KSE

The XDevice User's Guide is combined with the Korean System Environment User's Guide.

SSE

`sconv`, one tool of Simplified Chinese Toolkit (STK), provides two new code conversions: GBK to Big5 and Big5 to GBK.

Summary of Changes

ASE Common

- The path names of ASE online release notes and JSE Printer `README` file are changed as follows:

Table 4-3 New Path names for ASE online release notes and JSE Printer README file

Old	New
/usr/newconfig/RelNotes/ASX-JPN	/usr/share/doc/ASX-JPN
/usr/newconfig/RelNotes/ASX-JPN-E	/usr/share/doc/ASX-JPN-E
/usr/newconfig/RelNotes/ASX-JPN-S	/usr/share/doc/ASX-JPN-S
/usr/newconfig/RelNotes/ASX-KOR	/usr/share/doc/ASX-KOR
/usr/newconfig/RelNotes/ASX-SCH	/usr/share/doc/ASX-SCH
/usr/newconfig/RelNotes/ASX-TCH	/usr/share/doc/ASX-TCH
/usr/newconfig/RelNotes/PRINTER-JPN-E	/usr/share/doc/PRINTER-JPN-E
/usr/newconfig/RelNotes/PRINTER-JPN-S	/usr/share/doc/PRINTER-JPN-S

- Because VUE is deleted from 10.30, VUE features of ASE are not supported.

JSE

- Because `SharedPrint` is deleted from 10.30, `SharedPrint` is no longer supported by JSE.
- Japanese outline fonts FontWave, provided by ALPS, is deleted. The tools for FontWave are also deleted.

KSE

- The C1205A printer model file and the C1205A printer are no longer supported.

Impact

JSE

- If you use FontWave fonts, you are impacted. You can use the TrueType option of Ricoh instead of FontWave fonts.
- Printing applications that use Font Server access Library (`FSlib`) provided by JSE should be modified to use the X Print Server.
- If you print via SharedPrint, you are impacted. The lp spooler or HPDPS can be used instead.

KSE

- The C1205A printer is no longer supported.

Compatibility

JSE

- The font names (XLFDs) are different, but there are font aliases from FontWave font names to TrueType fonts, so applications can use existing FontWave XLFDs.

Alternatives

JSE

- Instead of the deleted FontWave, you can buy TrueType optional fonts from Ricoh to use Japanese online fonts.
- Instead of SharedPrint, use lp spooler or HP DPS.

KSE

- The C1205A printer is not supported; instead, use newer and supported printers. Refer to the KSE User's Guide to see what printers are supported.

ASE - Changes [11.0]

The following information is common to each of these environments:

- `ximsstart` is replaced with `dtimsstart` on HP-UX 11.0 CDE. With this change, `XimsMode` in "`Application Manager - Desktop_tools`" on CDE is replaced with `DtImsMode`.
- The command `/usr/bin/X11/ximsmode` has been removed at HP-UX 11.0. No alternative is provided.
- The following HP proprietary old locale names are no longer supported.

  `japanese`

  `japanese.euc`

  `korean`

  `chinese-s`

  `chinese-t`

  `chinese-t.big5`

  Instead of HP proprietary old locale names, use the following locales:

Table 4-4 New Asian locale names

Old	New
`japanese`	`ja_JP.SJIS`
`japanese.euc`	`ja_JP.eucJP`
`korean`	`ko_KO.eucKR`
`chinese-s`	`zh_CN.hp15CN`
`chinese-t`	`zh_TW.ccdc`
`chinese-t.big5`	`zh_TW.big5`

- HP proprietary old X11 fonts are no longer supported. These X11 fonts under the following directories have been deleted:

  `/usr/lib/X11/fonts/hp_japanese/75dpi`

`/usr/lib/X11/fonts/hp_korean/75dpi`

`/usr/lib/X11/fonts/hp_chinese_s/75dpi`

`/usr/lib/X11/fonts/hp_chinese_t/75dpi`

If you use the old X11 fonts, you are impacted. Use the new X11 fonts instead of Intellifont.

Instead of HP proprietary old X11 fonts, use the new fonts under the following directories:

`/usr/lib/X11/fonts/hp_japanese/100dpi`

`/usr/lib/X11/fonts/hp_korean/75dpi`

`/usr/lib/X11/fonts/hp_chinese_s/75dpi`

`/usr/lib/X11/fonts/hp_chinese_t/75dpi`

- Asian Starbase fonts have been deleted.
- *udcsc* (1) - The tool for User Defined Characters in Starbase stroke fonts has been deleted.
- Open NLIO library (`libnlio`) has been deleted. Instead of Open NLIO library (`libnlio`), use X11R5 Ximp or X11R6 XIM protocol.
- Printers in the LaserJet II Series are no longer supported. Instead of LaserJet II Series printers, use printers from the LaserJet III, 4, and 5 Series.
- Printing Asian PCL files to LIPS, PS, ESC/P, and Japanese PCL printers is no longer supported. Thus, you can no longer print Asian PCL files using the following model files as of HP-UX Release 11.0:

Table 4-5 Obsolete model files

Model file	Option
`LIPS3`	`-opcl`
`LIPS4`	`-opcl`
`PS.nlio`	`-opcl`
`ESCP`	`-opcl`
`PCL5.asian`	`-oapcl`

- Various printing options supporting the LP spooler and HPDPS have been removed as of this release. For details, see the HP-UX file `/usr/share/doc/PRINT-ASE-NOTE`.

JSE (The Japanese System Environment)

- The following printers are now supported:

Table 4-6 Newly supported printers in JSE

Printer	LP model	DPS model
Canon LBP-730PS	PS.nloo	LBP-730PS
Canon LBP-740	LIPS4	LBP-740
Canon LBP-750	LIPS4	LBP-750
Canon LBP-930	LIPS4	LBP-930
Oki Microline 900PSII LT	PS.nloo	ML900PSIILT
Oki Microline 902PSII	PS.nloo	ML902PSII
Oki Microline 903PSII	PS.nloo	ML903PSII
Oki Microline 903PSII+F	PS.nloo	ML903PSII+F
Oki Microline 905PSII+F	PS.nloo	ML905PSII+F
EPSON VP-2200	ESCP	VP-2200
EPSON VP-4200	ESCP	VP-4200
EPSON VP-5200	ESCP	VP-5200

- 106/109 Keyboards have been added to keyboard support.
- Some of UDC files have been removed, renamed, and resized.
- Japanese Intellifont is no longer supported.

  If you use Intellifonts, you are impacted. Use the TrueType option of Ricoh instead.
- Japanese FA/FM fonts are no longer supported.
- Xsi protocol of XJIM is no longer supported.

- FSlib - Font Server Access Library is no longer supported.

  Use the X Print Server instead.

- *jtos* (1) - the code conversion commands `jtos`, `jtou`, `stoj`, `stou`, `utoj` and `utos` are no longer supported.

  Instead of *jtos* (1), use *iconv* (1) as follows:

```
jtos: iconv -f jis   -t sjis
jtou: iconv -f jis   -t eucJP
stoj: iconv -f sjis  -t jis
stou: iconv -f sjis  -t eucJP
utoj: iconv -f eucJP -t jis
utos: iconv -f eucJP -t sjis
```

- The following function groups in the Japanese library (`libjpn`) are not supported:

  jcode (3X), *ibmjcode* (3X), *jisconv* (3X), *jctype* (3X), *jwctype* (3X), *jconv* (3X), *jwstring* (3X), *cset* (3X), *getwidth* (3X), *euclen* (3X), *wstostr* (3X)

  Instead of the Japanese library, use *iconv* (3C), *wctype* (3C).

- `dictmerge` - the dictionary merge tool for XJIM is not supported.

- The Kana Kanji conversion library (`libjpn`) is not supported.

  Instead of the conversion library, use Xlib or OSF/Motif.

- Printing applications that use Font Server access Library (`FSlib`) provided by JSE should be modified to use the `X Print Server`.

KSE (Korean System Environment)

No Korean-specific changes for this release.

SSE (Simplified-Chinese System Environment)

No Simplified-Chinese-specific changes for this release.

TSE (Traditional-Chinese System Environment)

`tconv`, a tool of Traditional Chinese Toolkit (TTK), provides two new code conversions: GBK to Big5 and Big5 to GBK.

Enhanced Print Capabilities in ASE [11.0 patch, 11i]

This release contains enhancements to the printer capabilities of four Asian-country system environments (JSE, KSE, SSE, TSE), as itemized below.

Changes Common to all ASEs

- LP Model File: Support new printers: `PCL5.nloo` model file supports Asian text printing on following printers.
  - — HP LaserJet 4000
  - — LaserJet 5000
  - — LaserJet 8000
- HPDPS: Provide common printer model directories: Provide new printer model directories, `PCL4.asx`, `PCL5.asx` and `ESCP.asx` for future printer support. Users can use these model directories as “model” or “sample” implementation of a printer-model. The user can copy these sample printer model directories to a directory under `/var/opt/pd/lib/model` with an appropriate name and customize it to be suited for the printer being configured.

  Support new printers: User can print Asian text on following printers through HPDPS by configuring the printer with `PCL5.asx` printer-model.
  - — HP LaserJet 4000
  - — LaserJet 5000
  - — LaserJet 8000

For more information, see the following files in `/usr/share/doc/`:

`ASX-JPN`, `ASX-JPN-S`, `ASX-JPN-E`, `ASX-KOR`, `ASX-SCH`, `ASX-TCH`

Japanese System Environment (JSE)

- `LP` model file: Support new printers. `PS.nlio` model file supports Japanese text printing on these printers:

— OKI ML703N

— ML600PSII

`ESCP` model file supports Japanese text printing on these printers:

— OKI 5330S

— 8350S

— 8580S

— EPSON VP-1800

`PCL5.asian` model file supports Japanese text printing on:

— HP LaserJet 5Si with 2Byte Font SIMM

— LaserJet 4000 with 2Byte Font DIMM

— LaserJet 5000 with 2Byte Font DIMM

— LaserJet 8000 with 2Byte Font DIMM

- HPDPS: Provide common printer model directories: Provide new printer model directories, `LIPS3.asx, LIPS4.asx, PS.asx` and `2BPCL5.asx` for future printer support. User can use these model directories as “model” or “sample” implementation of a printer-model. User may copy these sample printer model directories to a directory under `/var/opt/pd/lib/model` with an appropriate name and customize it to suit the printer being configured.

  Support new printers: User can print Japanese text on the following printers through HPDPS, by configuring the printer with 2BPCL5.asx printer-model:

  — HP LaserJet 5Si with 2Byte Font SIMM

  — LaserJet 4000 with 2Byte Font DIMM

  — LaserJet 5000 with 2Byte Font DIMM

  — LaserJet 8000 with 2Byte Font DIMM

  Users can print Japanese text on following printers through HPDPS by configuring the printer with PS.asx printer-model:

  — OKI ML703N

  — ML600PSII

The user can print Japanese text on following printers through HPDPS by configuring the printer with `ESCP.asx` printer-model:

— OKI 5330S

— 8350S

— 8580S

— EPSON VP-1800

For more information, see the following files in `usr/share/doc/`: `ASX-JPN`, `ASX-JPN-S`, `ASX-JPN-E`, `PRINTER-JPN-S`, `PRINTER-JPN-E`

Korean System Environment (KSE)

- X Print Server: KSE support printing via X Print Server to PCL printers.
- LP and HPDPS: Support new print options. Support new printers.
- HPDPS: Provide common template model directory for each print language.

For more information, see the following file: `/usr/share/doc/ASX-KOR`.

Simplified-Chinese System Environment (SSE)

- X Print Server: SSE support printing via X Print Server to PCL printers
- LP and HPDPS: Support new print options. Support new printers.
- HPDPS: Provide common template model directory for each print language.

For more information, see the following file: `/usr/share/doc/ASX-SCH`.

Traditional-Chinese System Environment (TSE)

- X Print Server: TSE support printing via X Print Server to PCL printers
- LP and HPDPS: Support new print options. Support new printers.
- HPDPS: Provide common template model directory for each print language.

For more information, see the file `/usr/share/doc/ASX-TCH`.

ASE - Changes [11i and 11i Version 1.5]

HP-UX provides several Asian enhancements as server features, including some new Asian codesets, UDC (User Defined Characters, or Gaiji), printing, and codeset conversions with mainframe codesets.

The new, changed, deleted features as well as some troubleshooting information is described below. For further information, see the following documentation:

- JSE
  - *Japanese System Environment User's Guide* (B3782-90873)
  - *HP XJIM Japanese Input Method Guide* (B3782-90869)
  - *ATOK8 Japanese Input Method Guide* (B3782-90870)
  - *EGBridge Japanese Input Method Guide* (B3782-90871)
  - *VJE-γJapanese Input Method Guide* (B3782-90872)
- *KSE - Korean System Environment User's Guide* (5969-4454)
- *SSE - Simplified Chinese System Environment User's Guide* (5969-4455)
- *TSE - Traditional Chinese System Environment User's Guide* (5969-4453)

To get release information on earlier versions of ASE, see the following files:

- JSE: `/usr/share/doc/ASX-JPN`
- KSE: `/usr/share/doc/ASX-KOR`
- SSE: `/usr/share/doc/ASX-SCH`
- TSE: `/usr/share/doc/ASX-TCH`

New Features

- ASE Common
  - ❏ New printer model

    New printer models are supported on both the LP Spooler and HPDPS. You can print plain text file on the following printers by configuring the printer using the `PCL5.nloo` (`PCL5.asian`) model file on the LP Spooler or `PCL5.asx` (`2BPCL5.asx`) printer model on HPDPS:

HP LaserJet 4000(N)
HP LaserJet 4050(N)
HP LaserJet 4500(N)
HP LaserJet 5000(N)
HP LaserJet 8000(N)
HP LaserJet 8100N

NOTE By installing optional Font DIMM on these printers, you can print text with TrueType fonts. To use TrueType fonts, you have to configure a printer with `PCL5.asian` model file for the LP Spooler, or with `2BPCL5.asx` printer model for HPDPS.

NOTE HPDPS related features are not included in HP-UX 11i Version 1.5.

- ❏ HPDPS common printer model directory [not applicable for 11i Version 1.5]

  For HPDPS, the common printer model directories `PCL5.asx`, `2BPCL5.asx` and `ESCP.asx` are provided for future new printer support. The user can copy these sample printer model directories to a directory under `/var/opt/pd/lib/model` with an appropriate name and customize it to be suited for the printer being configured.

- JSE

  - ❏ ATOK X for HP-UX Preview Edition [not applicable for 11i Version 1.5]

    The new version of ATOK is now supported. As a Kana-Kanji conversion feature, the ATOK12 engine is incorporated enabling you to achieve a comfortable and effective Japanese input environment. As this release of ATOK X is a Preview Edition, some of the customization tools are not yet available. In the next release, a full featured ATOK X for HP-UX will be provided.

  - ❏ Unicode

    Japanese UTF-8 locale `ja_JP.utf8` is supported. Using this locale, you can input, display and print UTF-8 characters. It supports characters defined in standards JIS X 0201 (1976), JIS X 0208 (1990), and JIS X 0212 (1990). UDC (User Defined Characters or GAIJI) and VDC (Vender Defined Characters) are not supported.

For details, see the document `/usr/share/doc/ASX-UTF8`.

- USB (Universal Serial Bus) Japanese 109 Keyboard support

  This allows for inputting Japanese characters by Japanese input methods.

- NEC VDC symbols for display on X Window System

  NEC special characters are included in Japanese fonts. NEC VDC has 83 characters which occupy following code areas:

  JIS[Kuten]: 13/01 - 13/92

  Shift-JIS: 0x8740 - 0x879C

  Those characters can be shown on X Window System.

- New Ricoh TrueType font package

  The new Ricoh TrueType font package “TrueTypeWorld ValueFontD2” is supported. The supported fonts are Windows 3.1 version of WABUN (Japanese) fonts.

- New printer model

  New printer models are supported on both the LP Spooler and HPDPS. You can print Japanese plain text file on the following printers by configuring the printer using the specified model file on the LP Spooler or printer model on HPDPS:

Table 4-7 New Printer Models for JSE

Printer	LP Spooler Model File	HPDPS Printer Model File [not for 11i Version 1.5]
HP LaserJet 5si[a]	`PCL5.nloo (PCL5.asian)`	`PCL5.asx (2BPCL5.asx)`
HP HITPCPDA	`ESCP`	`ESCP.asx`
HP HITHTS4A	`ESCP`	`ESCP.asx`
HP HITKD20A	`ESCP`	`ESCP.asx`
HP HITKD45A	`ESCP`	`ESCP.asx`
Canon LBP-850	`LIPS4`	`LIPS4.asx`
Canon LBP-930EX	`LIPS4`	`LIPS4.asx`

Table 4-7 New Printer Models for JSE

Printer	LP Spooler Model File	HPDPS Printer Model File [not for 11i Version 1.5]
Canon LBP-2030	LIPS4	LIPS4.asx
Canon LBP-2040	LIPS4	LIPS4.asx
Canon LBP-2160	LIPS4	LIPS4.asx
OKI Microline 9XXPSII[b]	PS2.nlio	PS2.asx
OKI Microline 9XXPSIII[b]	PS2.nlio	PS2.asx
OKI Microline 703N(3)[b]	PS2.nlio	PS2.asx
EPSON VP-1800	ESCP	ESCP.asx
OKI 533OS	ESCP	ESCP.asx
OKI 835OS	ESCP	ESCP.asx
OKI 858OS	ESCP	ESCP.asx
NEC LL-15 (NPDL2)	NPDLII	NPDLII
NEC LL-30 (NPDL2)	NPDLII	NPDLII
NEC LL-15 (ESC/P)[c]	ESCP	ESCP.asx
NEC LL-30 (ESC/P) [c]	ESCP	ESCP.asx

a. By installing optional Japanese Font DIMM on these printers, you can print Japanese text with TrueType fonts. To use Japanese TrueType fonts, you have to configure a printer with `PCL5.asian` model file for the LP Spooler. To see whether your printer has Japanese TrueType Font installed, follow these steps:

Press **Menu** on the control panel of the printer until "`INFORMATION MENU`" appears.
Press **Item** until "`PRINT PCL FONT LIST`" appears.
Press **Select** to print the font list.
If your printer has Japanese TrueType font, you will see "`MS Mincho`" and "`MS Gothic`" in the printed list.

b. Printing text files on expanded A3 (called "A3-Nobi" in Japan) paper is not supported.

c. There are restrictions of page length setting on ESC/P mode. For detail, see manual of the printer and online document `/usr/share/doc/PRINTER-JPN-S[E]`.

- HPDPS common printer model directory [not applicable for 11i Version 1.5]

  For HPDPS, the common printer model directories `LIPS3.asx`, `LIPS4.asx` and `PS.asx` are provided for future new printer support. The user can copy these sample printer model directories to a directory under `/var/opt/pd/lib/model` with an appropriate name and customize it to be suited for the printer being configured.

- Mainframe code set conversion [not applicable for 11i Version 1.5]

  The Mainframe code set conversions are provided to convert code sets between Mainframe code sets Hitachi KEIS, NEC JIPS, Fujitsu JEF, and IBM EBCDIC with existing code sets SJIS, eucJP, and ucs2. These code conversions are used by `iconv(1)` and `iconv(3C)`.

  The following code sets are supported:

  - Hitachi KEIS
    - keis7k: KEIS78 (Hitachi MF code set based on JIS C6226-1978) + EBCDIK
    - keis8k: KEIS83 (Hitachi MF code set based on JIS X0208-1983) + EBCDIK
    - keis7c: KEIS78 (Hitachi MF code set based on JIS C6226-1978) + EBCDIC
    - keis8c: KEIS83 (Hitachi MF code set based on JIS X0208-1983) + EBCDIC
  - NEC JIPS
    - jipsj: JIPS (NEC Mainframe code set) JIS
    - jipsec: JIPS (NEC Mainframe code set) EBCDIC
    - jipsek: JIPS (NEC Mainframe code set) EBCDIK
  - Fujitsu JEF
    - jefc: JEF (Fujitsu Mainframe code set) + EBCDIC (lower alphabet)
    - jefk: JEF (Fujitsu Mainframe code set) + EBCDIK (katakana)
    - jefc9p: JEF + EBCDIC designating 9 point size in printing
    - jefk9p: JEF + EBCDIK designating 9 point size in printing

The code set conversions are provided between the above Mainframe code sets and the following existing code sets:

SJIS

eucJP

ucs2

- ❏ New UDC feature [not applicable for 11i Version 1.5]

  A new UDC environment is provided for client/server or distributed environments. You can share UDC font on a single server machine and print UDC from client machines. As a UDC font, TrueType font is supported. You can use UDC TrueType font created on X Window or provided by some vendors. Two typefaces are supported as UDC fonts. ESC/P and PCL printers are supported.

- KSE

  - ❏ Unicode

    The Korean UTF-8 locale `ko_KR.utf8` is supported. On this locale, you can input, display and print UTF-8 characters. There is support for characters defined in standards KSC 5636 (1989) and KSC 5601 (1987). UDC (User Defined Characters or GAIJI) and VDC (Vender Defined Characters) are not supported. For details, see the document `/usr/share/doc/ASX-UTF8`.

    The full Hangul Syllables in KS X 1005-1 (old name is KS C 5700-1995) are supported on `ko_KR.utf8` locale. You can input full Hangul characters by XKIM and display on X Window System. With Korean font DIMM and `PCL5.asian` model file, you can print full Hangul characters.

  - ❏ Euro and registered trademark ® symbols

    The printing of the Euro symbol in the `ko_KR.eucKR` locale is supported. The registered trademark symbol ® is also supported. PCL printers are supported to print these symbols with `PCL5.asian` model file. Two typefaces, Dotum and Batang, are supported. You can print Euro and ® symbols without any printing options.

  - ❏ USB (Universal Serial Bus) Korean 106 Keyboard

    USB Korean 106 Keyboard is supported for inputting Korean characters by Korean input method XKIM.

- X Print Server

  KSE supports printing via the X Print Server to PCL printers.

- SSE

  - Unicode

    Simplified Chinese UTF-8 locale `zh_CN.utf8` is supported. On this locale, you can input, display and print UTF-8 characters. There is support for characters defined in standards ISO 646 (1991) and GB 2312 (1980). UDC (User Defined Characters or GAIJI) and VDC (Vender Defined Characters) are not supported. For details, see the document `/usr/share/doc/ASX-UTF8`.

  - USB (Universal Serial Bus) Simplified Chinese 104 Keyboard

    The USB Simplified Chinese 104 Keyboard is supported for inputting Simplified Chinese characters by the input method XSIM.

  - X Print Server

    SSE supports printing via the X Print Server to PCL printers.

- TSE

  - Unicode

    Traditional Chinese UTF-8 locales `zh_TW.utf8` and `zh_HK.utf8` are supported. On these locales, you can input, display and print UTF-8 characters. There is support for characters defined in standards ISO 646 (1991), CNS 11643 (1992) plane 1, 2, 3 and 4, except for some characters which are not supported by Unicode 2.0. UDC (User Defined Characters or GAIJI) and VDC (Vender Defined Characters) are not supported. For details, see the document `/usr/share/doc/ASX-UTF8`.

  - USB (Universal Serial Bus) Traditional Chinese 104 Keyboard

    USB Traditional Chinese 104 Keyboard is supported for inputting Traditional Chinese characters by the input method XTIM.

  - X Print Server

    TSE supports printing via the X Print Server to PCL printers.

  - HongKong big5 Support

    Locale support is provided with the big5 codeset for HongKong.

HP provides support for the HongKong big5 locale `zh_HK.big5`. HongKong big5 locale is similar to Traditional Chinese big5 locale. The difference between these two locales are in monetary and date/time properties which reflect local cultural conventions.

CDE has been enhanced to support this new locale by providing the required app-defaults files to CDE applications.

Impact

Applications must elect to enable big5 support by setting the `LANG` and/or `LC_*` environment variables to the HongKong big5 locale.

The size requirement for locale source and binaries is 1.7 MB

Applications using HongKong big5 locales should see the same performance as of Traditional Chinese big5.

Changed Feature

- JSE
  - ❏ 'EISUU' key mode change for 106/109 keyboard

    In the previous version, 'EISUU' key, 'Shift + EISUU (Caps Locks mode)' keys, and 'Alt + EISUU (KANJIBANGOU mode)' keys all worked as 'Caps Lock'. Now they work as original features of the key/keys.

Deleted Features

- ASE Common
  - ❏ Printing to LaserJet III series is now obsoleted. If you are currently using LaserJet III series printers, you should use newer printer models.
  - ❏ HPDPS related features are not included in HP-UX 11i Version 1.5.
- KSE
  - ❏ XDevice is not included from this release.

NOTE The Japanese input methods EGBridge and VJE-γ will be obsoleted in an upcoming release.

Troubleshooting Information

- JSE
  - ❑ XJIM
    - — On a low-resolution display, the customize window is cut off by default. Specify 14-dot font with `-fn` option or `XJim*fontList` resource.
    - — If you use 'KANA' input (not 'ROMAJI' input) as the key input method at 'YOMI' input, and you input a 'KANA' character and 'HANDAKUTEN' or 'DAKUTEN' successively, the input method server does not compose 'KANA' with 'DAKUTEN' or 'HANDAKUTEN' as one character, but displays the 'KANA' character and 'DAKUTEN' or 'HANDAKUTEN' symbol. In this case, you should make the composite character using 'ZENKAKU-HIRAGANA' conversion (press **Shift** + **F5** key), or 'ZENKAKU-KATAKANA' conversion (press **F6** key).
    - — If you install XJIM after NIS configuration, you will find that you can not use XJIM Conversion Server. To resolve this problem, move the following line in the `/etc/services` file

      ```
      nuekks          6897/tcp            # nuekks daemon
      ```

      to the position above the line which begins with a "+" sign indicating the start of NIS mapping.
  - ❑ EGBridge

    Closing the EGBridge main window during Kana-Kanji conversion on `hpterm` may also close `hpterm`. You should finish conversion before closing the EGBridge main window.
  - ❑ IMS common (XJIM/ATOK8/EGBridge/VJE-γ)
    - — Window focus sometimes cannot be moved by **Meta**(**Alt**)-**Tab** key if applications use `XIMStatusNothing` and they overlap each other with KANJI-ON state. To avoid this problem, set the `stackChange` resource to `False` as follows:

XJIM	`XJim*stackChange: False`
ATOK8	`Atok8*stackChange: False`
EGBridge	`EGIms*stackChange: False`
VJE	`Vje*stackChange: False`

See the “Resource” section in each Input Method manual for details.

— On Motif 1.2 and Motif 2.1 applications, the **F10** and **Shift-F10** keys cannot be used as the Japanese input function key because those keys are used to switch focus to the menu bar. To assign these keys to certain functions for IMS, set the following:

— for DIN keyboard: `$xmodmap -e "keycode 25 = F10"`
— for ITF keyboard: `$xmodmap -e "keycode 38 = F10"`

— Japanese IMS is not available with X11R4 (including Motif 1.1) applications using PS2-DIN-JIS keyboard if `LANG` is `ja_JP.SJIS` or `ja_JP.eucJP`. To avoid this problem, set `LANG` to `japanese` or `japanese.euc` when invoking X11R4 (Motif 1.1) applications.

— Even if you merge UDC in X font after running the input method server, the server cannot display UDC in the pre-edit and the candidate. You should merge UDC in X font server before running the input method server. Re-login makes sure that the input method server displays UDC on CDE.

❏ JIS keyboard

— Do not set the `KBD_LANG` shell variable or Motif 1.1 applications will not work with a JIS keyboard.

— The **Yen** key on JIS keyboard with X terminal does not work correctly. To use the **Yen** key, execute the command.

```
$ xmodmap -e "keysym yen = backslash bar prolongedsound"
```

❏ 106/109 Keyboard

— You cannot turn off `EGBridge` (although you can turn on). The solution is to change the key map file `$HOME/.gab/EGBMap` (for personal use) or `/etc/opt/egb/config/EGBMap` (for system use). You open the key map file with an editor and change the following entry:

old: `LKONOFF = XK_Henkan XK_Meta_L`
new: `LKONOFF = XK_Henkan XK_Meta_L XK_Alt_L`

Then save the updated key map file and restart EGBridge. You can turn EGBridge on/off with the left **Alt** key.

- ❏ udcload
  - — When UDCs are not arranged in the code order in the UDC file, udcload cannot load UDC. Therefore. you should arrange UDCs in the code order. UDCs generated by xudced have no problem because xudced generates UDSs arranged in code order.

- KSE
  - ❏ xk0input

    Xkim is not available with X11R4 (including Motif 1.1) applications using PS2-DIN keyboard if LANG is ko_KR.eucKR. To avoid this problem, set LANG to korean when invoking X11R4 (Motif 1.1) applications.

- ASE Common
  - ❏ xudced (UDC editor)

    When you select Search... in the Edit menu, you cannot specify the character directly. Only the Index number can be specified to search a character.

Software Availability in Native Languages

The commands used with this product are the ones supported by the Native Language Support Catalog of HP-UX.

Command and Library Support [not applicable for 11i]

To ensure smooth migration from PA to IPF systems, both PA and IPF locale libraries and iconv methods need to be present on the IPF system. See “Changed locale and localedef Commands [11i Version 1.5]” on page 83.

5 Conformance [10.0]

X/Open Portability Guide (XPG4)

New / Modified Functions

iconv(3C)

The **iconv(3C)** function converts the sequence of characters from one codeset, in the array specified by inbuf, into a sequence of corresponding characters in another codeset, in the array specified by outbuf.

```
size_t iconv (iconv_t cd, const char **inbuf, \
size_t *inbytesleft, char **outbuf, size_t *outbytesleft);
```

iconv_close(3C)

The **iconv_close(3C)** function deallocates the conversion descriptor cd and all other associated resources allocated by the **iconv_open(3C)** function.

```
int iconv_close (iconv_t cd);
```

iconv_open(3C)

The `iconv_open(3C)` function returns a conversion descriptor that describes a conversion from the codeset specified by the string pointed to by the `fromcode` argument to the codeset specified by the string pointed to by the `tocode` argument.

```
iconv_t iconv_open (const char *tocode, const char *from code);
```

localedef(1M)

The command **localedef(1M)** has undergone massive redesign and change. Please refer to the manpage for a complete description. The information below is only a summary of changes.

New supported options:

`-C`	Compiler options. Passes the specified options to the compiler used to build the locale.
`-L`	Linker option. Passes the specified options to the linker used to build the locale.

`-m`	Method source file. Specifies a method source file.
`-v`	Displays addition information for debugging.
`-w`	Displays duplicate definition warnings.

The following options have been removed:

`-d`	Dump locale is no longer needed as locale description source files are provided on the system (under `/usr/lib/nls/loc/src`). Please see comments under the compatibility section.
`-o`	Not needed since the `-d` option is no longer supported.

The grammar recognized by localedef is now fully XPG4 / POSIX compliant. Changes in grammar for compliance are as follows:

- Overall: characters in portable codeset no longer need to be quoted (e.g., A need not be specified as 'A'); use of charmap files are now the recommended approach.
- Header section: `version`, `hpversion` no longer supported.
- `LC_ALL` category: the following items in this category have been moved to `LC_CTYPE` (`context`, `direction`).
- `LC_CTYPE` category: `bytes_char`, `code_scheme`, `cswidth` no longer supported.
- `LC_COLLATE` category: `modifier` no longer supported
- `LC_MONETARY` category: `crncystr` no longer supported

Summary of Changes. Locales are shared objects in HP-UX 10.0 (as opposed to binary files in HP-UX 9.x). This facilitates an object-oriented approach to locale creation: inclusion of data and methods in a locale. Methods are essentially customized code to handle particular characters and encoding scheme.

Note that for users with statically bound executables who wish to change their customized locales, the HP-UX 10.0 version of **localedef(1M)** (under `/usr/bin`) cannot be used. Instead the HP-UX 9.x version (under `/usr/old/usr/bin`) should be used.

Impact. Due to grammar changes, customized locales description files will require modifications; charmap files will be needed. Examples will be provided on the system.

Please see “Build Change For Archive Internationalized Applications” on page 64 for details regarding build changes.

The old implementation of **localedef(1M)** will be provided under `/usr/old/usr/bin`. However, this will be a minimum functionality version. Specifically, the **localedef(1M)** under `/usr/old/usr/bin` will only be capable to dump a locale (-d option). This functionality is provided for users who have customized locales without a corresponding locale description (source) file. In such cases, the customized locales can be dumped into a source file, which can subsequently be modified for handling with the new **localedef(1M)**. Due to the vast amount of grammar changes, it is recommended that HP-UX 10.0 customized locales be based on 10.0 locale sources; i.e., 9.x customizations should be applied to the 10.0 sources. In most cases this will be easier than modifying a 9.x locale source for compilation with the 10.0 **localedef(1M)**.

For example, if one wishes to recreate a customized locale which in 9.x was based on the german locale, the following steps are recommended:

1. Obtain a dump of the customized locale:

   ```
   localedef -d customized_locale_name > ldf1
   ```

2. Obtain a dump of the closest standard locale (german in this case):

   ```
   localedef -d german > ldf2
   ```

3. Determine the changes between the previous two:

   ```
   diff ldf1 ldf2
   ```

4. Use the HP-UX 10.0 german locale description file (`/usr/lib/nls/loc/src/de_DE`) and apply the changes that were identified in the previous step.

Additional Modifications. Due to grammar changes, customized locales will require the following modifications:

- charmap files will need to be created.
- Some of the more mechanical changes (e.g., conversion of 'A' to A) can be achieved by using a sed script (details to provided later).
- Unrecognized keywords will have to be removed.
- `LC_ALL` category should be entirely removed. Keywords previously located in this category have been moved to other categories.

- The more complicated changes (e.g., `LC_COLLATE` category) have to be made manually.

If the source files for the HP-UX 9.x customized locales are not available, a special 9.x-based **localedef(1M)** is provided under `/usr/old/usr/bin`. See "Impact" on page 113.

Compatibility. HP-UX 9.x locales are currently being supported in HP-UX 10.0 for backwards compatibility of applications compiled archive in 9.x. Applications compiled shared in 9.x will use the 10.0 locales.

Limitations. List of known problems:

- Hard-coded characters should be avoided. Use charmap file to map them to a symbol and use the symbol instead.

Performance. The performance of localedef itself is not expected to change.

Potential Future Directions. The HP-UX 9.x locales and `localedef(1M)` in `/usr/old/usr/bin` may not be supported in the next major release.

strfmon(3C)

The **strfmon(3C)** libc function is now supported to convert monetary values to string. It places the characters into the array pointed to by `s` as controlled by the string pointed to by `format`. No more than `maxsize` bytes are placed into the array.

```
size_t strfmon (char *s, size_t maxsize, const char *format, ...);
```

Alternatives: See **localeconv(3C)** and **nl_langinfo(3C)**.

strptime(3C)

The **strptime(3C)** libc function is now supported to perform date and time conversion. It converts the character string pointed to by `buf` to values which are stored in the tm structure pointed to by `tm`, using the format specified by `format`.

```
char *strptime (const char *buf, const char *format, struct tm *tm);
```

Alternatives: See **strftime(3C)** and **nl_langinfo(3C)**.

Program Messaging

LC_MESSAGES Support for Messaging

catopen(3C) now supports XPG4 messaging which means that if the `oflag` parameter to **catopen(3C)** is set to `NL_CAT_LOCALE`, the directory path used to search for the message catalog will be based on `LC_MESSAGES` rather than `LANG` if `oflag` is set to 0.

To provide XPG4 messaging, an application should first call **setlocale(3C)** with the category set to `LC_ALL` for most cases and `LC_MESSAGES`, if the locale database if not being used for anything other than messaging. Next, set the oflag parameter of **catopen(3C)** to the value, NL_CAT_LOCALE, which is defined in nl_types.h.

Table 5-1 LC_MESSAGES Summary

catopen(3C) oflag value	X/Open Revision	Environment variable affecting message catalog selection
`NL_CAT_LOCALE`	XPG4	`LC_MESSAGES`
0 (zero)	pre-XPG4	`LANG`

Example using XPG4 messaging:

```
if (!setlocale(LC_ALL, ""))
printf("setlocale failed. Continuing in the \
default C locale.\n");
fd = catopen("filename", NL_CAT_LOCALE);
```

XPG4 messaging will attempt to use the C message catalog in `/usr/lib/nls/msg/C` if:

- **setlocale(3C)** fails
- **setlocale(3C)** defaults to the C or POSIX locale
- the C or POSIX locale is selected
- `LC_MESSAGES`, `LC_ALL` and `LANG` environment variables are all unset

If a message catalog by the specified name cannot be found in the `C` directory, the default string in **catgets(3C)** will be used.

Affirmative/Negative Responses (YESEXPR / NOEXPR)

Affirmative and negative response extended regular expression parsing will be supported in the following commands:

- `cp(1)`
- `ex(1)`
- `find(1)`
- `ln(1)`
- `mv(1)`
- `mvdir(1M)`
- `pax(1)`
- `rm(1)`
- `rmdir(1)`
- `tar(1)`

To maintain a similar look and feel, other applications that process affirmative and negative responses should also use these values for `YESEXPR` or `NOEXPR` via **nl_langinfo(3C)**.

The `YESSTR` and `NOSTR` keywords are only provided for backwards compatibility with pre-XPG4 and have been marked as obsolete by X/Open. Some locales no longer contain this data. Applications should use `YESEXPR` or `NOEXPR` instead.

POSIX 1003.1

Note: the System V Interface Definition, issue 3 is the same as POSIX 1003.1.

Summary of Change

1. The following POSIX 1003.1 / SVID3 internationalization commands and libc interfaces are provided:
   - `mkmsgs(1)`
   - `addsev(3C)`
   - `fmtmsg(3C)`
   - `gettxt(3C)`
   - `setcat(3C)`
   - `setlabel(3C)`
   - `pfmt(3C)`
   - `vpfmt(3C)`
2. Collation For Many-to-One Characters

   See “Changes to locales [10.0]” on page 64 for details regarding the many-to-one character collation changes.

Comparison of POSIX and XPG

The following is a comparison between SVID and XPG. See the man pages for more information.

- **mkmsgs(1)** is similar to **gencat(1)**.
- **fmtmsg(3C), gettxt(3C), pfmt(3C)**, and **vpfmt(3C)** are similar to **catgets(3C)** except that **fmtmsg(3C)**, also writes to `stderr` and **pfmt(3C)** and **vpfmt(3C)** write to an open file descriptor.
- **setcat(3C)** specifies the message catalog to use.
- **addsev(3C), setlabel(3C)** - modify the output string.

Performance

Performance is slower than the X/Open NLS equivalents with `catopen(3C)` and `catgets(3C)`. The `gettxt(3C)` and `pfmt(3C)` interfaces open, read and then close the catalog file every time a message is needed from the catalog. The interface `catopen(3C)` is called once, multiple `catgets(3C)` can be called and then `catclose(3C)` when the catalog is no longer needed. The extra open and close operations will degrade the performance of the SVID 3 interfaces.

Alternatives/compatibility

These changes are an alternative to the X/Open NLS specification. They should be used only to port SVR4 applications.

Potential Future Directions

We will be following the SVID through its life cycle. The interface `fmtmsg(3C)` is in its obsolescence phase. We are providing it just so old SVR4 applications can be ported. The `pfmt(3C)` and `vpfmt(3C)` interfaces are the replacement interfaces for `fmtmsg(3C)`. However, the interfaces are still in their introductory phase and may be changed in the next edition of the SVID. We are providing the interfaces, but we are warning our customers that the interface may change to follow the SVID.

6 Commands and Libraries

New Commands [10.0, 10.01]

This section lists commands that are new to 10.0 or 10.01 and are not discussed elsewhere in this document. See the manpages for details.

dmpxlt(1)

Converts the compiled version of the `iconv` tables to ASCII text that can be edited and used as input to `genxlt`. Supports single and multi-byte character codesets.

See also `genxlt(1)`.

genxlt(1)

Generates a compiled version of the `iconv` table suitable for use by `iconv(1)` and `iconv(3C)`. Supports single and multi-byte character codesets.

See also `dmpxlt(1)`.

Changed Commands [10.0]

This section summarizes changes to commands in Section 1 of the *HP-UX Reference* manual (and the online manpages) that are not discussed elsewhere in this document.

adjust(1)

Modified to support Japanese line-breaking rules:

- No line will start with a *bkinsoku* character
- No line will end with a *ekinsoku* character

One or other of these rules will be broken if the line cannot be ended any other way.

The `japanese` locale has two new `char` classes, `bkinsoku` and `ekinsoku`.

gencat(1)

New option:

`-`	specifies `stdin` or `stdout` for the source or catalog files, respectively.

nljust(1)

New syntax:

`nljust [-acilnt] [-d` *`digits`*`] [-e` *`seq`*`] [-j` *`just`*`] [-m` *`mode`*`] [-o` *`order`*`] [-r` *`margin`*`] [-w` *`width`*`] [-x` *`ck`*`] [`*`file...`*`]`

New options:

`-i`	Triggers `ISO 8859/6` interpretation of the data.
`-d` *`digits`*	Where digits is `h`, `w`, or `b`. Processes digits for output as Hindi (`h`), Western (`w`), or both (`b`).

These options support the `ISO 8859/6` character set (also known as `ASMO 708`).

nroff(1)

New option:

`-P`	directs Asian printers to print two-column-wide characters in boxes 1 1/2 columns wide.

sed(1)

For 10.0:

Only printable multi-byte characters printed; non-printable characters displayed in octal.

sort(1)

New option:

`-A`	Sorts on a byte-by-byte basis using each character's encoded value. Extended (signed byte) characters will be considered negative values, and sort before ASCII characters. If you are sorting ASCII characters in a non-`C/POSIX` locale, this flag performs much faster.

Changed options:

Option	Changed Behavior
`-f`	Supports folding of multibyte characters, if folding is possible in the locale.
`-t`	Supports multibyte characters as field delimiters.

Option no longer supported:

`-l` (previously ignored).

tr(1)

- New, XPG4-compliant syntax
- New `-A` option

New syntax:

```
tr [-Acs] string1 string2
tr -s [-Ac] string1
tr -d [-Ac] string1
tr -ds [-Ac] string1 string2
```

New option:

`-A` Translates on a byte-by-byte basis. `tr` with this option does not support extended characters.

Changed options:

- Some changes affecting `-c`, `-d`, and `-s`.

  If `-c` and `-d` are both specified, all characters except those specified by *`string1`* are deleted. The contents of *`string2`* are ignored, unless `-s` is also specified.

  The same string cannot be used for both the `-d` and the `-s` flags.

  If `-d` is not specified, each input character or collating element found in the array specified by *`string1`* is replaced by the character or collating element in the same relative position in the array specified by *`string2`*.

  The abbreviation `[.cc.]` is no longer supported because multi-character collating elements no longer require special treatment.

- Changes affecting the following abbreviations used to introduce ranges of characters or repeated characters into the strings:

  - `[:class:] or [[:class:]]`

    Represents all characters belonging to the defined character class, as defined by the current setting of the `LC_CTYPE` locale category.

  - `[=equiv=]`

    Can be used for *`string1`* or *`string2`* only in combination with the `-d` and `-s` flags.

  - `[a*n]`

    Valid only when it occurs in *`string2`*.

Changed Commands [10.01]

This section summarizes changes to commands in Section 1 of the *HP-UX Reference* manual (and the online manpages) that are not discussed elsewhere in this document.

grep(1), egrep(1), fgrep(1)

For 10.01:

The collating order of characters has been changed in some locales. See the “Locale Changes” section in the *README for NLS in 10.01*.

Changed Commands [10.10]

This section summarizes changes to commands in Section 1 of the *HP-UX Reference* manual (and the online manpages) that are not discussed elsewhere in this document.

cal(1)

- Now handles different international locales.
- The output of `cal(1)` has changed:
  - The column width of a multi-byte character can be at most 4. At least one multi-byte character will be printed.
  - It is only possible to format two months per row in the calendar.
  - The column width of a day has increased from 3 to 5.
- The internationalized environment variables, such as `LANG` are now supported.
- The day and month names have been replaced by corresponding abbreviated day and month versions of `nl_langinfo` items.
- Hard-coded values have been changed to `#defined` values whenever possible.

eucset(1)

- A new option `-c` *`codeset`* has been added to support the HP-15 codesets. Options to `-c` can be one of the following, depending on the codeset:
  - `BIG5`
  - `CCDC`
  - `GB`
  - `SJIS`

  This enables you to set the `cswidth` parameter for HP-15 codesets. You do not have to explicitly specify the `cs` width parameter (it is derived from the locale).

nl(1)

- Now conforms to XPG4 specifications.
- The options can now be intermingled with the optional file operand.
- If - is specified as a file name, input will be taken from `stdin`.
- The limit for input lines is increased to 2048 bytes.
- If text numbering option is used, the text line will now be numbered only if that line contains graphic characters.
- Only one file can now be specified. If multiple files are specified, an error occurs.

sh-posix(1)

- The `POSIX sh` supports the POSIX internationalization module. You have to set the proper `LC_*` or `LANG` environment parameters to input and output the proper local languages.

Refer to the `sh-posix(1)` manpage for details.

Changed Commands [10.20]

This section summarizes changes to commands in Section 1 of the *HP-UX Reference* manual (and the online manpages) that are not discussed elsewhere in this document.

findmsg(1)

`findmsg(1)` supports print specifiers, such as `PRI*` defines in `inttypes.h`, as part of a message within a `catgets()` format. It also supports `#ifdefs` to select the specific messages from the input file.

For example, the message for

```
printf(catgets(catd, NL_SETN, mno, "mesg1 " PRIdMAX" \n"));
```

could be read as

"mesg1 *value_of_PRIdMAX* \n"

Input file(s) will be preprocessed using the C preprocessor (`cpp(1)`) to achieve the above features. The preprocessor recognizes the following new options:

`-D`*sym*.	Defines the symbol *sym* for #ifdef.
`-U`*sym*.	Undefine the symbol *sym*
`-i`	Using this option, you can only extract the required messages under #ifdefs. That is, `-D` and `-U` are also used to select the text in the input file. Without this option, `-D` and `-U` options are only used to select the print specifiers, not the messages from the input file.
`-v`	This option outputs the errors issued by `cpp(1)`. By default, errors issued by `cpp` will not be displayed.

Impact

If more than one string occurs in the message part of the `catgets()` format (`catgets(catd, NL_SETN, msgno, "`*str1*`" "`*str2*`")`), all strings will be concatenated as a single string (`catgets(catd, NL_SETN, msgno, "`*str1str2*`")`.

Performance

Input files that do not contain print specifiers are preprocessed even when it is not required. This is an overhead for such files.

Changed Commands [11.0]

spell(1)

In 11.0:

This cannot be well specified in an internationalized environment. There is no known technology that can be used to recognize general language for user-specified input without providing a complete dictionary along with the input file.

Status

X/Open Withdrawn/To be Withdrawn.

tar(1)

In 11.0:

Cannot be used for portable communication of data with codesets outside the ISO/IEC 646:1991 7-bit characters. `pax` should be used instead. This may be implemented as a wrapper/link to `pax`.

Status

X/Open Withdrawn/To be Withdrawn.

Alternative

pax (1)

Functions and Interfaces [11.0]

Internationalization- and Localization-related Changes

`multibyte.3c` - `mblen()`, `mbtowc()`, `mbstowcs()`, `wctomb()`, `wcstombs()`:

The multibyte routines have been corrected for several Asian locales.

The multibyte routines (`mblen()`, `mbtowc()`, `mbstowcs()`, `wctomb()`, and `wcstombs()`) in past releases have incorrectly identified certain invalid characters as valid for the `ko_KR.eucKR`, `zh_CN.hp15CN`, `zh_TW.big5`, and `zh_TW.ccdc` locales.

The locale definitions and method libraries have been modified so that these locales only recognize 7-bit ASCII single-byte values and the following two-byte values as valid characters:

Table 6-1 Two-byte values recognized by Asian locales

Locale	First byte	Second byte
`ko_KR.eucKR`	0xa1-0xfe	0xa1-0xfe
`zh_CN.hp15CN`[a]	0xa1-0xfe 0xfb 0xfc-0xfe	0xa1-0xfe 0x3f-0x7e 0x21-0x7e
`zh_TW.ccdc`	0xa1-0xfe	0x21-0x7e,0xa1-0xfe
zh_TW.big5	0x81-0xfe	0x40-0x7e,0xa1-0xfe

a. The Simplified Chinese locale is no longer supported by the HP15 method library. A new method library, `libhp15CN.sl`, has been created to support the `zh_CN.hp15CN` locale. To access this library, the -m option of `localedef` must be used with a method file that specifies the `libhp15CN.sl` library. See `/usr/lib/nls/loc/src/zh_CN.hp15CN.m` for an example.

The `multibyte.3c` routines will now exhibit the behavior that is documented in the manpage. As the input methods for these languages do not support the invalid characters, this should not impact customer applications. Customers will now be able to use the multibyte routines to check for character validity.

There should be limited impact on performance in most cases. The simplified Chinese locale (`zh_CN.hp15CN`) user-defined character range ([0xfb,0x3f-0x7e] and [0xfc-0xfe,0x21-0x7e]) requires additional checks that will cause the routines to be slower for UDCs than the other character ranges.

No applications conforming to the documented behavior will be affected. HP has made every attempt to verify that no applications depend on the prior behavior.

The ability exists to create and use user-defined locales and method libraries instead of the system-provided locales.

Obsolete Commands And Interfaces

Obsolete Core Commands and libc [10.0]

The following commands and interfaces are now obsolete and have been removed from libc:

Table 6-2 Obsolete Commands And Interfaces

Obsolete Item	Recommended Replacement	Comments
buildlang(1M)	localedef(1M)	similar, but not equal replacement
not needed	byte_status(3C)	obsolete; removed from libc
c_colwidth(3C)	not needed	obsolete; removed from libc
catgetmsg(3C)	catgets(3C)	similar, but not equal replacement
catread(3C)	catgets(3C)	similar, but not equal replacement
currlangid(3C)	not needed	
firstof2(3C)	not needed	obsolete; removed from libc
fprintmsg(3C)	fprintf(3S)	
getmsg(3C)	catgets(3C)	similar, but not equal replacement
ICONV(3C)	iconv(3C)	similar, but not equal replacement
iconvclose(3C)	iconv_close(3C)	
iconvlock(3C)	not needed	
iconvopen(3C)	iconv_open(3C)	similar, but not equal replacement
iconvsize(3C)	not needed	
idtolang(3C)	not needed	
langinfo(3C)	nl_langinfo(3C)	
langinit(3C)	setlocale(3C)	

Table 6-2 Obsolete Commands And Interfaces

Obsolete Item	Recommended Replacement	Comments
langtoid(3C)	not needed	
nl_asctime(3C)	strftime(3C)	
nl_ascxtime(3C)	strftime(3C)	
nl_atof(3C)	atof(3C)	
nl_catopen(3C)	catopen(3C)	
nl_ctime(3C)	strftime(3C)	
nl_cxtime(3C)	strftime(3C)	
nl_fprintf(3C)	fprintf(3C)	
nl_fscanf(3C)	fscanf(3C)	
nl_gcvt(3C)	gcvt(3C)	
nl_init(3C)	setlocale(3C)	
nl_isalnum(3C)	isalnum(3C)	
nl_isalpha(3C)	isalpha(3C)	
nl_iscntrl(3C)	iscntrl(3C)	
nl_isdigit(3C)	isdigit(3C)	
nl_isgraph(3C)	isgraph(3C)	
nl_islower(3C)	islower(3C)	
nl_isprint(3C)	isprint(3C)	
nl_ispunct(3C)	ispunct(3C)	
nl_isspace(3C)	isspace(3C)	
nl_isupper(3C)	isupper(3C)	
nl_isxdigit(3C)	isxdigit(3C)	

Table 6-2 Obsolete Commands And Interfaces

Obsolete Item	Recommended Replacement	Comments
nl_msg(3C)	catgets(3C)	similar, but not equal replacement
nl_printf(3C)	printf(3C)	
nl_scanf(3C)	scanf(3C)	
nl_sscanf(3C)	sscanf(3C)	
nl_sprintf(3C)	sprintf(3C)	
nl_strcmp(3C)	strcoll(3C)	similar, but not equal replacement
nl_strncmp(3C)	strcoll(3C)	similar, but not equal replacement
nl_strtod(3C)	strtod(3C)	
nl_tolower(3C)	tolower(3C)	
nl_toupper(3C)	toupper(3C)	
nlsinfo(1)	locale(1)	similar, but not equal replacement
printmsg(3C)	printf(3S)	
not needed	secof2(3C)	obsolete; removed from libc
sprintmsg(3C)	sprintf(3S)	
strcmp8(3C)	strcoll(3C)	
strcmp16(3C)	strcoll(3C)	
strncmp8(3C)	strcoll(3C)	similar, but not equal replacement
strncmp16(3C)	strcoll(3C)	similar, but not equal replacement

Compatibility

Applications that call these obsolete interfaces and were compiled archive pre-10.0 will operate as expected. However, if the application was compiled shared pre-10.0, it will fail at run time due to an unresolved symbol. If the application is recompiled in 10.0, the application will not compile due to an unresolved symbol.

Obsolete HP-UX Proprietary Multi-byte nl_tools_16(3C) Interfaces [10.0]

The following **nl_tools_16(3C)** interfaces are now obsolete, but these interfaces must still be provided in order to give customers time to convert to XPG4 wide character interfaces. The following will apply to these interfaces:

- no longer supported
- no enhancements made
- interfaces moved to `/usr/old/usr/include` to discourage use and for backwards compatibility.

Table 6-3 Obsolete nl_tools_16(3C) Interfaces

Obsolete Interface	Recommended Replacement	New Location
ADVANCE(3C)	none	`/usr/old/usr/include/nl_ctype.h`
BYTE_STATUS(3C)	none	`/usr/old/usr/include/nl_ctype.h`
C_COLWIDTH(3C)	wcwidth(3C)	`/usr/old/usr/include/nl_ctype.h`
CHARADV(3C)	mbtowc(3C)	`/usr/old/usr/include/nl_ctype.h`
_CHARADV(3C)	mbtowc(3C)	`/usr/old/usr/include/nl_ctype.h`
CHARAT(3C)	mbtowc(3C)	`/usr/old/usr/include/nl_ctype.h`
FIRSTof2(3C)	none	`/usr/old/usr/include/nl_ctype.h`
PCHAR(3C)	wctomb(3C)	`/usr/old/usr/include/nl_ctype.h`
PCHARADV(3C)	wctomb(3C)	`/usr/old/usr/include/nl_ctype.h`
SECof2(3C)	none	`/usr/old/usr/include/nl_ctype.h`
WCHAR(3C)	mbtowc(3C)	`/usr/old/usr/include/nl_ctype.h`
_WCHAR(3C)	mbtowc(3C)	`/usr/old/usr/include/nl_ctype.h`
WCHARADV(3C)	mbtowc(3C)	`/usr/old/usr/include/nl_ctype.h`
_WCHARADV(3C)	mbtowc(3C)	`/usr/old/usr/include/nl_ctype.h`
WC_COLWIDTH(3C)	wcwidth(3C)	`/usr/old/usr/include/nl_ctype.h`

Compatibility

Applications compiled shared or archive on pre-HP-UX 10.0 will operate as expected. However, if an application is re-compiled on HP-UX 10.0, the application will need to change the #include statement to find nl_ctype.h in the new location:

```
#include "/usr/old/usr/include/nl_ctype.h"
```

Alternatives

The XPG4 wide character interfaces (i.e. `mbtowc(3C)`, `wctomb(3C)`, ...) should be used instead.

WARNING **Do not intermix nl_tools_16(3C) macros and interfaces with wide character interfaces; unspecified, undesirable results may occur.**

Obsolete HP-UX Proprietary Multi-byte nl_tools_16(3C) Interfaces [10.30]

The APIs in `nl_ctype` listed in the previous section are no longer provided as of HP-UX 10.30. But, they will remain the versioned `libc.1` for binary compatibility.

7 Graphical User Interfaces

HP Common Desktop Environment (CDE) & Motif

HP CDE 1.0 [10.10]

Printing and Building Help Volumes in Multibyte Locales

The user's LANG environment variable must match the locale specified for a given help volume to print or build the volume correctly.

The locale associated with a help volume is set using the `LanguageElementDefaultLocale` entity declaration, or using a combination of the `LanguageElementDefaultLocale` and `LanguageElementDefaultCharset` entity declarations.

If the locale is not specified for a help volume, the value of the `LANG` environment variable is used.

Accessing Font Aliases for Remote CDE Sessions

Your CDE session must have access to CDE font aliases to run properly. Typically, these aliases are installed with the fonts provided with your X server.

However, when running CDE as a remote session using the command

`X -query...  fp+ tcp/`*`hostname`*`:7000`

or running CDE from an X terminal, it is possible that the font aliases may not be available to the X server.

This can happen when the fonts on the server are from a prior release of HP-UX or from another vendor. In this situation, you may see fonts that do not work correctly with the local language you have selected. Fortunately, CDE provides “backup” CDE font aliases that your X server can use. These are located in the directory

`/usr/dt/config/xfonts/`*`locale_name`*

This location must also contain a `fonts.dir` file. This file is required by the X server to use the font aliases mechanism.

If you are running CDE remotely, your system administrator can cause the CDE font aliases to be used by setting up a font server to run on the system hosting CDE, making that font server use the CDE font aliases for the language you choose.

For example, if you want to work in Japanese EUC (locale name `ja_JP.eucJP`), a font server should be started on the CDE system using this command:

```
/usr/bin/X11/fs -port 7000 -daemon -quiet_if_addrinuse
```

In the font server's configuration file (the default configuration file is `/etc/X11/fs/config`), make sure the CDE font alias directory is contained in the catalogue definition. For example, to support a user running CDE in Japanese EUC, the font server's configuration file might contain the following:

```
catalogue = ...,/usr/dt/config/xfonts/ja_JP.eucJP/
```

Alternatively, your system administrator can copy the `fonts.alias` files from the CDE host system to your local system and incorporate them into your X server's `fonts.alias` file.

Changing Languages Between Sessions in CDE

When you log into the CDE desktop, you can select the language (or locale) that you want to use for your session. A list of languages is displayed when you click the Option button in the login screen.

If you change the language you are using between CDE sessions, previously saved local language customizations may conflict with your new language session.

For example, suppose you log in with language `ja_JP.eucJP` and create Japanese workspace names. You then save the session when you exit the desktop. Suppose the next time you log in, instead of choosing `ja_JP.eucJP`, you select German, `de_DE.iso88591`. The desktop will attempt to display the Japanese workspace names using fonts appropriate for the German language. The workspace names will be unreadable.

This conflict results only if you change between two languages that use different coded character sets. You can change from either C or English locale without difficulty. Likewise, you can switch between similar locales, such as French and German, which are both Western European locales.

Any personal customizations that contain local language characters can cause similar symptoms as that described above. Personal session customizations are saved in your `$HOME/.dt` directory.

If you want to select a language that is different from your previously saved session, you should first start CDE in a fail safe session. From the terminal window provided, remove the files that contain local language text.

Files that might contain local language characters include:

- `$HOME/.dt/session/current/dt.resources`
- `$HOME/.dt/help`
- `$HOME/.dt/types/*`

If you are not sure what files contain local language characters, remove the directory `$HOME/.dt/session` using this command:

```
rm -r $HOME/.dt/session
```

This causes a default session to be invoked when you log in with the new language.

In addition, you should remove any help files accessed with the prior local language by executing:

```
rm -r $HOME/.dt/help
```

Labels and comments in action definitions can also include localized data. If so, the action labels need to be modified to be readable in the new language selected for your CDE session. User-defined actions are located in `$HOME/.dt/types/*`.

NOTE

If you delete your personal customization directory, `$HOME/.dt/*`, you can avoid the problems that occur when changing languages. However, remember that you will lose any actions or other customizations you might have created. Removing the entire directory is therefore not recommended.

HP CDE 2.1 & Motif [11.0]

HP CDE 2.1

HP CDE 2.1 is the latest version on HP-UX 11.0, superseding HP CDE 1.0 as shipped with previous releases.

There are a few small functional differences between the run-time functionality of HP CDE 1.0 and 2.1, including:

- For selected locales, input methods can be selected at session start (using `dtimsstart`).

HP CDE 1.0 was localized in 10 languages. HP CDE 2.1 will be fully localized (including interfaces, messages, on-line help, and manuals) for English, Japanese, German, and French. It will have localized interfaces and messages for Chinese (traditional and simplified), Korean, Swedish, Italian and Spanish.

User Documentation The *HP CDE 2.1 Getting Started Guide* is provided in hardcopy for all languages mentioned above.

X/Motif Libraries

For 11.0, the changes fall into two types:

- To the X Window System Version 11 from Release 6.1 libraries to Release 6.2 libraries
- To OSF Motif from 1.2 libraries to 2.1 libraries

Changes to X Window System Changes from X11 R6.1 to 6.2

- Print Extension
- Vertical Writing & User-Defined characters

Changes to Motif Changes from Motif 1.2 to 2.1

- Internationalization support: on-the-spot input method and vertical text writing

HP VUE 3.0

Vuelogin 'langSetup' Resource [10.10]

A new resource, langSetup, has been added to vuelogin. This resource points to a script that will be executed whenever the user selects a different locale from the login screen's language options. The default setting for langSetup is the script `/etc/vue/config/Xlangsetup`.

This script checks the newly selected language and, if required, will install the appropriate keymap to the X server.

For more information, refer to the `vuelogin(1)` manpage.

Support for New Locales in VUE [10.10]

Support was added for the following locales as part of the fileset `VUE-RUN`:

- `ar_SA.iso88596` - Arabic
- `bg_BG.iso88595` - Bulgarian
- `cs_CZ.iso88592` - Czech
- `el_GR.iso88597` - Greek
- `hr_HR.iso88592` - Croatian
- `hu_HU.iso88592` - Hungarian
- `iw_IL.iso88598` - Hebrew
- `pl_PL.iso88592` - Polish
- `ro_RO.iso88592` - Rumanian
- `ru_RU.iso88595` - Russian
- `sk_SK.iso88592` - Slovakian
- `sl_SI.iso88592` - Slovenian
- `tr_TR.iso88599` - Turkish

Image Help When Using Localized Environments with HP VUE [10.20]

If you install Imaging help for a localized environment, the help may not appear in the main help window. To fix this problem, log into a localized VUE session as `root` and run `helpgen` in a terminal window.

Text Help When Using Roman8 Locales with HP VUE [10.20]

Translated text in the main help window for Roman8 locales (locale names ending with `.roman8`) that contain accents are not displayed correctly. To fix this problem, change the links to the help browser volume for the help browser. As `root`, execute the following:

```
cd /etc/vhelp/volumes/locale
rm browser.hv
ln -s ../../../../etc/vhelp/help/newlocale/Browser/browser.hv .
```

where `[`*`locale`*`]` is the Roman8 locale name, and *`newlocale`* is the locale name with `.iso88591` in place of `.roman8`.

X11R4/Motif 1.0/1.1 Application Font Usage [10.20]

HP VUE is now built on X11R5 and Motif 1.2. Motif 1.2 extended the concept of font lists to allow a font list to contain either a font struct, a font set (a new X11R5 structure used for internationalized applications), or any mixture of font structs and font sets.

Use of a font set by an application makes sure that the font selected for that application matches the encoding of the data generated by that application. For example, if the application is run in a locale that uses the ISO 8859.1 character set, it is important that an ISO 8859.1 font be used to render that data. Further, some languages require more than one font to correctly render all characters. The concept of font sets is used to open multiple fonts for these languages, treating them as a single entity.

Since HP VUE is made up of internationalized clients, vuestyle has been changed to generate font list resources that utilize the internationalization capabilities provided by Motif 1.2 and X11R5. This means that Motif 1.2 clients will get internationalized font resources generated by vuestyle.

Applications built archived with Motif 1.1 or Motif 1.0 will experience problems with the new font list resources generated by vuestyle. These applications may obtain the default fixed font, or in extreme cases, may fail completely. Several simple workarounds exist for these clients:

- Font settings in the resource environment can be targeted specifically to those particular applications that have problems with the vuestyle exported font set settings. This can be done by creating or appending to an application's app-defaults file with the old-style font list resource specifications.

  This allows new Motif 1.2/X11R5 clients to still obtain the benefits of internationalized font resources under HP VUE while allowing the older clients to operate as they have in the past.

  Changes to font resources made by vuestyle will not affect clients that use this workaround.

- Modify vuestyle's app-defaults file. In this file, the old 1.1 vuestyle font list resources are provided, but are commented out. Simply uncomment the old resources, and comment out the new resources. Then, bring up vuestyle and select a font; save the session as your home session or restore it as your current session to cause the change to be reflected for future sessions.

  Using this workaround has the benefit that your old clients will still respond to font changes made with vuestyle. The disadvantage is that new X11R5/Motif 1.2 clients will not obtain the internationalization functionality from the fonts that they expect. Those clients will still work, but they may obtain the wrong font (that is, use a Roman8 font when an ISO 8859.1 font is required).

X Window

Functions Marked for Obsolescence [10.0]

XHP functions for internationalized keyboard input will be removed at the next release of HP-UX. These HP proprietary functions are no longer needed since X11R5 provides public, standard APIs that provide equivalent functionality. Specifically, the following functions will be removed with the next release of HP-UX:

- XHPConvertLookup()
- XHPGetEurasianCvt()
- XHPInputChinese_s()
- XHPInputChinese_t()
- XHPInputJapanese()
- XHPInputKorean()
- XHPInputRoman8()
- XHPInputISO7sub()
- XHPNlioctl()

Instead, applications should use the following X11R5 routines:

- XOpenIM()
- XCloseIM()
- XGetIMValues()
- XCreateIC()
- XDestroyIC()
- XSetICFocus()
- XUnsetICFocus()
- XmbResetIC()
- XIMOfIC()
- XSetICValues()
- XGetICValues()
- XmbLookupString()
- XwcLookupString()

The new X11R5 functionality gives applications greater control and more flexibility than the HP proprietary functions.

Changes in Keyboard Functionality [10.0]

The `XHPSetKeyboardMapping` function and the related routines `XHPRefreshKeyboardMapping()` and `XHPSetKbdMapInit()` will be removed at the next release of HP-UX. These functions are used to allow an application to change their local copy of the keymap to emulate a particular national language keyboard. The functions only work for HP HIL keyboards. Future HP workstations may no longer support HP HIL keyboards, so these functions have a diminishing value. An application or user cannot depend on the functions working in all cases.

In HP-UX 9.x, many applications (such as `vuepad` and `hpterm`) call `XHPSetKeyboardMapping`, passing through the value of *`KBD_LANG`* environment variable to indicate the keyboard language to be used by the application. As of 10.0, HP VUE no longer set the *`KBD_LANG`* environment variable. Clients and libraries have removed the call to `XHPSetKeyboardMapping()` since its behavior is not reliable in a mixed DIN and HIL keyboard environment.

Rendering International Characters [10.0]

In X11R3 and X11R4, support for rendering Asian characters was done through a proprietary mechanism called the associate font mechanism. Through HP-UX 9.0, rendering of Asian character data requires that two fonts be opened to correctly render all Asian characters. When an Asian font is opened with `XLoadFont()` or `XLoadQueryFont()`, the font is checked for the existence of an associate font property. If this property exists, an additional font is opened and is transparent to the application. All calls to calculate metrics or to perform rendering with the font causes the implementation to also use the associate font when present.

With support for 3-byte EUC and 4-byte EUC with HP-UX 10.0, rendering Asian character data often requires that more than two fonts be opened and used. The old associate font mechanism could not be reliably expanded to handle these cases. In addition, X11R5 and Motif 1.2 provide alternate, standard mechanisms for rendering internationalized text data. Because of this, support for the associate font mechanism will be removed in a future release of HP-UX.

Applications directly calling the X library to render international text should replace their calculation of font metrics and their calls to `XDraw*`, `XLoadFont()`, and `XLoadQueryFont()` with the following:

```
XCreateFontSet()

XExtentsOfFontSet()

XmbDrawString()              XwcDrawString()

XmbDrawImageString()         XwcDrawImageString()

XmbDrawText()                XwcDrawText()

XmbTextEscapement()          XwcTextEscapement()

XmbTextExtents()             XwcTextExtents()

XmbTextPerCharExtents()      XwcTextPerCharExtents()
```

Applications and users who set Motif FontList resources and desire rendering of internationalized text data should modify those resources to specify font sets. As of Motif 1.2, a FontList resource can contain any combination of font structs and font sets. A font set is specified in a FontList resource by adding a ":" to the end of the FontList entry. For a font set, a FontList entry can be one or more ";" separated base names. A base name is either a font name (such as jpn.8x18) or an X Logical Font Description (XLFD). A typical FontList setting for Japanese might be `app_name*Text.fontList: *-mincho-*-18-*:`

The ":" character at the end of the FontList resource specification is mandatory to use font set technology.

To cause Motif to use font sets correct in the application, either `setlocale()` or `XtSetLanguageProc()` must be called prior to the toolkit being initialized. As of 10.0, HP VUE was converted to use X11R5, Motif 1.2, and the new font set technology. So font resources for all HP VUE clients on 10.x should be set to use font sets if internationalized functionality is desired.

Terminal Emulators [10.0]

`hpterm` has changed in several ways for this release:

- The ability to dynamically change `hpterm`'s keyboard language has been removed. This functionality required proprietary technology and has become impossible to implement across all HP platforms. See "Changes in Keyboard Functionality" earlier in this section for more information. `hpterm` now depends on the user to properly initialize the *LANG* environment variable to operate correctly in the desired locale.
- `hpterm` does not support 3- and 4-byte locales; 3- and 4-byte characters are silently discarded from the data stream. If you need a terminal emulator that fully supports locales, you should use `dtterm`.

X11 New Bitmap Fonts [10.10]

New fonts were added for the following codesets:

- iso8859-2
- iso8859-5
- iso8859-7
- iso8859-8
- iso8859-9

Each of the codesets consists of eight user (fixed space) fonts and seven system (proportional) fonts.

X11 FontServer [10.10]

The HP-UX fontserver is based on X11 R6. New fontserver capabilities include:

- R6 mechanisms for font manipulation
- R6 character encoding set
- Glyph caching and scalable aliases
- Universal Font Scaling Technology (UFST) rasterizer
- Truetype fonts

This section provides a short description of the new R6 fontserver capabilities. For complete information, refer to the *Programmer's Supplement for R6* (ISBN 1-56592-089-9) by O'Reilly & Associates, Inc.

Font Manipulation

New R6 mechanisms are used to specify anamorphic scaling, obliquing, mirroring, and rotation of fonts.

The new mechanism uses a set of four numbers delineated by brackets to replace the `pointsize` or `pixelsize` fields. It is used to create a two-dimensional matrix that controls the transformation for each character.

Charset Encoding Extension

HP-UX 10.10 supports the X11 R6 charset encoding extension. The charset encoding extension allows subsetting of fonts so that not all characters need to be generated for a font.

Glyph Caching and Scalable Aliases

Glyph caching and scalable aliases are standard capabilities in the X11 R6 fontserver. Glyph caching is the deferred loading of character glyphs and allows X to reduce memory and computation requirements associated with the generation of fonts.

Scalable aliases increase the capability of font name aliases to allow them to be used with scalable font names and with the matrix XLFD enhancement.

New Rasterizer

The new Universal Font Scaling Technology (UFST) rasterizer from AGFA replaces the type1 and intellifont rasterizers. The UFST rasterizer supports truetype, type1 and intellifont font files.

New Fonts

The 10.10 release includes a set of 35 truetype fonts. These include treatments from the following families of AGFA fonts:

- Albertus
- AntiqueOlive
- CGomega

- CGtimes
- ClarendonCondensed
- Coronet
- Courier
- Garamond
- LetterGothic
- Marigold
- Univers
- UniversCondensed

The Intellifont fonts are provided in an updated format that can be handled with the UFST rasterizer. Some Intellifont fonts may not be provided in future releases. However, the same font name would be supported using a `fonts.alias` file.

Command Changes

In a future release, the `-tfm` option will be removed from the `stmkdirs` command. These metrics are no longer used and are not generated for truetype fonts.

X11 FontServer [10.20]

TrueType Fonts

The same set of fonts available in Intellifont format are now available in TrueType format. Hewlett-Packard will discontinue support for AGFA's Intellifont font technology in a future release.

The Intellifont fonts are provided in an updated format that can be handled with the UFST rasterizer. Some Intellifont fonts might not be provided in future releases. However, the same font name would be supported using a `fonts.alias` file.

Administering Charsets for True Type Fonts

Character set definitions for True Type fonts are stored in the directory `/usr/lib/X11/fonts/stadmin/ttf/charsets`. Files in this directory named *`characterset`*`.sym` define the character mapping for a particular character set.

Wildcard Aliases

HP provides an extension to support wildcard aliases. If a `fonts.alias` file contains the line

```
WILDCARD_ALIASES_SUPPORTED
```

an alias name can contain wildcard entries. For example, the entry

```
-agfa-univers-medium-r-normal--10-*-*-*-p-*-hp-roman8 \
-adobe-utopia-medium-r-normal--10-100-75-75-p-60-hp-roman8
```

will cause all of the following font descriptions to reference the adobe-utopia font instead of the agfa-univers font:

```
agfa-univers-medium-r-normal--10-*-*-*-p-*-hp-roman8
agfa-univers-medium-r-normal--10-72-100-100-p-0-hp-roman8
agfa-univers-medium-r-normal--*-72-100-100-p-*-hp-roman8
```

Compatibility [10.30]

Under the following conditions, X11 Font matching may cause excessive delays in application execution:

- The X Server being used is not the Hewlett-Packard X Server.
- The client's default font is not available on the X Server.

These delays can be eliminated by changing the client's default font to a font available on the X Server; that is, explicitly setting the environment variable LANG to be `C.iso88591`.

X Window System (X11 R6) Run-Time Libraries on Workstations [11.0 patch, 11i]

This release provides workstation support for:

- Japanese 109-key keyboards.
- 64-bit X Window System shared library (stack).

The following X and Motif libraries are available in 64-bits:

- libMrm.a
- libXm.4
- libICE.2

- libSM.2
- libX11.3
- libXIE.2
- libXext.3
- libXhp11.3
- libXi.3
- libXp.2
- libXmu *(new with HP-UX 11.0 ACE 9911)*
- libXaw *(new with HP-UX 11.0 ACE 9911*)

To date, these libraries are only found in release 6 of the X libs (X11 R6) and Motif version 2.1. No 64-bit versions of the tooltalk libraries, libtt or libDtSvc are available.

The 64-bit X Window System (X11 R6) run-time libraries are usable only on systems that support the 64-bit operating system. To use the 64-bit run-time libraries, you must specify that the application will run (compile) in 64-bit mode. The 64-bit libraries are then used automatically.

8 Miscellaneous Modifications

Configuration Files [10.0]

The NLS configuration file, `/usr/lib/nls/config` will be modified to include the actual file name of the locale and a comment with descriptive information about the locale. The fields in the template are each separated by <space> character(s) and a comment is preceded by a #. Only characters from the POSIX portable character set are supported within this file.

Table 8-1 Changes to the NLS configuration file

lang id	*locale name*	*file name*	*# comment*
8	`german`	`de_DE.roman8`	# German, Germany, ROMAN8
12	`spanish`	`es_ES.roman8`	# Spanish, Spain, ROMAN8
99	`C`	`C`	# Computer default, same as # POSIX
100	`POSIX`	`POSIX`	# specified by POSIX - # contains only the POSIX # portable characters
101	`american.iso88591`	`en_US.iso88591`	# English, United States, # ISO 8859/1
102	`c-french.iso88591`	`fr_CA.iso88591`	# French, Canada. # ISO 8859/1

NLS libc Interfaces [10.0]

WARNING **Applications compiled archive in HP-UX 9.x will continue to work as expected; however, shared applications may experience new, different behavior.**

catopen(3C)

Standards Conformance: POSIX, X/Open

1. The second parameter to **catopen(3C)** was previously always set to 0. For XPG4 compliance, the modified **catopen(3C)** allows oflag to be set to a value other than 0.
2. The default NLSPATH has changed.

   HP-UX 9.x default NLSPATH:

   `/usr/lib/nls/%l/%t/%c/%N.cat`

   HP-UX 10.0 default NLSPATH:

   `/usr/lib/nls/msg/%L/%N.cat:/usr/lib/nls/%l/%t/%c/%N.cat`

   This change is needed in order to parallel the change to the locales, which are all stored at the same level in the directory. The HP-UX 9.x path is appended in order to permit backwards compatibility.

getlocale(3C)

Standards Conformance: HP-specific proprietary

NOTE will be obsoleted; use the **setlocale(3C)** command

The only type of data that can be retrieved with **getlocale(3C)** is `LOCALE_STATUS`. The other types of data: `ERROR_STATUS` and `MODIFIER_STATUS` are no longer supported. The field, `LC_ALL_D`, in the data structure `locale_data`, has no meaning in the current NLS implementation and is no longer valid.

Impact

An application that is compiled on HP-UX 10.0 and accesses the `locale_data` field, `LC_ALL_D`, or the constants `ERROR_STATUS`, `MODIFIER_STATUS`, will require code changes to remove usage of these obsolete references.

setlocale(3C)

Standards Conformance: X/Open, POSIX, ISO-C

The format of the return string of **setlocale(3C)** has changed. Applications should not parse this string especially if portability across vendor's platforms is important, since the format of this string is not standardized and may change from release to release. The return string of **setlocale(3C)** should only be used to restore the locale settings with **setlocale(3C)**. Other usage is not guaranteed and may have undesired consequences.

Alternatives

The `getlocale(3C)` utility may be used to retrieve the locale settings, but this interface is not standard and may be modified or obsoleted in a future release.

nl_langinfo(3C)

Standards Conformance: X/Open

The modified langinfo.h constants include:

Table 8-2 Modified langinfo.h Constants

Modified Constant	Description
`CRNCYSTR`	The algorithm for determining the value of `CRNCYSTR` has been modified to be consistent with XPG4 `localeconv(3C)` fields: `p_cs_precedes`, `n_cs_precedes`, `mon_decimal_point`. The value returned with `nl_langinfo(3C)` may vary from the results in previous releases.

Table 8-2 Modified langinfo.h Constants

Modified Constant	Description
NOEXPR and YESEXPR	The value returned for this constant is now a true extended regular expression; whereas, in previous releases the value was a string. Parsing and interpreting this constant should be accomplished via regular expression interfaces. See `regcomp(3C)` for more details.

The new langinfo.h constants include:

Table 8-3 New langinfo.h Constants

New Constant	Description
ALT_DIGITS	alternative symbols for digits
CODESET	codeset of the locale
ERA	era description segments, which describe how years are counted and displayed for each era
ERA_D_FMT	era date format
ERA_D_T_FMT	alternative date and time format

The obsolete langinfo.h constants include:

Table 8-4 Obsolete langinfo.h Constants

Obsolete Constant	Recommended Replacement
BYTES_CHAR	MB_CUR_MAX in `stdlib.h`
CODE_SCHEME	no replacement
COMM_CHAR	no replacement
CONTEXT	no replacement
CSWIDTH	`mblen(3C)` or `wcwidth(3C)`
ERA_FMT	ERA constant in `langinfo.h`
LANGID	no replacement

Table 8-4 Obsolete langinfo.h Constants

Obsolete Constant	Recommended Replacement
LANGNAME	no replacement
REVISION	no replacement

Multibyte Support Extensions

There are a number of differences between the MSE and HP-UX's 10.0 XPG4 interfaces:

Table 8-5 Differences With MSE / HP-UX 10.0 XPG4 Interfaces

HP-UX 10.0 XPG4	MSE
(f)printf() and (f)scanf() use %C, %S to specify wide char and wide string, respectively.	(f)printf() and (f)scanf() use %lc, %ls to specify wide char and wide string, respectively.
(f)putwc() and putwchar(): first argument is wint_t.	(f)putwc() and putwchar(): first argument is wchar_t.
wcschr() and wcsrchr(): second argument is wint_t.	wcschr() and wcsrchr(): second argument is wchar_t.
wcstok(): only 2 arguments are specified.	wcstok(): third argument is wchar_t **.
wcsftime(): third argument is char *.	wcsftime(): third argument is wchar_t

Shells [10.0]

Differences between the Bourne and POSIX Shells

Although the POSIX shell is a superset of the Bourne shell and contains all the Bourne shell's syntactic constructs, Bourne shell users will notice a few changes, including the following.

Changing the value of `LC_CTYPE` or `LANG` after the shell has started will not affect the lexical processing of shell commands in the current shell execution environment or its subshells.

Recompilation Note [10.0]

NOTE In order to access any of the features and changes available in HP-UX 10.0, an application *must* be recompiled with HP-UX 10.0.

SD-UX partial Internationalization and Localization [10.20]

You will see messages on the screen and in the SD log file displayed in the local language. This feature requires that the translated SD message catalog is available on the system and that the `LANG` environment variable in `etc/rc.config.d` has been changed to designate that local language. For example, to make SD agent and daemon log files display in Japanese, `etc/rc.config.d/LANG` must be `LANG=ja_JP.SJIS` or `LANG=ja_JP.eucJP`.

Archive Internationalized Applications [10.30]

WARNING

It is not recommended to build internationalized applications using an archive `libc`. But if you must, there are build changes for *`all`* archive programs that use NLS.

For any application that calls `setlocale(3C)` or `iconv(3C)` and is compiled, the archive will not be a complete archive and will now contain position-independent code and data, which are loaded at runtime. Two calls in `libdld.sl` are used to load the position-independent code and data must be loaded as shared. However, the rest of the libraries can be archived in the executable.

The compile and load options for applications built with `-Wl,-a,archive` are as follows. Be aware that the `-Wl,-a,archive link` option is a positionally-dependent option and should be at the beginning of the `cc` line to compile the program.

Example of `sh` using CCOPTS and LDOPTS

9.x compile:

```
export CCOPTS="${CCOPTS:-} your_options"
export LDOPTS="${LDOPTS:-} your_options"
cc -Wl,-a,archive -o executable source_file
```

10.x compile:

```
export CCOPTS="${CCOPTS:-} | your_options -Wl,-E -l:libdld.sl"
export LDOPTS="${LDOPTS:-} your_options -E -l:libdld.sl"
cc -Wl,-a,archive -o executable source_file
```

Example Using `cc` Line Only

9.x compile:

```
cc -Wl,-a,archive -o executable source_file
```

10.x compile:

```
cc -Wl,-a,archive -Wl,-E -l:libdld.sl -o executable source_file
```

elm 2.4 [10.30]

For 10.30, the new version of `elm` is elm 2.4. elm 2.4 on HP-UX continues to support all the features of `elm` prior to 10.30. In addition, there is added functionality, briefly described in this section.

MIME support (RFC 1521)

RFC 1521 describes a mechanism for denoting textual body parts that are coded in various character sets, as well as methods for encoding such body parts as sequences of printable ASCII characters.

`elm 2.4` allows you to send and view MIME-encoded messages. This enables you to send messages other than 7-bit ASCII texts. `elm 2.4` supports a variety of non-text messages like audio, video, image, PostScript, and so on.

If a mailer does not support a particular Content-TYPE or CHARACTER-SET, `elm 2.4` calls the `metamail` program which selects the appropriate interpreter to display the contents of the message. The `metamail` program is extensive in terms of the variety of CONTENT-TYPES and SUBTYPES it can display via the MAILCAP file mechanism.

elm 2.4 also supports MIME encoding for the mail headers (RFC 1522). This allows character sets other than ASCII in RFC 822 message headers.

JIS Support for Japanese email Messages

This facility supports the industry standard JIS encoding scheme for sending and receiving Japanese mail. Regardless of the internal code of the sender (Japanese EUC or Shift JIS), `elm 2.4` correctly displays the mail using the locale of the receiver.

Backward compatibility is provided for interacting with the non-JIS mailers. You have the option of suppressing the JIS encoding for outgoing mail, and, if desired, using the `elmrc` variable `jisconversion`. Using the `elmrc` variable `savecharset`, you can specify the codeset to be used (EUC, SJIS, or JIS) to save mail into a folder.

JIS support is also provided for the stand-alone utilities, such as `answer`, `readmail`, `newmail`, `mailfrom`, and `fastmail`.

GeoCustoms [10.30]

HP-UX will support ignition of multiple languages on a new system. `/usr/sbin/geocustoms` provides a multi-lingual user interface to enable any European language user to manage the language configuration of a 10.30 system. `set_parms` calls GeoCustoms when the default system language has not been predetermined. If you order the European option, HP-UX must find out from you whether to use English, French, German, Italian, Spanish or Swedish. GeoCustoms determines the language through a localized interface that takes the international customer quickly into the remaining system configuration screens. Networking configuration questions are presented by `set_parms` (when applicable) in the chosen language for navigating GeoCustoms.

Therefore, `set_parms` and `login` will come up in the correct language.

Compatibility

`auto_parms` and `set_parms` are delivered with GeoCustoms. Syntax rules for edits of `/etc/rc.config.d/LANG` and `/etc/dt/config/Xconfig` have been followed.

Performance

GeoCustoms only appears automatically at first boot on systems where the language choice is ambiguous; therefore, there is no performance loss for single language users.

GeoCustoms quickly walks you through a process that otherwise might be overwhelming. This is a performance improvement.

If you select “English (Universal)”, also known as “C”, the performance of some applications may be slower than if the language “English (`SET_NULL_LOCALE`)” was used.

Alternatives

To prevent GeoCustoms from running make sure that a valid locale is assigned to `_hp_locale` in `/tmp/install.vars` (if the file exists).

Obsolescence

Instant Ignition no longer includes the 9.x tools known as `datebook`, `xcal`, `xdialog`, and `xhpcalc`.

New supported USB keyboards [10.20 patch, 11.0 patch, 11i]

New supported USB keyboards are:

- A4983-60401 English, U.S.
- A4983-60403 German, Germany
- A4983-60404 Spanish, European
- A4983-60405 French, France
- A4983-60406 Japanese, Kanji
- A4983-60409 Norwegian
- A4983-60411 Swiss-German
- A4983-60412 Swedish
- A4983-60413 English, U.K.
- A4983-60414 Belgian/Flemish
- A4983-60416 Danish
- A4983-60417 Italian
- A4983-60421 Korean
- A4983-60423 Chinese, Traditional

Multibyte Support Extension and Unix98 Support [11i]

A new set of multibyte API's have been added to libc following C99 specification (ISO/IEC 9899:1999), and the Unix98 specification.

These APIs extend the already existing multibyte and wide character APIs in order to be able to:

- perform input and output of wide character, or multibyte character, or both
- perform general wide string manipulation
- provide extended capabilities for conversion between multibyte and wide character sequences

Several new design concepts have been introduced:

- Stream orientation
- Restartable APIs and the conversion state

Stream Orientation

A stream can be either wide character or byte oriented. The orientation of a stream is a concept based on an input/output model that assumes that characters are handled as wide-characters within an application and stored as multi-byte characters in files, and that all the wide-character input/output functions begin executing with the stream positioned at the boundary between two multi-byte characters.

After a stream is associated with a file, but before any operations are performed on the stream, the stream is without orientation. If a wide-character input or output function is applied to a stream without orientation, the stream becomes wide-oriented implicitly. Likewise, if a byte input or output operation is applied to a stream without orientation, the stream becomes byte-oriented implicitly. Once the stream becomes oriented, the orientation is fixed and cannot be changed until the stream is closed.

Restartable APIs and the Conversion State

A new set of APIs have been introduced to facilitate the conversion between multibyte character representations to wide character representations. These APIs use a new object type, `mbstate_t`, that can hold the conversion state information necessary to convert between sequences of multibyte characters and wide characters. The conversion state determines the behavior of a conversion between multibyte and wide-character encodings. For conversion from multibyte characters to wide characters, the conversion state stores information, such as the position, within the current multibyte character (as a sequence of characters or a wide character accumulator). For conversions in either direction, the conversion state stores the current shift state, if any, and possibly, the encoding rule.

As these APIs store the partial character information, a multibyte sequence can be processed one byte at a time, and the processing can be interrupted and continued (i.e. restarted) at some other point in time, so the new multibyte/wide conversion utilities are thus made restartable by using the information in the mbstate_t object.

How to Get MSE/Unix98 Behavior

In order to get MSE/Unix98 behavior, the programs have to be compiled with the `-D_XOPEN_SOURCE=500` macro definition and the variable `UNIX_STD` has to be defined in the environment.

Under the Korn, Bourne, and POSIX shells, this is done with:

```
UNIX_STD=98
export UNIX_STD
```

Under the C shell this is done using

```
setenv UNIX_STD 98
```

A `cc` compiler equal to HP92453-01 A.11.01.20 HP C Compiler or newer is required to get this functionality.

Below is a summary list of new and modified APIs. For further details, please refer to the corresponding manpages.

New Interfaces

The following APIs are newly added to libc and will not affect existing code:

btowc

`btowc()` returns the wide-character representation of a given single-byte character.

fwide

`fwide()` sets the stream orientation.

fwprintf, swprintf, wprintf

These APIs print formatted wide-character output.

fwscanf, swscanf, wscanf

These APIs process formatted wide-character input.

mbrlen

`mbrlen()` returns the number of bytes in a wide character. Note that the behavior of this function is affected by the `LC_CTYPE` category of the current locale.

mbrtowc

`mbrtowc()` converts a stream of bytes to a wide-character code. Note that the behavior of this function is affected by the `LC_CTYPE` category of the current locale.

mbsinit

`mbsinit()` determines whether the object pointed to by the first argument, which contains shift state information, describes an initial conversion state.

mbsrtowcs

`mbsrtowcs()` converts a character string to a wide-character string. Note that the behavior of this function is affected by the `LC_CTYPE` category of the current locale.

towctrans

towctrans() is provided for character transliteration. The current setting of the LC_CTYPE category should be the same as during the call to wctrans().

vfwprintf, vswprintf, vwprintf

These APIs are provided for printing wide-character formatted output of a stdarg argument. They are similar to fwprintf(3C) except that instead of being called with a variable number of arguments, they are called with an argument list as defined by stdarg.h.

wcrtomb

wcrtomb() converts a wide-character to a multibyte character. It determines the number of bytes needed to represent the character corresponding to the wide-character code whose value is specified by the second argument.

wcsrtombs

wcsrtombs() converts a wide-character string to a character string. Note that the behavior of this function is affected by the LC_CTYPE category of the current locale.

wcsstr

wcsstr() finds a substring in a wide-character string. Note that the behavior of this function is affected by the LC_CTYPE category of the current locale.

wctob

wctob() converts wide-character to single-byte.

wctrans

wctrans() defines character mapping in the current locale. Note that the values returned by wctrans() are valid until a call to setlocale() that modifies the category LC_CTYPE.

wmemchr, wmemcmp, wmemcpy, wmemmove, wmemset

These APIs operate with wide-character in memory areas:

— `wmemchr()` finds a wide-character in a memory array.
— `wmemcmp()` compares wide-characters in memory.
— `wmemcpy()` copies wide-character in memory.
— `wmemmove()` copies wide-characters in memory with overlapping areas.
— `wmemset()` sets wide-characters in memory.

Modified interfaces

The following APIs may have a change in behavior or a parameter type change that could affect existing HP-UX code when the Unix98 support is selected:

fprintf, printf, snprintf, sprintf, fscanf, scanf, sscanf

`printf(3C)`, `scanf(3C)` and related functions support the new qualifier l (the letter) to select wide character conversion in a given format string and set `errno` to EILSEQ if the data obtained from the input stream does not form a valid wide character.

fputwc, putwc, putwchar

The type of first argument is changed from `wint_t` to `wchar_t`.

freopen

Regardless of the mode of underlying stream, after a successful call to the `freopen()` function, the orientation of the stream is cleared and the associated `mbstate_t` object is set to describe an initial conversion state.

wcschr, wcsrchr

The type of second argument is changed from `wint_t` to `wchar_t`.

Potential Modifications to NLS as of HP-UX 11i

The following interfaces are subject to change or obsolescence in the next major release:

- The NLS configuration file is an HP-specific file and not specified by standards, and as such, it is subject to change in future releases. The `/usr/lib/nls/config` file may be moved to `/usr/lib/nls/loc/config`.
- The utility **getlocale(3C)** may be obsoleted or modified due to standards requirements.
- The command **findmsg(1)** may be obsoleted.
- The commands **dumpmsg(1), findstr(1), forder(1), insertmsg(1), nljust(1)**, and **eucset(1)** are HP-proprietary or non-standard utilities, which may be obsoleted over time as standard utilities emerge. Usage of these utilities will not be portable to other vendor's platforms.
- HP-proprietary 9.x locale names may be obsoleted in favor of ISO standard locale names. See Table 3-1, “New ISO Locale Names” on page 67 for a list of locale names that may be obsoleted.
- The locales ja_JP.kana8 and katakana may be obsoleted. The locales el_GR.greek8, iw_IL.hebrew8, and tr_TR.turkish8 are provided for backwards compatibility and may be obsoleted in the next major release in favor of `el_GR.iso88597`, `iw_IL.iso88598` and `tr_TR.iso88599`.
- The processing of right-to-left languages such as Arabic and Hebrew may change due to emerging standards. Some directly affected components include:

  — **strord(3C)**

  — **forder(1)**

  — **nljust(1)**

  — `nl_types.h`

  — `LANGOPTS` (environment variable)

  Other utilities may also be affected less directly.

- The `fr_FR.*` locales will eventually be modified to handle translations (**toupper(3C)**) from lower case characters with diacritics to upper case characters with diacritics. Currently most diacritics arelost in the lower case to upper case translation.
- In our efforts to comply with standards, the default codeset for many Western European languages may be changed from ROMAN8 to ISO 8859/1 in future releases.
- X/Open is investigating an interface that will retrieve the locale settings. When this interface is finalized, it may be provided in a future release.
- Future releases of the fontserver will not support the following Hewlett-Packard proprietary XLFD extensions:

  — `slant` (obliquing)

  — `addstylename` (mirroring and rotation)

  — `pixelsize` and `pointsize` (anamorphic scaling)

  The Hewlett-Packard proprietary weightname extension used to specify darker or lighter fonts will be supported.
- The Hewlett-Packard syntax for the charset encoding extension is obsolete and will be removed in a future version of the font server.
- The second component of the HP-UX 10.0 default `NLSPATH` may be obsoleted in the next major release leaving only this path:

  `/usr/lib/nls/msg/%L/%N.cat`
- Future MSE changes:

  ISO/MSE 9899:1990/Amendment 1:1994(E) defines extensions to ISO/MSE:1990 Programming Language -C. The amendment provides a more complete and consistent set of utilities for application programming using multi-byte and wide-character functions, as well as providing alternate spellings for language-specific tokens. ANSI C has since adopted this amendment, and X/Open will eventually align with this amendment.

A Bibliography

The below publications are referenced in this document by using specifying the first four letters of the author's name in brackets. Example:

[AUTH]

where [AUTH] corresponds to the bibliographic entry:

#. [AUTH] Author Last Name, Author First Name; *Title*; Publisher, Copyright date

1. [MADE] Madell, Tom; Parsons, Clark; Abegg, John; *Developing And Localizing International Software*; Prentice Hall, 1994
2. [MCFA] McFarland, Thomas C.; X Windows On The World: *Developing Internationalized Software With X, Motif, and CDE*; Prentice Hall, 1996
3. [ODON] O'Donnell, Sandra Martin; *Programming For The World: A Guide To Internationalization*; Prentice Hall, 1994

Glossary

AJEC Advanced Japanese EUC

ASE Asian System Environment

ASX Asian System Extension (formerly known as NLIO; as of HP-UX 10.0, renamed to ASE)

CCDC Chinese Code for Data Communication

CNS Chinese National Standard

EBCDIC Extended Binary Coded Decimal Interchange Code

EUC Extended UNIX Code

IPSJ Information Processing Society of Japan

ITSCJ Information Technology Standards Commission of Japan

JIS Japanese Industrial Standard

JNP Japanese National Profile

OSF Open Software Foundation

PRO Precision RISC Organization

SJIS Shift-JIS

UDC User Defined (or Definable) Character

UI UNIX International

UJIS old name for Japanese EUC (EUCJP)

USLP UNIX System Laboratories Pacific

VDC Vendor Defined (or Definable) Character