



Solaris Internationalization Guide For Developers

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 805-4123-10
October 1998

Copyright 1998 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, SunDocs, Java, the Java Coffee Cup logo, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. SunOS, Solaris, X11, SPARC, UNIX, PostScript, OpenWindows, AnswerBook, SunExpress, SPARCprinter, JumpStart, Xlib

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 1998 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, Californie 94303-4900 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, SunDocs, Java, le logo Java Coffee Cup, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Contents

Preface xvii

1. Solaris Internationalization Overview 1

New Internationalization Features in Solaris 7 1

Internationalization and Localization 2

 Basic Steps in Internationalization 3

What Is a Locale? 3

 Full and Partial Locales 4

Locales 4

 Locale Categories 5

Using Locale Categories for Localization 6

 Time Formats 6

 Date Formats 7

 Numbers 8

 Currency 9

 Word and Letter Differences 10

Keyboard Differences 12

Other Differences 13

 Punctuation 13

 Symbols 13

	Measurements	13
	Gender	13
	Titles and Addresses	13
	Paper Sizes	14
	<i>Creating Worldwide Software: The Book</i>	14
	Overview	15
2.	Contents of the Base Solaris Product	17
	Summary of the Base Product	17
	Core Set of Locales	18
	New Locales	19
	Extended Set of Locales	22
	Unicode Locale: <code>en_US.UTF-8</code>	23
	User Locales in the Base Solaris Product	24
	Multiple Key Compose Sequences for Locales	25
	Keyboard Support in the Base Solaris 7 Product	26
	Changing Between Keyboards on SPARC	26
	Changing Between Keyboards on x86	27
	Codesets for x86	28
	Locales in the Base Installation	28
	Using JumpStart	29
3.	Contents of the Localized Solaris 7 Products	31
	The European Localized Solaris 7 Product	31
	Font Formats	44
	Summary of Asian Locales	44
	Korean in the Solaris 7 Product	45
	Chinese: Simplified and Traditional	47
	Japanese Input Systems	53
	Korean Solaris 7 Product	58

	61
How to Use the <code>iconv</code> Command	61
4. Overview of <code>en_US.UTF-8</code> Locale Support	63
<code>en_US.UTF-8</code> Locale Support Overview	63
System Environment	65
Code Conversions	71
Script Selection and Input Modes	75
Unicode Hexadecimal Code Input Method Input Mode	89
Table Lookup Input Method Input Mode	89
Input Mode Switch Key Sequence Summary	89
Printing	90
DtMail	91
Programming Environment	92
Font Set Used with <code>x</code> Applications	92
XmFontList Definition as CDE/Motif Applications	94
5. Installation	95
Adding Packages	95
▼ How to Add Packages to a Standalone System	95
Installing Software From a Mounted CD	96
Installing Software From a Remote Package Server	97
Installing the Localization Product	98
European Packages	98
French Files	99
German Files	100
Italian Files	101
Spanish Files	102
Swedish Files	103
Detailed Descriptions of European Files	104

	European Codesets	113
	European Font Packages	113
	Asian Packages	114
	Description of General Packages	124
	Asian Localization Packages Disk Space	158
6.	Internationalization Framework in the Solaris 7 Environment	163
	Codeset Independence Support	163
	The CSI Approach	164
	CSI-enabled Commands	164
	Solaris 7 CSI-enabled Libraries	166
	Locale Database	167
	Process Code Format	167
	Multibyte Support Environment (MSE)	167
	Dynamically Linked Applications	168
	libw and libintl	169
	ctype Macros	170
	Internationalization APIs in libc	171
	genmsg Utility	179
7.	X/DPS	181
	Localization Resource Category	182
	Information on Language Interpreters	182
8.	Desktop Environments	183
	Overview	183
	Locales	184
	Integrating Fonts	184
	Input Methods	185
	Internationalization and CDE	185
	Matching Fonts to Character Sets	185

	Storage of Localized Text	186
	Xlib Dependencies	186
	Message Guidelines	186
	Internationalization and Distributed Networks	187
	Mail Interchange	187
	OpenWindows	188
9.	Printing	189
	Localization Printing Support Under the Solaris 7 Operating Environment	189
	European Printing Support	189
	Asian Multibyte Printing Support	191
	CDE Font Downloader	192
	Technical Description	192
	Reference Documents	193
10.	Complex Text Layout	195
	Overview of CTL Technology	195
	Overview of CTL Architecture	196
	Changes in Motif to Support CTL Technology	196
	XmDirection	197
	Description	197
	For More Information	197
	XmStringDirection	198
	Description	198
	Related Information	198
	XmRendition	198
	New Resources	199
	Additional Behavior	200
	XmText, XmTextField	200
	Description	200

New Resources	201
Action Routines	202
Additional Behavior	202
Action Routines	203
XmTextFieldGetLayoutModifier	211
Purpose	211
Synopsis	211
Description	212
Return Value	212
Related Information	212
XmTextGetLayoutModifier	212
Purpose	212
Synopsis	212
Description	212
Return Value	213
Related Information	213
XmTextFieldSetLayoutModifier	213
Purpose	213
Synopsis	213
Description	213
Related Information	213
XmTextSetLayoutModifier	214
Purpose	214
Synopsis	214
Description	214
Related Information	214
XmStringDirectionCreate	214
Synopsis	214

Description	214
Related Information	215
UIL	215
How to Develop CTL Applications	215
Layout Direction	215
Creating a Rendition	217
Editing a Rendition	218
Related Information	218
Creating a Render Table in a Resource File	218
Creating a Render Table in an Application	219
Horizontal Tabs	220
Mouse Selection	221
Keyboard Selection	222
Text Resources and Geometry	222
Porting Instructions	223
Index	225

Tables

TABLE P-1	Typographic Conventions	xx
TABLE 1-1	International Time Formats	6
TABLE 1-2	International Date Formats	7
TABLE 1-3	International Numeric Conventions	8
TABLE 1-4	International Monetary Conventions	9
TABLE 1-5	Common International Page Sizes	14
TABLE 2-1	Core Set of Locales in SUNWploc and SUNWplow	18
TABLE 2-2	New or Changed User Locales	19
TABLE 2-3	New User Locales To Support the Euro Currency	21
TABLE 2-4	Extended Set of Locales in SUNWploc1 and SUNWplow1	22
TABLE 2-5	User Locales Included in Solaris 7 Product	24
TABLE 2-6	Layouts for Type 4 Keyboards	26
TABLE 2-7	Locales Offered at Installation	28
TABLE 3-1	European 7 Locales	31
TABLE 3-2	Eastern European Locales in the Solaris 7 Product	36
TABLE 3-3	iconv Support	37
TABLE 3-4	New Locales and Corresponding Codeset Names	39
TABLE 3-5	Summary of Asian Locales	44
TABLE 3-6	Codeset Conversions Supported for Korean ko, ko.UTF-8	46

TABLE 3-7	Solaris 7 TrueType Fonts for the zh Locale	48
TABLE 3-8	Solaris 7 Bitmap Fonts for the zh Locale	48
TABLE 3-9	TrueType Fonts for the zh.GBK Locale	48
TABLE 3-10	Bitmap Fonts for the zh.GBK Locale	49
TABLE 3-11	Codeset Conversions for Simplified Chinese	49
TABLE 3-12	Traditional Chinese Truetype Fonts for the zh_TW Locales	50
TABLE 3-13	Traditional Chinese BitMap Fonts for the zh_TW Locales	51
TABLE 3-14	Traditional Chinese TrueType Fonts for the zh_TW.BIG5 Locales	51
TABLE 3-15	Traditional Chinese BitMap Fonts for the zh_TW.BIG5 Locales	51
TABLE 3-16	Codeset Conversions for Traditional Chinese	51
TABLE 3-17	Japanese Input Systems	53
TABLE 3-18	Japanese TrueType Fonts	53
TABLE 3-19	Japanese Bitmap Fonts	54
TABLE 3-20	iconv Conversion Support	54
TABLE 3-21	Solaris 7 Korean CID/Type 1 Fonts for the ko Locale	58
TABLE 3-22	Solaris 7 Korean Bitmap Fonts for the ko Locale	59
TABLE 3-23	Solaris 7 Korean CID/Type 1 Fonts for the ko.UTF-8 Locale	59
TABLE 3-24	Solaris 7 Korean Bitmap Fonts for the ko.UTF-8 Locale	59
TABLE 3-25	Korean ICONV	60
TABLE 4-1	32-bit STREAMS Modules Supported by en_US.UTF-8	66
TABLE 4-2	64-bit STREAMS Modules Supported by en_US.UTF-8	66
TABLE 4-3	Available Code Conversions in en_US.UTF-8	71
TABLE 4-4	Common Latin-1 Compose Sequences for Sparc	76
TABLE 4-5	Common Latin-2 Compose Sequences	80
TABLE 4-6	Common Latin-4 Compose Sequences	82
TABLE 4-7	Common Latin-5 Compose Sequences	84
TABLE 4-8	Common Latin-9 Compose Sequences	84
TABLE 4-9	Input Mode Switch Key Sequences	89

TABLE 5-1	Pan-European Files for Localization and Windowing	98
TABLE 5-2	French Files for Localization and Windowing	99
TABLE 5-3	German Files for Localization and Windowing	100
TABLE 5-4	Italian Files for Localization and Windowing	101
TABLE 5-5	Spanish Files for Localization and Windowing	102
TABLE 5-6	Swedish Files for Localization and Windowing	103
TABLE 5-7	European Package Descriptions	104
TABLE 5-8	Font Packages in the Solaris 7 Product	113
TABLE 5-9	Common Asian Packages for Localization and Windowing	114
TABLE 5-10	Korean Packages for Localization and Windowing	115
TABLE 5-11	Simplified Chinese Packages for Localization and Windowing	116
TABLE 5-12	Traditional Chinese Packages for Localization and Windowing	118
TABLE 5-13	Simplified Chinese Packages	120
TABLE 5-14	Japanese Packages for Localization and Windowing	122
TABLE 5-15	Thai Packages for Localization and Windowing	124
TABLE 5-16	General Packages	124
TABLE 5-17	Korean Packages	125
TABLE 5-18	Traditional Chinese Packages	126
TABLE 5-19	zh.GBK Packages	130
TABLE 5-20	Thai Language Packages	131
TABLE 5-21	Japanese Packages	132
TABLE 5-22	ko Locale	138
TABLE 5-23	ko.UTF-8 Locale	140
TABLE 5-24	zh Locale	141
TABLE 5-25	zh.GBK Locale	142
TABLE 5-26	th_TH Locale	143
TABLE 5-27	zh_TW Locale	144
TABLE 5-28	zh_TW.BIG5 Locale	146

TABLE 5-29	ja/ja_JP.PCK Common Packages	146
TABLE 5-30	ja Locale	148
TABLE 5-31	ja_JP.PCK Locale	150
TABLE 5-32	CDE Packages	152
TABLE 5-33	MB Required for Software Groups (SPARC)	158
TABLE 5-34	MB Required for Software Groups (x86)	158
TABLE 5-35	MB Required for ko and ko plus ko.UTF-8 (SPARC)	159
TABLE 5-36	MB Required for ko and ko plus UTF-8 (x86)	159
TABLE 5-37	MB Required for zh_TW and zh_TW.BIG5 (SPARC)	160
TABLE 5-38	MB Required for zh_TW and zh_TW.BIG5 (x86)	160
TABLE 5-39	MB Required for zh and zh.GBK (SPARC)	160
TABLE 5-40	MB Required for zh and zh.GBK (x86)	161
TABLE 6-1	CSI-enabled Commands in Solaris 7	165
TABLE 6-2	Stub Entry Points in libw and libintl	169
TABLE 6-3	Internationalization APIs in libc	171
TABLE 9-1	prolog.ps Fonts	190
TABLE 9-2	Japanese Printer Support	191

Figures

Figure 4-1	Cyrillic Keyboard	85
Figure 4-2	Greek Euro Keyboard	86
Figure 4-3	Greek UNIX Keyboard	86
Figure 4-4	Arabic Keyboard	87
Figure 4-5	Hebrew Keyboard	88
Figure 4-6	Thai Keyboard	88
Figure 10-1	Tabbing Behavior	221

Preface

The *Solaris Internationalization Guide for Developers* describes internationalization features that are new in Solaris™ .7. It contains important information on how to use Solaris .7 to build global software products that support various languages and cultural conventions.

Specifically, this guide contains:

- Guidelines and tips for developers on how to use Solaris 7 to write applications for international markets.
- An overall view of internationalization topics that apply to various layers within the Solaris environment.
- Pointers to more detailed documentation.

Where appropriate, this guide points you to other guides in the documentation set that contain additional or more detailed information on internationalization features in this release.

Who Should Use This Guide

This guide is intended for software developers who want to design global products and applications for the Solaris 7 environment software developers.

This guide assumes knowledge of the C programming language, and a few chapters discuss X11™ NeWS window system toolkits.

All operating system information pertains to the Solaris7 SunOS™ 5.7 operating environment. The hardware platforms covered are SPARC™ and Intel x86. For the most part, support for these architectures is identical, but a note appears when this is not the case. SunOS 5.6 SPARC architectures x86 architectures (SPARC and x86)

Organization and Summary

The chapters in this guide are organized as follows:

- Chapter 1, tells what's new and provides an overview of the localized products available on the base Solaris release, the European localized release, and the Asian localized releases.
- Chapter 2, describes the contents of the Solaris 7 base product as it relates to locales.
- Chapter 3, describes Codeset Independence (CSI) support for Extended UNIX® Code (EUC) and non-EUC codesets.
- Chapter 4, covers the system environment, code conversions, script selection, printing, and the programming environment.
- Chapter 5, describes the procedures for installing the localization packages.
- Chapter 6, contains details about the internationalization features incorporated into this release.
- Chapter 7, contains a detailed look at the procedures to write a localized version of codesets, formats, collation, and messaging.
- Chapter 8, covers the Solaris desktop environments: the Common Desktop Environment (CDE) and OpenWindows™. The section on CDE has an overview of the application internationalization process, including locale management, localized resources, and font management.
- Chapter 9, covers printing support under the Solaris 7 operating environment, with specific information for European and Asian printing.
- Chapter 10, includes information about CTL extensions that enable Motif APIs to support writing systems that require complex transformation between logical and physical text representations, such as Arabic, Hebrew, and Thai.

Related Books and Sites

For information about the Java development Kit, see <http://java.sun.com/docs/books/tutorial/i18n/index.html> <http://java.sun.com/docs/books/tutorial/i18n/index.html>.

Tuthill, Bill and David Smallberg. *Creating Worldwide Software: Solaris International Developer's Guide*, 2nd edition. Mountain View, California, Sun Microsystems Press, 1997. Available through books@sun.com and www.sun.com/books/. The book offers a general overview of the internationalization process under the Solaris operating system.

Common Desktop Environment: Internationalization Programmer's Guide. Mountain View, California, SunSoft Press, 1996. The CDE documentation set can be ordered by title through SunExpress. The CDE Programmer's guide is also part of the CDE Developer's AnswerBook™ set that is shipped on the Solaris documentation CD. Available through the SunDocs program (see "Ordering Sun Documents" on page xix). Contains information on locale management, font management, distributed networks, User Interface Language (UIL), Xt, and Xlib dependencies.

OSF/Motif Programmer's Guide, Release 1.2. Englewood Cliffs, New Jersey, Prentice-Hall, 1993. The Open Software Foundation's (OSF) *Guide* describes how to use the OSF/Motif application programming interface to create Motif applications. It presents an overview of Motif widget set architecture, explains the Motif toolkit, and gives models and examples of Motif applications.

OSF/Motif Programmer's Reference, Release 1.2. Englewood Cliffs, New Jersey, Prentice-Hall, 1992. The Open Software Foundation's (OSF) *Reference* is the collection of reference pages to OSF/Motif commands, functions, toolkit, window manager, user interface language commands, and functions.

PostScript Language Reference Manual, Second Edition. Adobe Systems Inc., Addison-Wesley, 1990. The standard reference work for PostScript covers the fundamentals of PostScript as a device-independent printing language.

PostScript Language Reference Manual Supplement. Adobe Systems Inc., 1994.

Programming the Display PostScript System with X. Reading, Mass., Adobe Systems Inc., Addison-Wesley, 1993. For application developers working with X Windows and Display PostScript to produce information for the screen display and the printer output.

OLIT Reference Manual. Sun Microsystems, 1994.

XView Developer's Notes. O'Reilly & Associates, 1992.

Ordering Sun Documents

The SunDocs program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals from SunDocs.

For a list of documents and how to order them, see the catalog section of the SunExpress™ Internet site at <http://www.sun.com/sunexpress>.

Typographic Conventions

Table P-1 describes the typographic conventions used in this guide.

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name% You have mail.</code>
AaBbCc123	What you type, contrasted with on-screen computer output	<code>machine_name% su</code> Password:
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type <code>rm filename</code> .
<i>AaBbCc123</i>	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be root to do this.

Solaris Internationalization Overview

The Solaris 7 product includes full Unicode 2.0 support, as defined in ISO-10646, for selected locales. Solaris 7 is a major release for Sun's international markets. It includes a number of new features for Asian customers and significantly expands language support for Eastern Europe and the Baltic States.

New Internationalization Features in Solaris 7

- Increased Unicode support
 - Unicode 2.0 supported through English and Korean locales.
 - Six new UTF-8 Unicode locales added: French, German, Italian, Spanish, Swedish, and Europe. (Europe returns the Euro as the default currency symbol).
 - UTF-8 locales support multiple input and output for all European locales as well as Korean, Japanese, Traditional Chinese and Simplified Chinese. Enhancements have been made to the `en_US.UTF-8` locale so that users can input and display text from different writing scripts such as Japanese, Thai, Chinese, Hebrew, Arabic, Korean and Russian. Users can easily switch between the scripts without having to change or install a new locale.
- Codeset conversion utilities have been enhanced for better data interoperability in the Russian locale.
- Expanded language coverage

- Euro currency. All foreign exchange, banking, and finance industries in the European community are converting from using their local currencies to using the Euro. Solaris 7 software has added support for the Euro currency with six new user locales.
- CDE applications included in Asian versions to support Complex Text Layout (CTL) locales. Complex text support has been integrated for complex text layout languages, which require special text pre-processing to handle bidirectional, composite, and context-sensitive text.
- Solaris 7 software supports Motif 2.1, which includes five new Motif widgets. Motif 2.1 is MT-safe and includes software for CTL locale support.
- zh.GBK locale for Simplified Chinese in the People's Republic of China. This feature supports GBK Character Set, a superset of GB2312 which is used in the zh locale.
- The English and European translations for the Solaris 7 operating environment have been combined on a single CD. As a result, more locale selections are available during installation of this combined CD than were seen previously.
- The Desktop Font Downloader allows users to download, remove, re-encode and convert fonts, check status, and perform other administrative tasks on a PostScript printer.

Internationalization and Localization

Internationalization is the process of making software portable between languages or regions, while localization is the process of adapting software for specific languages or regions. International software can be developed using interfaces that modify program behavior at run time in accordance with specific cultural requirements. Localization involves establishing on-line information to support a language or region, called a *locale*.

Unlike software that must be completely rewritten before it can work with different native languages and customs, internationalized software does not require rewriting. It can be ported from one locale to another without change. The Solaris system is internationalized, providing the infrastructure and interfaces you need to create internationalized software. and Chapter 4 describe what facilities are available and how to use them.

Internationalization and localization are different procedures.

- Internationalization is the process of making software that is independent of any locale. It can then be easily adapted to specific locales.

The following localized products are available in the Solaris 7 operating environment:

- English and European Solaris (German, French, Spanish, Swedish, Italian)

- Simplified Chinese Solaris
- Traditional Chinese Solaris
- Japanese Solaris
- Korean Solaris

Basic Steps in Internationalization

An internationalized application's executable image is portable between languages and regions. To internationalize software, you should:

- Use the interfaces described in this book to create software whose environment can be modified dynamically without the necessity of recompiling the software.
- Separate software into executable and messages. The messages include all printable and displayable messages that the user sees. Keep the message strings in a message catalog.

Message strings are translated for a language and a region. A *locale* includes the message strings and methods to specify sorting.

Locales are not the same as a language. A language may contain various regions. For example, French is spoken in France and Canada, but each country has different ways of displaying monetary and time information.

To use a localized version of a product, the user sets the environment variables (described in "Locale Categories" on page 5). The product then displays the user messages in their translated form. , , Date, time, currency and other information is formatted and displayed according to locale-specific conventions.

What Is a Locale?

A locale may be composed of both a base language and the country of use. This allows for specific differences by country such as currency units notation.

The key concept for application programs is that of a program's *locale*. The locale is an explicit model and definition of a native-language environment. The notion of a locale is explicitly defined and included in the library definitions of the ANSI C Language standard.

The locale consists of a number of categories for which there is language-dependent formatting or other specifications. A program's locale defines its codesets, date and time formatting conventions, monetary conventions, decimal formatting conventions, and collation (sort) order.

A locale name contains language, territory, and possibly codeset, although territory is dropped when not needed. Codeset is usually assumed. For example, German is `de`, an abbreviation for Deutsch, while Swiss German is `de_CH`, `CH` being an abbreviation for Confederation Helvetica.

Note - More than one locale may be associated with a particular language. This allows for regional differences such as currency notation. For example, an English-speaking user in the United States can select the `en_US` locale (English for the United States). An English-speaking user in Great Britain can select `en_GB` (English for Great Britain).

Generally the locale name is specified by the `LANG` environment variable. Locale categories are subordinate to `LANG`, but may be set separately, in which case they override `LANG`. If `LC_ALL` is set, it overrides not only `LANG`, but all the separate locale categories as well.

Full and Partial Locales

A full Solaris locale has all of the listed functions and the localized system messages in that language. The German `de` locale is a *full* locale. A German language user sees all system messages in German.

Partial locales have the listed functions but they don't provide localized messages. For example, the Russian `ru` locale can process input, output, sorting, and so on, but it does not have localized messages in Russian. For this reason it is a *partial* locale.

Some partial locales do use non-English messages because there may be a full locale with the localized messages. For example, the `de_AT` is a partial locale for Austria. Austrians speak German but use a different currency. The Austrian locale is a subset of the German `de` locale. It displays messages in German and currency in Austrian shillings instead of German marks.

Locales

Different cultures use different conventions for writing the date, the time, numbers, currency, delimiting words and phrases, and quoting material.

A locale defines the behavior of a program at runtime according to a language or cultural region's conventions. Throughout the system, a locale determines the behavior of the following:

- Encoding and processing of text data
- Identifying the language and encoding of resource files and their text values

- Rendering and layout of text strings
- Interchanging text that is used for interclient text communication
- Encoding and decoding for interclient text communication
- Selecting the input method (that is, which codeset is generated) and the processing of text data
- Font and icon files that are culturally specific
- Actions and file types
- User Interface Definition (UID) files
- Date and time formats
- Numeric formats
- Monetary formats
- Collation order
- Format for informative and diagnostic messages and interactive responses

The Solaris environment separates language and culture-dependent information from the application and saves it outside the application.

By separating the language and culture-dependent information from the application, the developer does not need to translate, rewrite, or recompile the application for each market. The only requirement to enter a new market is to localize the external information to the local language and customs.

Locale Categories

The locale categories are as follows:

- `LC_CTYPE`
Controls the behavior of character handling functions.
- `LC_TIME`
Specifies date and time formats, including month names, days of the week, and common full and abbreviated representations.
- `LC_MONETARY`
Specifies monetary formats. Few SunOS system commands or library routines actually use this category.
- `LC_NUMERIC`
Specifies the decimal separator (or radix character) and the thousands separator.
- `LC_COLLATE`
Specifies the sorting order for a locale and the string conversions required to attain this ordering.

- LC_MESSAGES

Specifies the language in which the localized messages are written.

- LO_LTYPE

Specifies the language engine which provides information about language rendering. Language rendering (or text rendering) consists of text shaping and directionality.

Using Locale Categories for Localization

The localization of a product should be done in consultation with native users in that target language or region. Certain styles and information styles and formats may seem perfectly obvious and universal to the developer, but to the user, these look either awkward, wrong, or even offensive. The following pages describe the elements that the Solaris operating environment allows you to control and specify so that you can successfully internationalize your product.

Time Formats

Table 1-1 shows some of the ways to write 11:59 P.M.

TABLE 1-1 International Time Formats

Locale	Format
Canadian	23:59
Finnish	23.59
German	23.59 Uhr
Norwegian	Kl 23.59
U.K.	11.59 PM
Thai	13:10 PM

Time is represented by both a 12-hour clock and a 24-hour clock. The hour and minute separator can be either a colon (:) or a period (.).

Time zone splits occur between and within countries. Although a time zone can be described in terms of how many hours it is ahead of, or behind, Greenwich Mean Time (GMT), this number is not always an integer. For example, Newfoundland is in a time zone that is half an hour different from the adjacent time zone.

Daylight Savings Time (DST) starts and ends on different dates that can vary from country to country.

Date Formats

Table 1-2 shows some of the date formats used around the world. Note that even within a country, there may be variations.

TABLE 1-2 International Date Formats

Locale	Convention	Example
Canadian (English and French)	yyyy-mm-dd	1998-08-13
Danish	dd/mm/yy	13/08/98
Finnish	dd.mm.yyyy	13.08.1998
French	dd/mm/yy	13/08/98
German	dd.mm.yy	13.08.98
Italian	dd.mm.yy	13.08.98
Norwegian	dd.mm.yy	13.08.98
Spanish	dd-mm-yy	13-08-98
Swedish	yyyy-mm-dd	1998-08-13
UK-English	dd/mm/yy	13/08/98
US-English	mm-dd-yy	08-13-98
Thai	dd/mm/yyyy	10/12/2539

Numbers

Decimal and Thousands Separators

Great Britain and the United States are two of the few places in the world that use a period to indicate the decimal place. Many other countries use a comma instead. The decimal separator is also called the *radix* character. Likewise, while the U.K. and U.S. use a comma to separate thousands groups, many other countries use a period instead, and some countries separate thousands groups with a thin space. Table 1-3 shows some commonly used numeric formats.

TABLE 1-3 International Numeric Conventions

Locale	Large Number
Canadian (English and French)	4 294 967 295,00
Danish	4.294.967.295,00
Finnish	4.294.967.295,00
French	4.294.967.295,00
German	4 294 967 295,00
Italian	4.294.967.295,00
Norwegian	4.294.967.295,00
Spanish	4.294.967.295,00
Swedish	4.294.967.295,00
UK-English	4,294,967,295.00
US-English	4,294,967,295.00
Thai	4,294,967,295.00

Data files containing locale-specific formats will be misinterpreted when transferred to a system in a different locale. For example, a file containing numbers in a French format is not useful to a U.K.-specific program.

List Separators

There are no particular locale conventions that specify how to separate numbers in a list. They are sometimes comma-delimited in the UK and the U.S., but often spaces and semicolons are used.

Currency

Currency units and presentation order vary greatly around the world. Table 1-4 shows monetary formats in some countries.

TABLE 1-4 International Monetary Conventions

Locale	Currency	Example
Canadian (English)	Dollar (\$)	\$1 234.56
Canadian (French)	Dollar (\$)	1 234.56\$
Danish	Kroner (kr)	kr.1.234,56
Finnish	Markka (mk)	1.234 mk
French	Franc (F)	F1.234,56
German	Deutsche Mark (DM)	1,234.56DM
Italian	Lira (L)	L1.234,56
Japanese	Yen	41,234 Yen
Norwegian	Krone (kr)	kr 1.234,56
Spanish	Peseta (Pts)	1.234,56Pts
Swedish	Krona (Kr)	1234.56KR
UK-English	Pound	31,234.56 pounds
US-English	Dollar (\$)	\$1,234.56
Thai	Baht	2539 Baht

Note - Local and international symbols for currency can differ. For example, the designation for the French franc is “F” in France but this is often written as FRF’ internationally to distinguish it from other francs, such as the Swiss franc or the Polynesian franc.

Be aware also that a *converted* currency amount may take up more or less space than the original amount. To illustrate: \$1,000 can become L1.307.000.

Word and Letter Differences

Word Delimiters

In English, words are separated by a space character. In languages such as Chinese, Japanese and Thai, however, there is often no delimiter between words.

Word Order

The order of words in phrases and sentences varies between languages. For instance, the order of the words “cat” and “black” in “a black cat” is reversed in the equivalent Spanish phrase, “uno gato negro.” And in French, the negatives “ne” and “pas” surround the word they negate, as in the phrase “I do not speak,” which in French is “Je ne parle pas.”

Sort Order

Sorting order for particular characters is not the same in all languages. For example, the character “ö” sorts with the ordinary “o” in Germany, but sorts separately in Sweden, where it is the last letter of the alphabet. In some languages, characters have weight to determine the priority of the character sequences. For example, in Thai, the Thai dictionary defines sorting through the sequences of characters which have different weights.

Character Sets

Number of Characters

While the English alphabet contains only 26 characters, some languages contain many more characters. Japanese, for example, can contain over 40,000 characters; Chinese even more.

Western European Alphabets

The alphabets of most western European countries are similar to the standard 26-character alphabet used in English-speaking countries, but there are often some additional basic characters, some marked (or accented) characters, and some ligatures.

Japanese Text

Japanese text is composed of three different scripts mixed together: Kanji ideographs derived from Chinese, and two phonetic scripts (or syllabaries), Hiragana and Katakana.

Although each character in Hiragana has an equivalent in Katakana, Hiragana is the most common script, with cursive rather than block-like letter forms. Kanji characters are used to write root words. Katakana is mostly used to represent “foreign” words—words “imported” from languages other than Japanese.

There are tens of thousands of Kanji characters, but the number commonly used has been declining steadily over the years. Now only about 3500 are frequently used, although the average Japanese writer has a vocabulary of about 2000 Kanji characters. Nonetheless, computer systems must support more than 7000 because that is what the Japan Industry Standard (JIS) requires. In addition, there are about 170 Hiragana and Katakana characters. On average 55% of Japanese text is Hiragana, 35% Kanji, and 10% Katakana. Arabic numerals and Roman letters are also present in Japanese text.

Although it is possible to avoid the use of Kanji completely, most Japanese readers find text containing Kanji easier to understand.

Korean Text

Korean text can be written using a phonetic writing system called Hangul. Hangul has more than 11,000 characters, which are composed by 19 consonants, 21 vowels and optional 27 consonants. About 3,000 Hangul characters from the whole Hangul characters are usually used in Korean computer systems. Korean also uses ideographs based on the set invented in China, called Hanja. Korean text requires over 6,000 Hanja characters. Hanja is used mostly to avoid confusion when Hangul would be ambiguous. Hangul characters are formed by combining consonants and vowels. After combining them together, they can compose one syllable, which is a Hangul character. Hangul characters are often arranged in a square, so that the group takes up the same space as a Hanja character. Arabic numerals, Roman letters and special symbol characters are also present in Korean text.

Thai Text

A Thai character can be defined as a column position on a display screen with four display cells. Each column position can have up to three characters. The composition

of a display cell is based on the Thai character's classification. Some Thai characters can be composed with another character's classification. If they can be composed together, both characters will be in the same cell. Otherwise, they will be in separate cells.

Chinese Text

Chinese usually consists entirely of characters from the ideographic script called Hanzi. In the People's Republic of China (PRC) there are about 7000 commonly used Hanzi characters in GB2312 (zh locale) and more than 20,000 characters in the GBK (zh.GBK) locale. In Taiwan, current standards require more than 13000 characters; 6000 others have been recently standardized but are considered rare.

If a character is not a root character, it usually consists of two or more parts, two being most common. In two-part characters, one part generally represents meaning, and the other represents pronunciation. Occasionally both parts represent meaning. The radical is the most important element, and characters are traditionally arranged by radical, of which there are several hundred. The same sound can be represented by many different characters, which are not interchangeable in usage. The same character can even have different sounds.

Some characters are more appropriate than others in a given context—the appropriate one is distinguished phonetically by the use of tones. By contrast, spoken Japanese and Korean lack tones.

There are several phonetic systems for representing Chinese. In the People's Republic of China the most common is pinyin, which uses roman characters and is widely employed in the West for place names such as Beijing. The Wade-Giles system is an older phonetic system, formerly used for place names such as Peking. In Taiwan zhuyin (or bopomofo), a phonetic alphabet with unique letter forms, is often used instead.

Commercial applications, particularly those that deal with people's names, need to consider the impact of codeset expansion. Many Chinese people have names containing characters that do not exist in any standard codeset. Space needs to be provided in unassigned codesets to deal with this issue.

Keyboard Differences

Not all characters on the U.S. keyboard appear on other keyboards. Similarly, other keyboards often contain many characters not visible on the U.S. keyboard. However, on Sparc machines, the Compose key can be used to produce any character in the ISO Latin-1 codeset on any keyboard that supports it.

Note - The Compose key can be used with English or European locales, but not with Korean, Chinese, or Japanese locales.

Other Differences

Punctuation

Both the position and the type of punctuation symbols can vary between languages. In Spanish, “¿” and “¡” appear at the beginnings of sentences, while in Finnish colons (:) can occur inside words.

Symbols

Commonly used symbols in one culture often have no meaning in another culture. For example, because the common U.S. rural mailbox does not exist in other countries, it would not make a universal email icon.

Measurements

While most countries now use the metric system of measurement, the United States, parts of Canada, and the Great Britain (albeit unofficially) still use the imperial system. The symbols for feet (') and inches (") are not understood in all countries.

Gender

The spelling of adjectives, articles, and nouns are gender-dependent in some languages. In French, for example, “un petit gamin” and “une petite gamine” both mean “a cute kid.” The first expression, however, refers to a boy, and the second expression to a girl. Also, neuter objects in English (“a computer” for example) have gender in other languages (“un ordinateur” is a masculine noun in French).

Titles and Addresses

Mr., Miss, Mrs., and Ms. are common titles in the U.S. but are not used in many other countries. The order in which addresses are written is different too.

Address formats differ from country to country. In many countries, the postal code includes letters as well as numbers.

The order of writing addresses differs from country to country. The order of writing first name and last name is also different.

Paper Sizes

Within each country a small number of paper sizes are commonly used, normally with one of those sizes being much more common than the others. Most countries follow ISO Standard 216 “Writing paper and certain classes of printed matter—Trimmed sizes—A and B series.”

Internationalized applications should not make assumptions about the page sizes available to them. The Solaris system provides no support for tracking output page size; this is the responsibility of the application program. Table 1-5 shows Common International Page Sizes.

TABLE 1-5 Common International Page Sizes

Paper Type	Dimensions	Countries
ISO A4	21.0 cm by 29.7 cm	Everywhere except U.S.
ISO A5	14.8 cm by 21.0 cm	Everywhere except U.S.
JIS B4	25.9 cm by 36.65 cm	Japan
JIS B5	18.36 cm by 25.9 cm	Japan
U.S. Letter	8.5 inch by 11 inches	U.S. and Canada
U.S. Legal	8.5 inch by 14 inches	U.S. and Canada

Creating Worldwide Software: The Book

The book *Creating Worldwide Software*, 2nd edition, by Bill Tuthill and David Smallberg (SunSoft Press, 1997), is a guide to localizing for the Solaris platform. The book is recommended for developers who work with the Solaris system. See “Related Books and Sites” on page xviii for a full citation.

Overview

The book *Creating Worldwide Software* is for developers and managers who develop products for the worldwide UNIX platform, especially for the Sun Solaris system.

- Chapter 1, “Winning in Global Markets,” briefly shows the market potential of internationalizing your products and defines the steps of internationalization and localization.
- Chapter 2, “Understanding Linguistic and Cultural Differences,” shows through examples how an item will appear in various cultures.
- Chapter 3, “Encoding Character Sets,” describes how to encode character sets in any language.
- Chapter 4, “Establishing Your Locale Environment,” looks at how a user selects a locale. It leads you through the steps of creating a specific locale for your product, including formats for time, date, money, and so on.
- Chapter 5, “Messaging for Program Translation,” explains how to prepare your product to handle localized messages. It discusses how to create and install your translated message catalogs.
- Chapter 6, “Displaying Localized Text,” discusses font, user interface, and printing issues.
- Chapter 7, “Handling Language Input,” discusses the various input methods for various languages.
- Chapter 8, “Working with CDE,” explains the CDE environment and your localization.
- Chapter 9, “Motif Programming,” discusses how to write applications under Motif and CDE.
- Chapter 10, “X11 Programming,” discusses internationalization with X11.
- Chapter 11, “Communicating Network Data,” discusses issues in sharing and distributing data across networks.
- Chapter 12, “Writing International Documentation,” includes guidelines for writing manuals and documentation to be translated.
- Chapter 13, “Product Localization,” discusses business issues.
- Chapter 14, “Standards Organizations,” is a summary of the international standards organizations.
- Chapter 15, “Internationalization Checklist,” has a checklist for internationalization.
- Appendix A, “Languages, Territories, and Locale Names,” lists the standard names for languages, locales, and so on.
- Appendix B, “Locale Summaries and Keyboard Layouts,” lists many locale-specific information and keyboard layouts.
- Appendix C, “OpenWindows and DevGuide,” explains how internationalization works with OpenWindows.

- Appendix D, “XView Programming,” discusses internationalization with XView.
- Appendix E, “OLIT Programming,” discusses internationalization with OPEN LOOK Intrinsic Toolkit (OLIT).
- Appendix F, “Example Program,” offers a complete source code for an internationalized Motif application.
- Appendix G, “Annotated Bibliography,” is a summary of additional suggested books.
- Appendix H, “Glossary,” is a list of key terms.

Contents of the Base Solaris Product

Summary of the Base Product

Solaris 7 includes partial locales, which provide the functionality needed for entering, displaying, and printing in local languages while using an English interface. It also includes the `en_US.UTF-8` locale, which also uses an English interface, and supports the Unicode UTF-8 character encoding standard.

The base English Solaris 7 product includes the Euro full locales, a number of partial European locales as well as the `en_US.UTF-8` locale.

The File System Safe Universal Transformation Format, or UTF-8, is an encoding defined by X/Open as a multi-byte representation of Unicode. UTF-8 is a variant of UNICODE. UTF-8 provides input and output support for all Solaris single-byte locales.

Partial locales can be split into two groups: the core set and the extended set. The core set is packaged in `SUNWploc` (operating system locale) and `SUNWplow` (window system locale). Since these packages are part of the end user cluster, they are installed automatically. The extended set of locales is packaged in `SUNWploc1` (operating system locale) and `SUNWplow1` (Window system locale). `SUNWpldte` has CDE support for the Eastern European locales.

`SUNWploc1` and `SUNWplow1` are available on the entire cluster only. `SUNWploc1` and `SUNWplow1` need to be added to your system before you can use the locales in the extended set.

Core Set of Locales

The core set of locales is installed automatically. The core sets are listed in Table 2-1.

TABLE 2-1 Core Set of Locales in SUNWploc and SUNWplow

Locale	Language	Country	Encoding
de	German	Germany	ISO-8859-1
en_AU	English	Australia	ISO-8859-1
en_CA	English	Canada	ISO-8859-1
en_UK changed to en_GB	English	Great Britain	ISO-8859-1
en_US	English	United States	ISO-8859-1
en_US.UTF-8	English	United States	UTF-8
es	Spanish	Spain	ISO-8859-1
es_AR	Spanish	Argentina	ISO-8859-1
es_BO	Spanish	Bolivia	ISO-8859-1
es_CL	Spanish	Chile	ISO-8859-1
es_CO	Spanish	Columbia	ISO-8859-1
es_CR	Spanish	Costa Rica	ISO-8859-1
es_EC	Spanish	Ecuador	ISO-8859-1
es_GT	Spanish	Guatemala	ISO-8859-1
es_MX	Spanish	Mexico	ISO-8859-1
es_NI	Spanish	Nicaragua	ISO-8859-1
es_PA	Spanish	Panama	ISO-8859-1

TABLE 2-1 Core Set of Locales in SUNWploc and SUNWploc (continued)

Locale	Language	Country	Encoding
es_PE	Spanish	Peru	ISO-8859-1
es_PY	Spanish	Paraguay	ISO-8859-1
es_SV	Spanish	El Salvador	ISO-8859-1
es_UY	Spanish	Uruguay	ISO-8859-1
es_VE	Spanish	Venezuela	ISO-8859-1
fr	French	France	ISO-8859-1
it	Italian	Italy	ISO-8859-1
sv	Swedish	Sweden	ISO-8859-1

New Locales

Solaris software already supports most of the Western European locales and, in this release, has focused on expanding its support for the Eastern European, Thai, and the Middle Eastern regions. New and changed user locales in the Solaris 7 operating environment are listed in Table 2-2

TABLE 2-2 New or Changed User Locales

Region	Locale Name	ISO Codeset	Comments
Albania	sq_AL	8859-2	
Bosnia	nr	8859-2	
Bulgaria	bg_BG	8859-5	
Croatia	hr_HR	8859-2	

TABLE 2-2 New or Changed User Locales *(continued)*

Region	Locale Name	ISO Codeset	Comments
Finland	su changed to fi	8859-15	Changed to comply with ISO standards
France	fr	UTF-8	
Germany	de	UTF-8	
Macedonia	mk_MK	8859-5	
Israel	he	8859-8	
Italy	it	UTF-8	
Norway (nynorsk)	no_NY	8859-1	
P.R. China	zh.GBK	GBK	GBK is a superset of GB2312
Romania	ro_RO	8859-2	
Russia	ru	KOI-8	The default codeset has been changed to KOI-8 from ISO 8859-5
Saudi Arabia	ar	8859-6	
Serbia	sr_SP	8859-5	
Slovakia	sk_SK	8859-2	
Slovenia	sl_SI	8859-2	
Spain	es	UTF-8	
Sweden	sv	UTF-8	
Thailand	th_TH	TIS 620-2533	Thai character codeset has been registered to ISO 8859-11

TABLE 2-2 New or Changed User Locales *(continued)*

Region	Locale Name	ISO Codeset	Comments
Great Britain	en_UK changed to en_GB	8859-15	Changed to comply with ISO standards
United States	en_US	UTF-8	

Solaris 7 software has added support for the Euro currency by adding six new user locales. These are included in Table 2-3. Note that local currency symbols are still available for backwards compatibility.

TABLE 2-3 New User Locales To Support the Euro Currency

Region	Locale Name	ISO Codeset
Austria	de_AT	8859-15
Belgium (French)	fr_BE	8859-15
Belgium (Dutch)	nl_BE	8859-15
Denmark	da	8859-15
England	en_EU	8859-15
Finland	su changed to fi	8859-15
France	fr	8859-15
Germany	de	8859-15
Ireland	en_IE	8859-15
Italy	it	8859-15
Netherlands	nl	8859-15

TABLE 2-3 New User Locales To Support the Euro Currency *(continued)*

Region	Locale Name	ISO Codeset
Portugal	pt	8859-15
Spain	es	8859-15
Sweden	sv	8859-15
Great Britain	en_GB	8859-15
Europe	en_EU	8859-15

Extended Set of Locales

The extended set of locales is not installed automatically. If you want to use locales listed in Table 2-4 you need to install them manually.

TABLE 2-4 Extended Set of Locales in SUNWploc1 and SUNWplow1

Locale	Language	Country	Encoding
cz	Czech	Czechoslovakia	ISO-8859-2
da	Danish	Denmark	ISO-8859-15
de_AT	German	Austria	ISO-8859-15
de_CH	German	Switzerland	ISO-8859-1
el	Greek	Greece	ISO-8859-7
en_IE	English	Ireland	ISO-8859-1
en_NZ	English	New Zealand	ISO-8859-1
et	Estonian	Estonia	ISO-8859-15
fr_BE	French	Belgium	ISO-8859-1
fr_CA	French	Canada	ISO-8859-1

TABLE 2-4 Extended Set of Locales in SUNWploc1 and SUNWplow1 (continued)

Locale	Language	Country	Encoding
fr_CH	French	Switzerland	ISO-8859-1
hu	Hungarian	Hungary	ISO-8859-2
lt	Lithuanian	Lithuania	ISO-8859-13
lv	Latvian	Latvia	ISO-8859-13
nl	Dutch	Netherlands	ISO-8859-1
nl_BE	Dutch	Belgium	ISO-8859-1
no	Norwegian	Norway	ISO-8859-1
pl	Polish	Poland	ISO-8859-2
pt	Portuguese	Portugal	ISO-8859-1
pt_BR	Portuguese	Brazil	ISO-8859-1
ru	Russian	Russia	ISO-8859-5
su	Finnish	Finland	ISO-8859-1
tr	Turkish	Turkey	ISO-8859-9

Unicode Locale: en_US.UTF-8

The `en_US.UTF-8` locale is a multiscrypt locale that can input and output text in multiple scripts, including single-byte and multi-byte scripts. This locale is part of the developer cluster. This is the first locale with this capability in the Solaris operating environment.

This locale uses UTF-8 (Universal Character Set Transformation Format for 8 bits) encoding, which was developed by the X/Open-Uniform Joint Internationalization Working Group (XoJIG). This standard has been adopted by the Unicode Consortium, the International Standards Organization, and the International Electrotechnical Commission as a part of Unicode 2.0 and ISO/IEC 10646-1.

en_US.UTF-8 supports computation for every code point value, which is defined in Unicode 2.0 and ISO/IEC 10646-1. In Solaris 7, language script support is not limited to pan-European locales, but also includes Asian scripts such as Korean, Traditional Chinese, Simplified Chinese, and Japanese. Input method support has been enabled for the following language scripts only. Due to limited font resources, Solaris 7 software includes only character glyphs from the following codesets:

- ISO 8859-1 (most Western European languages, such as English, French, Spanish, and German)
- ISO 8859-2 (most Central European languages, such as Czech, Polish, and Hungarian)
- ISO 8859-4 (Scandinavian and Baltic languages)
- ISO 8859-5 (Russian)
- ISO 8859-6 (Arabic)
- ISO 8859-7 (Greek)
- ISO 8859-8 (Hebrew)
- ISO 8859-9 (Turkish)
- ISO 8859-11 (Thai) or TIS 620.2533

User Locales in the Base Solaris Product

The Base Solaris 7 product includes the locale support listed in Table 2-5.

TABLE 2-5 User Locales Included in Solaris 7 Product

Country	Locale-Name	ISO codeset
Austria	de_AT (German Partial Locale)	8859-1
Estonia	et	8859-1
Czech	cz	8859-2
Hungary	hu	8859-2
Poland	pl	8859-2
Latvia	lv	8859-4

TABLE 2-5 User Locales Included in Solaris 7 Product (continued)

Country	Locale-Name	ISO codeset
Lithuania	lt	8859-13
Russia	ru	8859-5
Greece	el.sun_eu_greek	8859-7 (modified)
Turkey	tr	8859-9

These locales are supported through the `SUNWploc1` (for operating system support), `SUNWplow1` (for OpenWindows support), and `SUNWplde` (for locales support) packages, which are part of the entire cluster. The fonts for these locales have the format `SUNiXxf`.

- `iX` represents the ISO 8859 codeset.
- `xf` indicates whether the font is optional or required.

`SUNWilrf` contains the required font and `SUNWiof` contains the optional font for an ISO 8859-1 codeset locale. These packages are in different clusters; install the entire cluster or selectively add the appropriate packages. After the packages have been installed, users can login through `dtlogin` to either CDE or OpenWindows and use the characters associated with their locale.

Multiple Key Compose Sequences for Locales

The Solaris 7 operating environment supports “compose sequences” to create the diacritical marks used in writing the scripts covered in the following codesets:

- ISO 8859-2 (Latin2) Czech, Polish, and Hungarian
- ISO 8859-4 (Latin4) Latvian and Lithuanian
- ISO 8859-9 (Latin5) Turkish

These are the diacritic characters that can be created with the following keys and the Compose key.

- diaeresis = citation (“)(for example, Compose + A + “ = Ä)
- caron = ˇ (for example, Compose + E + ˇ = E caron)

- breve = u
- ogonek = a
- doubleacute = > greater
- degree symbol = O + 0 (o plus zero)
- currency symbol = 0 + x (zero plus x)

Keyboard Support in the Base Solaris 7 Product

The following locales have keyboard layouts for SPARC (X-server) and X86 (Xserver PLUS console):

- Czech
- Hungary
- Poland
- Latvia
- Lithuania
- Russia
- Greece
- Turkey

[X-server is CDE and OW, console is command line]

Changing Between Keyboards on SPARC

Support for changing layouts in the Solaris product is achieved only by using the dip-switch settings under the keyboard. The keyboard layout is determined by the dip switches. A list of keyboard layouts and corresponding defined dip-switch settings is at `/usr/openwin/share/etc/keytables/keytable.map`.

The following table Table 2-6 is for a type 4 keyboard (.1=switch up 0=switch down).

TABLE 2-6 Layouts for Type 4 Keyboards

Dip Switch in Hex	Keyboard	Setting in Binary
51	Hungary5.kt	110011
52	Poland5.kt	110100
53	Czech5.k	110101
54	Russia5.kt	110110
55	Latvia5.k	110111
56	Turkey5.kt	111000
57	Greece5.kt	111001
58	Lithuania5.kt	111011

Changing the layout from U.S./UK to Czech is done by changing the dip-switch settings to the setting defined in the file (the file defines the switches in hex. This needs to be converted into binary as it was shown in Table 2-6) and then re-booting.

Russian and Greek keyboard support can be toggled on and off using the SPARC Compose key (Ctrl+Shift+F1 on x86).

Changing Between Keyboards on x86

On x86, a keyboard is selected during the `kdmconfig` part of install. To change this at any time after installation, use `kdmconfig`:

1. Exit CDE/OW to the command line.
2. Type `kdmconfig -u` (*kdmconfig unconfigure*).
3. Type `kdmconfig` to run the program.
4. Follow instructions to get a keyboard layout.

There are no 'utilities' for either SPARC or x86 (apart from standard UNIX tools such as `xmodmap`, `pcmapkeys`) bundled into Solaris 7 for switching keyboards.

Codesets for x86

The default codeset on the Solaris system for x86 is ISO-8859-1. The IBM DOS 437 codeset is provided as an option in text mode. That is, if you choose to download IBM DOS 437 codeset by typing:

```
loadfont -c 437
pcmapkeys -f /usr/share/lib/keyboards/437/en_US
```

there is no support for nonstandard U.S. date, time, currency, numbers, units, and collation. There will be no support for non-English message and text presentation, and no multibyte character support. Therefore, non-Microsoft Windows users should use the IBM DOS 437 codeset only in the default C locale.

- You must be in the text mode to download the IBM codeset, not the graphics mode.
- If you are not using the standard U.S. PC keyboard, replace `en_US` with the keyboard map related to your keyboard.
- To download the default codeset in text mode, type:

```
loadfont -c 8859
pcmapkeys -f /usr/share/lib/keyboards/8859/en_US
```

- See the *man Pages(1): User Commands* `loadfont(1)` and `pcmapkeys(1)` man pages.

Locales in the Base Installation

The installation window in the base Solaris 7 product offers several English language locales. To use 8-bit characters, install one of the `en_XX` options, as shown in Table 2-7. The locale used in the installation becomes the default system locale.

TABLE 2-7 Locales Offered at Installation

Locale Name	Language/Territory	Codeset
<code>c</code>	U.S. English	7-bit
<code>en_AU</code>	Australian English	8-bit
<code>en_CA</code>	Canadian English	8-bit

TABLE 2-7 Locales Offered at Installation *(continued)*

Locale Name	Language/Territory	Codeset
en_UK	UK English	8-bit
en_US	U.S. English	8-bit

Using JumpStart

To enable JumpStart™ for the 8-bit locales, add the line `localexx` (substituting the appropriate 8-bit locale for `xx`, for example, `en_US`) to the JumpStart profile file. (For complete instructions, see Chapter 4 of *Automating Solaris Installation*, available from SunSoft Press.) Current JumpStart users should set the default locale to bypass the language prompt during installation.

Contents of the Localized Solaris 7 Products

The European Localized Solaris 7 Product

European Solaris is available in three localized versions: French, German, and European. All three versions of Solaris share the same software media, which includes a fully localized CDE environment, error messages, and on-line documentation in six languages—French, German, Spanish, Swedish, Italian, and English. The difference is in the printed documentation. The French and German Solaris products include localized printed documentation, while the printed documentation for the European version is in English only.

Table 3-1 shows a list of locales in the European product. This includes both full and partial locales.

TABLE 3-1 European 7 Locales

Locale Name	Language/Territory
c	POSIX English (7-bit) ASCII C
cz	Czech Republic
da	Denmark

TABLE 3-1 European 7 Locales *(continued)*

Locale Name	Language/Territory
de	Germany
de_AT	Austria
de_CH	Switzerland
de.ISO8859-15	Germany
el	Greece
en_AU	Australia
en_CA	Canada
en_IE	Ireland
en_NZ	New Zealand
en_UK	Great Britain
en_US	U.S.
es	Spain
es_AR	Argentina
es_BO	Bolivia
es_CL	Chile
es_CO	Colombia
es_CR	Costa Rica
es_EC	Ecuador
es_GT	Guatemala

TABLE 3-1 European 7 Locales *(continued)*

Locale Name	Language/Territory
es_MX	Mexico
es_NI	Nicaragua
es_PA	Panama
es_PE	Peru
es_PY	Paraguay
es_SV	El Salvador
es_UY	Uruguay
es_VE	Venezuela
et	Estonia
fr	France
fr_BE	Belgium (French)
fr_CA	Canada (French)
fr_CH	Switzerland (French)
fr.ISO8859-15	France
fr.UTF-8	France
hu	Hungary
it.ISO8859-15	Italy
it.UTF-8	Italy
it.ISO8859-15	Italy

TABLE 3-1 European 7 Locales *(continued)*

Locale Name	Language/Territory
lt.ISO8859-13	Lithuania
lv.ISO8859-13	Latvia
nl	Netherlands
nl_BE	Netherlands/Belgium
no	Norway
pl	Poland
pt_BR	Portuguese Brazil
ru	Russia
it.ISO8859-15	Italy
es.ISO8859-15	Spain
sv.ISO8859-15	Sweden
en_EU.ISO8859-15	Europe
en_GB.ISO8895-15	Britain
fr_BE.ISO8895-15	Belgium
nl.ISO8895-15	Netherlands
nl_BE.ISO8895-15	Belgium
pt.ISO8895-15	Portugal
de.-AT.ISO8895-15	Austria
en_IE.ISO8859-15	Ireland

TABLE 3-1 European 7 Locales *(continued)*

Locale Name	Language/Territory
da.ISO8859-15	Denmark
fi.ISO8859-15	Finland
el_EURO	Greece
sun_eu_greek	Greece
de.UTF-8	Germany
de.ISO8859-15	Germany
fr.UTF-8	France
it.UTF-8	Italy
es.UTF-8	Spain
es.ISO8859-15	Spain
sv.UTF-8	Sweden
sv.ISO8859-15	Sweden
en_UTF.8	Europe
en_ISO8859-15	Europe

All of these locales are also present in the base Solaris 7 release.

As mentioned, the locales include partial locales. These are based on core locales for the main language. For example, the `fr_CA` (French Canadian) is based on the `fr` (French) locale. These partial locales utilize the messages that are delivered into its parent locale (French for `fr_CA`). If a locale hasn't been fully localized, then it may contain only English messages.

A number of Eastern European locales have also been added into the Solaris 7 product, which may be based on other ISO standards. Previously Sun locales were

based on ISO-8859-1. The Eastern European locales are based on other ISO standards, as shown in Table 3-2.

Locales that are not listed are still based on ISO-8859-1.

TABLE 3-2 Eastern European Locales in the Solaris 7 Product

Locale Name	Language/Territory	ISO
de_AT	German (Austrian)	8859-1
et	Estonian	8859-15
cz	Czech	8859-2
hu	Hungarian	8859-2
pl	Polish	8859-2
lv	Latvian	8859-13
lt	Lithuanian	8859-13
ru	Russian	8859-5
el	Greek	8859-7
tr	Turkish	8859-9
sq_AL	Albanian	8859-2
sk_SK	Slovakian	8859-2
sl_SL	Slovenian	8859-2
hr_HR	Croatian	8859-2
nr	Bosnian	8859-2
ro_RO	Romanian	8859-2
sr_SP	Serbian	8859-5

TABLE 3-2 Eastern European Locales in the Solaris 7 Product *(continued)*

Locale Name	Language/Territory	ISO
bg_BG	Bulgarian	8859-5
mk_MK	Macedonian	8859-5
ru.KOI8-R	Russian	KOI8-R
ar	Arabic	8859-6
he	Hebrew	8859-8
th_TH	Thai	8859-11 (TIS 620.2533)

All of the locales support character input and output. There is also `iconv` support for many of the major codesets. (For more on `iconv`, see `iconv(1)`) The `iconv` modules are available on the end-user cluster of the Euro product. See Table 3-3 for details.

TABLE 3-3 `iconv` Support

Code	Symbol	Target Code	Symbol	Comment
ISO 8859-2	iso2	MS 1250	win2	Windows Latin 2
ISO 8859-2	iso2	MS 852	dos2	MS-DOS Latin 2
ISO 8859-2	iso2	Mazovia	maz	Mazovia
ISO 8859-2	iso2	DHN	dhn	Dom Handlowy Nauki
MS 1250	win2	ISO 8859-2	iso2	ISO Latin 2
MS 1250	win2	MS 852	dos2	MS-DOS Latin 2
MS 1250	win2	Mazovia	maz	Mazovia
MS 1250	win2	DHN	dhn	Dom Handlowy Naduki

TABLE 3-3 `iconv` Support (continued)

Code	Symbol	Target Code	Symbol	Comment
MS 852	dos2	ISO 8859-2	iso2	ISO Latin 2
MS 852	dos2	MS 1250	win2	Windows Latin 2
MS 852	dos2	Mazovia	maz	Mazovia
MS 852	dos2	DHN	dhn	Dom Handlowy Nauki
Mazovia	maz	ISO 8859-2	iso2	ISO Latin 2
Mazovia	maz	MS 1250	win2	Windows Latin 2
Mazovia	maz	MS 852	dos2	MS-DOS Latin 2
Mazovia	maz	DHN	dhn	Dom Handlowy Nauki
DHN	dhn	ISO 8859-2	iso2	ISO Latin 2
DHN	dhn	MS 1250	win2	Windows Latin 2
DHN	dhn	MS 852	dos2	MS-DOS latin 2
DHN	dhn	Mazovia	maz	Mazovia
ISO 8859-5	iso5	KOI8-R	koi8	KOI8-R
ISO 8859-5	iso5	PC Cyrillic	alt	Alternative PC Cyrillic
ISO 8859-5	iso5	MS 1251	win5	Windows Cyrillic
ISO 8859-5	iso5	Mac Cyrillic	mac	Macintosh Cyrillic
KOI8-R	koi8	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
KOI8-R	koi8	PC Cyrillic	alt	Alternative PC Cyrillic
KOI8-R	koi8	MS 1251	win5	Windows Cyrillic

TABLE 3-3 iconv Support *(continued)*

Code	Symbol	Target Code	Symbol	Comment
KOI8-R	koi8	Mac Cyrillic	mac	Macintosh Cyrillic
PC Cyrillic	alt	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
PC Cyrillic	alt	KOI8-R	koi8	KOI8-R
PC Cyrillic	alt	MS 1251	win5	Windows Cyrillic
PC Cyrillic	alt	Mac Cyrillic	mac	Macintosh Cyrillic
MS 1251	win5	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
MS 1251	win5	KOI8-R	koi8	KOI8-R
MS 1251	win5	PC Cyrillic	alt	Alternative PC Cyrillic
MS 1251	win5	Mac Cyrillic	mac	Macintosh Cyrillic
Mac Cyrillic	mac	ISO 8859-5	iso5	ISO 8859-5 Cyrillic
Mac Cyrillic	mac	KOI8-R	koi8	KOI8-R
Mac Cyrillic	mac	PC Cyrillic	alt	Alternative PC Cyrillic
Mac Cyrillic	mac	MS 1251	win5	Windows Cyrillic

Table 3-4 contains a list of the Solaris 7 environment locales and their corresponding codeset names.

TABLE 3-4 New Locales and Corresponding Codeset Names

Locale	nl_langinfo (CODESET)	ICONV name	Product
ar	ISO8859-6	ISO8859-6	Base/Euro
bg_BG	ISO8859-5	ISO8859-5	Base/Euro

TABLE 3-4 New Locales and Corresponding Codeset Names *(continued)*

Locale	nl_langinfo (CODESET)	ICONV name	Product
C	646	646	Base/Euro
cz	ISO8859-2	ISO8859-2	Base/Euro
da	ISO8859-1	ISO8859-1	Base/Euro
da.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
de	ISO8859-1	ISO8859-1	Base/Euro
de.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
de_UTF-8	UTF-8	UTF-8	Base/Euro
de_AT	ISO8859-1	ISO8859-1	Base/Euro
de_AT.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
de_CH	ISO8859-1	ISO8859-1	Base/Euro
el	ISO8859-7	ISO8859-7	Base/Euro
el.sun_eu_greek	ISO8859-15	ISO8859-15	Base/Euro
en_AU	ISO8859-1	ISO8859-1	Base/Euro
en_CA	ISO8859-1	ISO8859-1	Base/Euro
en_EU.ISO8859-15	ISO8859-15	ISO8859-1	Base/Euro
en_EU.UTF-8	UTF-8	UTF-8	Base/Euro
en_GB	ISO8859-1	ISO8859-1	Base/Euro
en_GB.ISO8859-15	ISO8859-15	ISO8859-1	Base/Euro
en_IE	ISO8859-1	ISO8859-1	Base/Euro
en_IE.ISO8859-15	ISO8859-15	ISO8859-1	Base/Euro
en_NZ	ISO8859-1	ISO8859-1	Base/Euro
en_US	ISO8859-1	ISO8859-1	Base/Euro
en_US.UTF-8	UTF-8	UTF-8	Base/Euro
es	ISO8859-1	ISO8859-1	Base/Euro
es.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
es_AR	ISO8859-1	ISO8859-1	Base/Euro
es_BO	ISO8859-1	ISO8859-1	Base/Euro
es_CL	ISO8859-1	ISO8859-1	Base/Euro
es_CO	ISO8859-1	ISO8859-1	Base/Euro

TABLE 3-4 New Locales and Corresponding Codeset Names *(continued)*

Locale	nl_langinfo (CODESET)	ICONV name	Product
es_CR	ISO8859-1	ISO8859-1	Base/Euro
es_EC	ISO8859-1	ISO8859-1	Base/Euro
es_GT	ISO8859-1	ISO8859-1	Base/Euro
es_MX	ISO8859-1	ISO8859-1	Base/Euro
es-NI	ISO8859-1	ISO8859-1	Base/Euro
es_PA	ISO8859-1	ISO8859-1	Base/Euro
es_PE	ISO8859-1	ISO8859-1	Base/Euro
es_PY	ISO8859-1	ISO8859-1	Base/Euro
es_SV	ISO8859-1	ISO8859-1	Base/Euro
es.UTF-8	UTF-8	UTF-8	Base/Euro
es_UY	ISO8859-1	ISO8859-1	Base/Euro
et_VE	ISO8859-1	ISO8859-1	Base/Euro
et	ISO8859-1	ISO8859-1	Base/Euro
fi	ISO8859-1	ISO8859-1	Base/Euro
fi.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
fr	ISO8859-1	ISO8859-1	Base/Euro
fr.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
fr.UTF-8	UTF-8	UTF-8	Base/Euro
fr_BE	ISO8859-1	ISO8859-1	Base/Euro
fr_BE.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
fr_CA	ISO8859-1	ISO8859-1	Base/Euro
fr_CH	ISO8859-1	ISO8859-1	Base/Euro
he	ISO8859-8	ISO8859-8	Base/Euro
he_IL	ISO8859-8	ISO8859-8	Base/Euro
hr_HR	ISO8859-2	ISO8859-2	Base/Euro
hu	ISO8859-2	ISO8859-2	Base/Euro
it	ISO8859-1	ISO8859-1	Base/Euro
it.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
it.UTF-8	UTF-8	UTF-8	Base/Euro

TABLE 3-4 New Locales and Corresponding Codeset Names *(continued)*

Locale	nl_langinfo (CODESET)	ICONV name	Product
ja	eucJP	eucJP	Japanese
ja_JP.PCK	PCK	PCK	Japanese
ja_JP.UTF-8	UTF-8	UTF-8	Japanese
ko	5601	ko_KR-euc	Korean
ko.UTF-8	UTF-8	UTF-8	Korean
lt	ISO8859-4	ISO8859-4	Base/Euro
lv	ISO8859-4	ISO8859-4	Base/Euro
mk_MK	ISO8859-5	ISO8859-5	Base/Euro
nl	ISO8859-1	ISO8859-1	Base/Euro
nl.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
nl_BE	ISO8859-1	ISO8859-1	Base/Euro
nl_BE.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
no	ISO8859-1	ISO8859-1	Base/Euro
no_NY	ISO8859-1	ISO8859-1	Base/Euro
nr	ISO8859-2	ISO8859-2	Base/Euro
pl	ISO8859-2	ISO8859-2	Base/Euro
POSIX	646	646	Base/Euro
pt	ISO8859-1	ISO8859-1	Base/Euro
pt.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro
pt_BR	ISO8859-1	ISO8859-1	Base/Euro
ro_RO	ISO8859-2	ISO8859-2	Base/Euro
ru	ISO8859-5	ISO8859-5	Base/Euro
ru.KOI8-R	KOI8-R	KOI8-R	Base/Euro
sk_SK	ISO8859-2	ISO8859-2	Base/Euro
sl_SI	ISO8859-2	ISO8859-2	Base/Euro
sq_AL	ISO8859-2	ISO8859-2	Base/Euro
sr_SP	ISO8859-5	ISO8859-5	Base/Euro
sv	ISO8859-1	ISO8859-1	Base/Euro
sv.ISO8859-15	ISO8859-15	ISO8859-15	Base/Euro

TABLE 3-4 New Locales and Corresponding Codeset Names *(continued)*

Locale	nl_langinfo (CODESET)	ICONV name	Product
sv.UTF-8	UTF-8	UTF-8	Base/Euro
th_TH	TIS620.2533	TIS620.2533	Base/Euro
tr	ISO8859-9	ISO8859-9	Base/Euro
zh	gb2312	gb2312	Simplified Chinese
zh.GBK	GBK	zh_CN.gbk	Simplified Chinese
zh_TW	cns11643	zh_TW-euc	Traditional Chinese
zh_TW.BIG5	BIG5	zh_TW_Big5	Traditional Chinese

Note - Locale naming conventions are as follows:

language[_territory][.codeset] where language is from ISO639 and territory is from ISO3166.

All locales with Base/Euro in the Product column are also available as Japanese, Korean, Simplified Chinese, and Traditional Chinese products.

All Solaris product locales preserve the Portable Character Set characters with US-ASCII code values.

Note - 5601 signifies the Korean EUC codeset containing KS C 5636 and KS C 5601-1987.

646 signifies ISO/IEC 646, which is US-ASCII.

eucJP signifies the Japanese EUC codeset. It contains JIS X0201-1976, JIS X0208-1983, and JIS X0212-1990.

gb2312 signifies Simplified Chinese EUC codeset, which contains GV 1988-80 and GB 2312-80.

PCK is also known as Shift JIS (SJIS).

UTF-8 is the UTF-8 of ISO/IEC 10646-1 containing various approved amendments and UNICODE 2.1

GBK signifies GB extensions. This includes all GB 2312-80 characters and all Unified Han characters of ISO/IEC 10646-1, as well as Japanese Hiragana and Katagana characters. It also includes many characters of Chinese, Japanese, and Korean character sets and of ISO/IEC 10646-1.

Font Formats

There are many different font formats. The extension lets you determine the font type.

- PostScript Type 1 Fonts , which are also known as Adobe Type Manager (ATM) fonts, Type 1, and outline fonts, contain information in outline form that allows a PostScript printer or ATM to generate fonts of any size. Most of these fonts also contain hints that allow fonts to be rendered more readable at a low resolution or a small type size.
- Bitmap Fonts contain a picture of the font at a specific size that has been optimized to look good at that specific size. If the font is scaled larger or smaller, the quality may degrade. On the other hand, bitmap fonts display quickly.

Location of Fonts on the System

Fonts are located at:

```
/usr/openwin/lib/locale/iso_8859_x/X11/fonts/X11/Type1/afm
```

or

```
/usr/openwin/lib/locale/iso_8859_x/X11/fonts/X11/75dpi
```

Adding and Removing Font Packages

To manually add font packages to the system:

1. Always add the required font packages before the optional font packages.
2. When you are removing font packages from the system, remove the optional font packages first.

You must follow this procedure to add or remove fonts. The class action scripts in the font packages depend on this for proper function. The optional font packages contain scripts that concatenate information onto the required font packages that are already resident on the system. If the required font packages are not there, problems may occur.

Summary of Asian Locales

Table 3-6 shows the Asian locales supported by these Asian products.

TABLE 3-5 Summary of Asian Locales

CD Set	Locale Name	Description	Supported Character Set
Korean	ko UTF-8	Korean (UTF-8 locale)	KS C 5601-1992 KS C 5700-1995
Simplified Chinese	zh GBK	Simplified Chinese (EUC) Simplified Chinese (GBK)	GB 2312-1980 GBK
Traditional Chinese	zh_TW zh_TW.BIG5	Traditional Chinese (EUC) Traditional Chinese (BIG5)	CNS 11643 1992 BIG5
Japanese	ja ja_JP.PCK ja_JP.UTF-8	Japanese EUC Japanese PCK ¹ Japanese UTF-8	JIS x 0201-1976 JIS x 0208-1990 JIS x 0212-1990 VDC ² UDC ³

1. ja_JP.PCK doesn't support JIS x 0212-1990
2. VDC: Vendor Defined Character. VDCs occupy unused (reserved) code points of JIS X 0208-1990 or JIS X 0212-1990
3. UDC: User Defined Character. UDCs occupy unused (reserved) code points of JIS X 0208-1990 or JIS X 0212-1990 (also unused for VDCs.)

Korean in the Solaris 7 Product

In December 1995, the Korean government announced a standard Korean codeset, KSC-5700, which is based on ISO-10646-1/Unicode 2.0. The standard codeset replaces KSC 5601, which was based on ISO-2022.

The ISO-10646 character set uses 2 (UCS-2; Universal Character Set two-byte form) or 4 (UCS-4) bytes to represent each character.

The ISO-10646 character set cannot be used directly on IBM-PC-based operating systems. For example, the kernel and many other modules of the Solaris operating environment interpret certain byte values as control instructions, such as a null character (0x00) in any string. The ISO-10646 character set can be encoded with any bit combinations in the first or subsequent bytes. The ISO-10646 characters cannot be

freely transmitted through the Solaris system with these limitations. In order to establish a migration path, the ISO-10646 character set defines the UCS Transformation Format (UTF), which recodes the ISO-10646 characters without using C0 controls (0x00..0x1F), C1 controls (0x80..0x9F), space (0x20), and DEL (0x7F).

The `ko.UTF-8` is a Solaris locale to support KSC-5700, the Korean standard codeset. It supports all characters in the previous KSC 5601 and all 11,172 Korean characters. Korean UTF-8 supports the Korean language-related ISO-10646 characters and fonts. Because ISO-10646 covers all characters in the world, all of the various input methods and fonts are supplied so that you may input and output any character in any language. Before Universal UTF/UCS becomes available, Korean UTF-8 supports the ISO-10646 code subset that is related to Korean characters as well as all other characters in the previous Korean standard codeset, and Extended ASCII.

Table 3-6 lists the Korean codesets.

TABLE 3-6 Codeset Conversions Supported for Korean `ko`, `ko.UTF-8`

Code	Symbol	TargetCode	Symbol
UTF-8	ko_KR-UTF-8	Wansung	ko_KR-euc
UTF-8	ko_KR-UTF-8	Johap	ko_KR-johap92
UTF-8	ko_KR-UTF-8	Packed	ko_KR-johap
UTF-8	ko_KR-UTF-8	ISO-2022-KR	ko_KR-iso2022-7
Wansung	ko_KR-euc	UTF-8	ko_KR-UTF-8
Johap	ko_KR-johap92	UTF-8	ko_KR-UTF-8
Packed	ko_KR-johap	UTF-8	ko_KR-UTF-8
ISO-2022-KR	ko_KR-iso2022-7	UTF-8	ko_KR-UTF-8
Wansung	ko_KR-euc	Johap	ko_KR-johap92
Wansung	ko_KR-euc	Packed	ko_KR-johap
Wansung	ko_KR-euc	N-Byte	ko_KR-nbyte
Wansung	ko_KR-euc	ISO-2022-KR	ko_KR-iso2022-7

TABLE 3-6 Codeset Conversions Supported for Korean ko, ko.UTF-8 (continued)

Code	Symbol	TargetCode	Symbol
Johap	ko_KR-johap92	Wansung	ko_KR-euc
Packed	ko_KR-johap	Wansung	ko_KR-euc
N-Byte	ko_KR-nbyte	Wansung	ko_KR-euc
ISO-2022-KR	ko_KR-iso2022-7	Wansung	ko_KR-euc

Chinese: Simplified and Traditional

Simplified Chinese in the Solaris 7 environment provides two locales: zh and zh.GBK. In the zh locale, the EUC scheme is used to encode GB2312-80. The zh.GBK locale supports the GBK codeset, which is a superset of GB2312-80.

Simplified Chinese is used mostly in the People's Republic of China (PRC) and in Singapore.

The following input methods are supported for the zh locale

- New QuanPin
- New ShuangPin
- Quanpy
- Location
- PinYin
- Stroke
- Golden
- Intelligent Pinyin
- Simplified Chinese Symbol

The following input methods are supported for the zh.GBK locale

- New QuanPin
- New ShuangPin
- Quanpy

- GBK Code
- Japanese
- Hanja
- Zhuyin
- Unicode

Table 3-7 shows the TrueType Fonts for the zh Locale

TABLE 3-7 Solaris 7 TrueType Fonts for the zh Locale

Full Family Name	Subfamily	Format	Vendor	Encoding
Fangsong	R	TrueType	Hanyi	GB2312.1980
Hei	R	TrueType	Monotype	GB2312.1980
Kai	R	TrueType	Monotype	GB2312.1980
Song	R	TrueType	Monotype	GB2312.1980

Table 3-8 shows the Bitmap Fonts for the zh Locale

TABLE 3-8 Solaris 7 Bitmap Fonts for the zh Locale

Full Family Name	Subfamily	Format	Encoding
Song	B	PCF (14,16)	GB2312.1980
Song	R	PCF (12,14,16,20,24)	GB2312.1980

Table 3-9 shows the TrueType Fonts for the zh.GBK Locale

TABLE 3-9 TrueType Fonts for the zh.GBK Locale

Full Family NameS	Subfamily	Format	Vendor	Encoding
Fansong	R	TrueType	Zhongyi	GBK
Hei	R	TrueType	Zhongyi	GBK
Kai	R	TrueType	Zhongyi	GBK
Song	R	TrueType	Zhongyi	GBK

Table 3–10 shows the Bitmap Fonts for the zh.GBK Locale

TABLE 3–10 Bitmap Fonts for the zh.GBK Locale

Full Family Name	Subfamily	Format	Encoding
Song	R	PCF (12,14,16,20,24)	GBK

Table 3–11 shows the supported codeset conversions for Simplified Chinese.

TABLE 3–11 Codeset Conversions for Simplified Chinese

Code	Symbol	TargetCode	Symbol
GB2312-80	zh_CN.euc	ISO 2022-7	zh_CN.iso2022-7
ISO 2022-7	zh_CN.iso2022-7	GB2312-80	zh_CN.euc
GB2312-80	zh_CN.euc	ISO 2022-CN	zh_CN.iso2022-CN
ISO-2022-CN	zh_CN.iso2022-CN	GB2312-80	zh_CN.euc
UTF-8	UTF-8	GB2312-80	zh_CN.euc
GB2312-80	zh_CN.euc	UTF-8	UTF-8
zh.GBK	zh_CN.gbk	ISO2022-CN	zh_CN.iso2022-CN
ISO2022-CN	zh_CN.iso2022-CN	zh.GBK	zh_CN.gbk
zh.GBK	zh_CN.gbk	Big-5	zh_TW-Big5
Big-5	zh_TW-Big5	zh.GBK	zh_CN.gbk
GB2312-80	zh_CN.euc	Big-5	zh_TW-Big5
Big-5	zh_TW-Big5	GB2312-80	zh_CN.euc
UTF-8	UTF-8	zh.GBK	zh_CN.gbk

TABLE 3-11 Codeset Conversions for Simplified Chinese *(continued)*

Code	Symbol	TargetCode	Symbol
zh.GBK	zh_CN.gbk	UTF-8	UTF-8
UTF-8	UTF-8	ISO2022-CN	zh_CN.iso2022-CN
ISO2022-CN	zh_CN.iso2022-CN	UTF-8	UTF-8

Traditional Chinese in the Solaris 7 product provides two locales: `zh_TW` and `zh_TW.BIG5`. In the `zh_TW` locale, the EUC scheme is used to encode CNS 11643.1992 codeset. The `zh_TW.BIG5` locale supports the Big-5 codeset.

Traditional Chinese is used mostly in Taiwan and Hong Kong.

Traditional Chinese supports the following input methods:

- Chuyin
- I-Tien
- Telecode
- TsangChieh
- CheinI
- NeiMa
- ChuangHsing
- Array
- BoShiaMy
- DaYi

Table 3-12 shows Traditional Chinese Truetype Fonts for the `zh_TW` Locales

TABLE 3-12 Traditional Chinese Truetype Fonts for the `zh_TW` Locales

Full Family Name	Subfamily	Format	Vendor	Encoding
Hei	R	Truetype	Hanyi	CNS11643.1992
Kai	R	Truetype	Hanyi	CNS11643.1992
Ming	R	Truetype	Hanyi	CNS11643.1992

Table 3-13 shows the Traditional Chinese BitMap Fonts for the `zh_TW` Locales

TABLE 3-13 Traditional Chinese BitMap Fonts for the zh_TW Locales

Full Family Name	Subfamily	Format	Encoding
Ming	R	PCF (12,14,16,20,24)	CNS11643.1992

Table 3-14 shows the Traditional Chinese TrueType Fonts for the zh_TW.Big5 Locales

TABLE 3-14 Traditional Chinese TrueType Fonts for the zh_TW.Big5 Locales

Full Family Name	Subfamily	Format	Vendor	Encoding
Hei	R	TrueType	Hanyi	Big5
Kai	R	TrueType	Hanyi	Big5
Ming	R	TrueType	Hanyi	Big5

Table 3-15 shows the Traditional Chinese BitMap Fonts for the zh_TW.Big5 Locales

TABLE 3-15 Traditional Chinese BitMap Fonts for the zh_TW.Big5 Locales

Full Family Name	Subfamily	Format	Encoding
Ming	R	PCF (12,14,16,20,24)	Big5

Table 3-16 shows the supported codeset conversions for Traditional Chinese.

TABLE 3-16 Codeset Conversions for Traditional Chinese

Code	Symbol	TargetCode	Symbol
CNS 11643	zh_TW-euc	Big-5	zh_TW-Big5
CNS 11643	zh_TW-euc	ISO 2022-7	zh_TW-iso2022-7
Big-5	zh_TW-Big5	CNS 11643	zh_TW-euc
Big-5	zh_TW-Big5	ISO 2022-7	zh_TW-iso2022-7

TABLE 3-16 Codeset Conversions for Traditional Chinese *(continued)*

Code	Symbol	TargetCode	Symbol
ISO 2022-7	zh_TW-iso2022-7	CNS 11643	zh_TW-euc
ISO 2022-7	zh_TW-iso2022-7	Big-5	zh_TW-Big5
CNS 11643	zh_TW-eu	ISO 2022-CN-EXT	zh_TW-iso2022-CN-EXT
ISO 2022-CN-EXT	zh_TW-iso2022-CN-EXT	CNS 11643	zh_TW-euc
Big-5	zh_TW-Big5	ISO 2022-CN	zh_TW-iso2022-CN
ISO 2022-CN	zh_TW-iso2022-CN	Big-5	zh_TW-Big5
UTF-8	UTF-8	CNS 11643	zh_TW-euc
CNS 11643	zh_TW-euc	UTF-8	UTF-8
UTF-8	UTF-8	Big-5	zh_TW-Big5
Big-5	zh_TW-Big5	UTF-8	UTF-8
UTF-8	UTF-8	ISO 2022-7	zh_TW-iso2022-7
ISO 2022-7	zh_TW-iso2022-7	UTF-8	UTF-8
ISO 2022-CN-EXT	zh_TW-iso2022-CN-EX	Big-5	zh_TW-Big5
Big-5	zh_TW-Big5	ISO 2022-CN-EXT	zh_TW-iso2022-CN-EXT

Japanese Input Systems

Three Japanese input systems are bundled in Japanese Solaris 7. They can be used in the `ja`, `ja_JP.PCK` and `ja_JP.UTF-8` locales. However, some maintenance utilities do not support the PCK codeset.

The Japanese Input System is shown below in Table 3-17.

TABLE 3-17 Japanese Input Systems

Name	Description
<code>wnn6</code>	<code>wnn6</code> consists of the Kana-Kanji conversion server (<code>jserver</code>), interface module for <code>htt</code> (X Input Method Server) called <code>xjsi.so</code> , utilities, and dictionaries. <code>wnn6</code> is the default Japanese input system. <code>wnn6</code> supports JIS X 0201-1976, JIS X 0208-1990 and JIS X0212-1990 character sets.
<code>ATOK8</code>	<code>ATOK8</code> consists of <code>atok8</code> X Input Method Server, utilities, and dictionaries. <code>ATOK8</code> is a popular Japanese input system facility in the Japanese PC market. <code>ATOK7</code> was released with Solaris 2.1 until 2.5.1 has been replaced by <code>ATOK8</code> . <code>ATOK8</code> supports JIS X 0201-1976 and JIS X 0208-1990 character sets.
<code>cs00</code>	<code>cs00</code> consists of the Kana-Kanji conversion server (<code>cs00</code>), interface module for <code>htt</code> (X Input Method Server) called <code>xci.so</code> , utilities, and dictionaries. <code>cs00</code> has been bundled with Japanese Solaris since Solaris 2.1 <code>cs00</code> supports JIS X 0201-1976, JIS X 0208-1990 and JIS X 0212-1990 character sets.

Japanese TrueType Fonts are show below in Table 3-18.

TABLE 3-18 Japanese TrueType Fonts

Full Family Name	Subfamily	Format	Vendor	Encoding
<code>hg gothic b</code>	R	TrueType	RICOH	JISX0208.1983, JISX0201.1976
<code>hg mincho l</code>	R	TrueType	RICOH	JISX0208.1983, JISX0201.1976
<code>heiseimin</code>	R	TrueType	RICOH	JISX0212.1990

Japanese Bitmap Fonts are shown in Table 3-19 below.

TABLE 3-19 Japanese Bitmap Fonts

Full Family Name	Subfamily	Format	Vendor	Encoding
gothic	R, B	PCF(12,14,16,20,24)		JISX0208.1983, JISX0201.1976
minchou	R	PCF(12,14,16,20,24)		JISX0208.1983, JISX0201.1976
hg gothic b	R	PCF(12,14,16,18,20,24)	RICOH	JISX0208.1983, JISX0201.1976
hg mincho l	R	PCF(12,14,16,18,20,2)	RICOH	JISX0208.1983, JISX0201.1976
heiseimin	R	PCF(12,14,16,18,20,24)	RICOH	JISX0212.1990

Japanese Locales

Japanese Solaris 7 supports three locales. The `ja` locale is based on Japanese EUC. The `ja_JP.PCK` locale is based on PC-Kanji code (Shift JIS) and the `ja_JP.UTF-8` locale is based on UTF-8.

Japanese Messages and man Pages

Some messages and manual pages have been translated into Japanese in Japanese Solaris 7.

Japanese Character Code Converter for `iconv`

The following table shows supported conversion with `iconv(1)` and `iconv(3)`. See the `iconv_ja(5)` man page for details.

Table 3-20 shows `iconv` Conversion Support.

TABLE 3-20 iconv Conversion Support

Source Code	Target Code
eucJP	PCK
eucJP	JIS7
eucJP	SJIS
eucJP	UTF-8
eucJP	jis
eucJP	ibmj
SJIS	eucJP
SJIS	ISO-2022-JP
SJIS	UTF-8
SJIS	jis
SJIS	ibmj
PCK	eucJP
PCK	UTF-8
PCK	ISO-2022-JP
PCK	jis
PCK	ibmj
ISO-2022-JP	eucJP
ISO-2022-JP	PCK
ISO-2022-JP	SJIS

TABLE 3-20 `iconv` Conversion Support *(continued)*

Source Code	Target Code
UTF-8	eucJP
UTF-8	SJIS
UTF-8	PCK
JIS7	eucJP
jis	eucJP
jis	PCK
jis	SJIS
ibmj	eucJP
ibmj	PCK
UTF-8	ISO-2022-JP
ISO-2022-JP	UTF-8
eucJP	UTF-8-Java
UTF-8-Java	eucJP
PCK	UTF-8-Java
UTF-8-Java	PCK
eucJP	ISO-2022-JP.RFC1468
PCK	ISO-2022-JP.RFC1468
UTF-8	ISO-2022-JP.RFC1468
eucJP	ibmj-EBCDIK

TABLE 3-20 `iconv` Conversion Support (continued)

Source Code	Target Code
<code>ibmj-EBCDIK</code>	<code>eucJP</code>
<code>PCK</code>	<code>ibmj-EBCDIK</code>
<code>ibmj-EBCDIK</code>	<code>PCK</code>

Japanese Character Code Converter for TTY STREAMS

There are TTY STREAMS modules that perform code conversion between an encoding for a specific terminal and an encoding for a specific locale. With an appropriate STREAMS module, a user can log in from a Japanese terminal into a Japanese locale, even if the encoding between the terminal and the Japanese locale does not match. `ttym(1)` controls the behavior of those STREAMS modules.

Japanese-specific Printer Support

The Japanese Solaris 7 product supports the following Japanese-specific printers:

- Epson VP-5085 (based on ESC/P)
- NEC PC-PR201 (based on 201PL)
- Canon LASERSHOT (based on LIPS)
- Japanese PostScript Printer

JLE Binary Compatibility Package

The Japanese Solaris 7 package also provides Japanese Solaris 1.1.x binary-compatibility packages that are the same as the base products.

User-Defined Character (UDC) Support

To handle User-Defined Characters, `sdtudctool` has been available since the Solaris 2.6 release. `sdtudctool` handles both outline (Type1) and bitmap (PCF) fonts. Some utilities are also available to migrate the UDC fonts that were created by old utilities, such as `fontedit`, `type3creator` and `fontmanager` in prior releases.

Korean Solaris 7 Product

The Korean Solaris product, used mostly in Korea, supports all the locales available in the English/Euro products. Additionally, it supports two Korean locales: `ko` and `ko.UTF-8`. In the `ko` locale, the EUC scheme is used to encode KSC 5601-1987. The `ko.UTF-8` locale supports the KSC 5700-1995/Unicode 2.0 codeset, which is a super set of KSC 5601-1987. These two locales look the same for the end user, but the internal character encoding is different. The Korean Solaris product supports the following Input Methods

for the `ko` locale:

- Hangul 2-BeolSik (1 set of consonants and 1 set of vowels)
- Hangul-Hanja conversion
- Special character
- Hexadecimal code

for the `ko.UTF-8` locale:

- Hangul 2-BeolSik (1 set of consonants and 1 set of vowels)
- Hangul-Hanja conversion
- Special character
- Hexadecimal code

The following fonts are available in the Korean version of the Solaris 7 product:

TABLE 3-21 Solaris 7 Korean CID/Type 1 Fonts for the `ko` Locale

Full Family Name	Subfamily	Format	Vendor	Encoding
Gothic	R	CID/Type 1	Hanyang	Adobe-Korean
Graphic	R	CID/Type 1	Hanyang	Adobe-Korean
Haeso	R	CID/Type 1	Hanyang	Adobe-Korean
Kodig	R	CID/Type 1	Hanyang	Adobe-Korean
Myeongjijo	R	CID/Type 1	Hanyang	Adobe-Korean
Pilki	R	CID/Type 1	Hanyang	Adobe-Korean
Roundgothic	R	CID/Type 1	Hanyang	Adobe-Korean

TABLE 3-22 Solaris 7 Korean Bitmap Fonts for the ko Locale

Full Family Name	Subfamily	Format	Encoding
Gothic	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1987
Graphic	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1987
Haeso	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1987
Kodig	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1987
Myeongjjo	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1987
Pilki	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1987
Roundgothic	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1987

TABLE 3-23 Solaris 7 Korean CID/Type 1 Fonts for the ko.UTF-8 Locale

Full Family Name	Subfamily	Format	Vendor	Encoding
Gothic	R	CID/Type 1	Hanyang	Adobe-Korean
Graphic	R	CID/Type 1	Hanyang	Adobe-Korean
Haeso	R	CID/Type 1	Hanyang	Adobe-Korean
Kodig	R	CID/Type 1	Hanyang	Adobe-Korean
Myeongjjo	R	CID/Type 1	Hanyang	Adobe-Korean
Pilki	R	CID/Type 1	Hanyang	Adobe-Korean

TABLE 3-24 Solaris 7 Korean Bitmap Fonts for the ko.UTF-8 Locale

Full Family Name	Subfamily	Format	Encoding
Gothic	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1992 (Johap)
Graphic	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1992 (Johap)
Haeso	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1992 (Johap)
Kodig	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1992 (Johap)

TABLE 3-24 Solaris 7 Korean Bitmap Fonts for the ko.UTF-8 Locale (continued)

Full Family Name	Subfamily	Format	Encoding
Myeongjjo	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1992 (Johap)
Pilki	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1992 (Johap)

TABLE 3-25 Korean ICONV

Code	Symbol	Target Code	Symbol
KSC 5601-1987	1506	UTF-8	UTF-8
ISO 646	646	KSC 5601-1987	5601
KSC 5601-1987	EUC-KR	UTF-8	UTF-8
KSC 5601-1987	KSC5601	UTF-8	UTF-8
UTF-8	UTF-8	KSC 5601-1987	5601
UTF-8	UTF-8	KSC 5601-1987	EUC-KR
UTF-8	UTF-8	KSC 5601-1987	KSC 5601
UTF-8	ko-KR-UTF-8	IBM CP 933	cp 933
UTF-8	ko-KR-UTF-8	KSC 5601-1987	ko_KR-euc
UTF-8	ko-KR-UTF-8	ISO2022-KR	ko_KR-iso2022-7
UTF-8	ko-KR-UTF-8	KSC 5601-1987 - Johap	ko_KR-johap
UTF-8	ko-KR-UTF-8	KSC5601-1992 - Johap	ko_KR-johap92
IBM CP933	cp933	UTF-8	ko_KR-UTF-8
KSC 5601-1987	ko_KR-euc	UTF-8	ko_KR-UTF-8
KSC 5601-1987	ko_KR-euc	ISO 2022-KR	ko_KR-iso2022-7
KSC 5601-1987	ko_KR-euc	KSC 5601-1987 - Johap	ko_KR-johap
KSC 5601-1987	ko_KR-euc	KSC 5601-1992 - Johap	ko_KR-johap92
KSC 5601-1987	ko_KR-euc	KSC 5601-1992-Annex:4	ko_KR-nbyte
ISO 2022-KR	iso2022-7	UTF-8	ko_KR-UTF-8
ISO 2022-KR	iso2022-7	KSC 5601-1987	ko_KR-euc

TABLE 3-25 Korean ICONV (continued)

Code	Symbol	Target Code	Symbol
KSC 5601-1987 - Johap	ko-KR-johap	UTF-8	ko_KR-UTF-8
KSC 5601-1987 - Johap	ko-KR-johap	KSC 5601-1987	ko_KR-euc
KSC 5601-1992 - Johap	ko-KR-johap92	UTF-8	ko_KR-UTF-8
KSC 5601-1992 - Johap	ko-KR-johap92	KSC 5601-1987	ko_KR-euc
KSC 5601-1992 - Annex:4	ko-KR-nbyte	KSC 5601-1987	ko_KR-euc

How to Use the `iconv` Command

The `iconv` command converts the characters or sequences of characters in a file from one codeset to another, then writes the results to standard output. If there is no conversion for a particular character, it is converted into an underscore '_' in the target codeset. See the `iconv(1)` man page for more information.

The following options are supported:

- `-f fromcode` Symbol of the input codeset.
- `-t tocode` Symbol of the output codeset.

To convert a mail file from one encoding into another, use the `iconv` command:

```
example% iconv -f from_codeset -t to_codeset mail.codeset > mail.codeset
```


Overview of en_US.UTF-8 Locale Support

en_US.UTF-8 Locale Support Overview

en_US.UTF-8 locale is the flagship Unicode locale in the Solaris 7 product that supports and provides multi-scripts processing capability by using UTF-8 as its codeset.

Note - UTF-8 is a file system safe Universal Character Set Transformation Format of Unicode / ISO/IEC 10646-1 formulated by X/JIG of X/Open in 1992 and approved by ISO and IEC as Amendment 2 to ISO/IEC 10646-1:1993 in 1996.

The locale supports computation of all characters defined in Unicode 2.1 / ISO/IEC 10646-1:1993 with Amendment 1 to 5 in the locale. However, due to a limited set of font resources and the fact that few users intend to use all of the code point values, users of the en_US.UTF-8 locale will see only character glyphs from the following character sets:

- ISO 8859-1 (Latin-1)
- ISO 8859-2 (Latin-2)
- ISO 8859-4 (Latin-4)
- ISO 8859-5 (Latin/Cyrillic)
- ISO 8859-6 (Arabic)
- ISO 8859-7 (Latin/Greek)

- ISO 8859-8 (Hebrew)
- ISO 8859-9 (Latin-5)
- BIG5 (Traditional Chinese)
- GB 2312-1980 (Simplified Chinese)
- JIS X0201-1976, JIS X0208-1983 (Japanese)
- KS C 5601-1992 Annex 3 (Korean)
- ISO 8859-8 (Hebrew)
- ISO 8859-11 TIS 620.7573 (Thai)

The above coverage of the scripts has significantly increased in the Solaris 7 product, which now also supports the following range of scripts:

- Western/Eastern/Northern European
- Greek
- Turkish
- Cyrillic
- Simplified Chinese
- Traditional Chinese
- Japanese
- Korean
- Arabic
- Hebrew
- Thai

Since this locale is primarily for developers, it belongs to the developer's cluster of the Solaris 7 product. Thus, when you install the Solaris 7 product, you should choose the developer's cluster to install the locale on your system.

Exactly the same level of `en_US.UTF-8` locale support is provided for both 32-bit and 64-bit Solaris systems.

Note - Motif and CDE desktop applications and libraries support the `en_US.UTF-8` locale. However, OpenWindows, XView, and, OPENLOOK DeskSet applications and libraries do *not* support the `en_US.UTF-8` locale.

System Environment

Locale Environment Variable

To use the `en_US.UTF-8` locale environment, make sure to choose the locale first. Be sure you have the `en_US.UTF-8` locale installed on your system.

▼ To use the `en_US.UTF-8` locale environment

1. In a TTY environment, choose the locale first, by setting the `LANG` environment variable to `en_US.UTF-8`, as in the following C-shell example:

```
system% setenv LANG en_US.UTF-8
```

2. Make sure that other categories are not set (or are set to `en_US.UTF-8`) since the `LANG` environment variable has a lower priority than other environment variables such as `LC_ALL`, `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_NUMERIC`, `LC_MONETARY` and `LC_TIME` at setting the locale. See the `setlocale(3C)` man page for more details about the hierarchy of environment variables.
3. To check current locale settings in various categories, use the `locale(1)` utility. `locale(1)`
)

```
system% locale
LANG=en_US.UTF-8
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_ALL=
```

You can also start the `en_US.UTF-8` environment from the CDE desktop. At the CDE login screen's Options -> Language menu, choose `en_US.UTF-8`.

TTY Environment Setup

To ensure correct text edit operation by a terminal or by a terminal emulator such as `dtterm(1)`, users should push certain locale-specific STREAMS modules onto their Streams.

For more information on STREAMS modules and streams in general, see the *STREAMS Programming Guide*.

Table 4-1 shows STREAMS modules supported by the `en_US.UTF-8` locale in the terminal environment:

TABLE 4-1 32-bit STREAMS Modules Supported by `en_US.UTF-8`

32-bit STREAMS Module	Description
<code>/usr/kernel/strmod/eucu8</code>	UTF-8 STREAMS module for tail side
<code>/usr/kernel/strmod/u8euc</code>	UTF-8 STREAMS module for head side
<code>/usr/kernel/strmod/u8lat1</code>	Code conversion STREAMS module between UTF-8 and ISO 8859-1 (Western Europe)
<code>/usr/kernel/strmod/u8lat2</code>	Code conversion STREAMS module between UTF-8 and ISO 8859-2 (Eastern Europe)
<code>/usr/kernel/strmod/u8koi8</code>	Code conversion STREAMS module between UTF-8 and KOI8-R (Cyrillic)

Table 4-2 lists the 64-bit STREAMS Modules Supported by `en_US.UTF-8`.

TABLE 4-2 64-bit STREAMS Modules Supported by `en_US.UTF-8`

64-bit STREAMS module	Description
<code>/usr/kernel/strmod/sparcv9/eucu8</code>	UTF-8 STREAMS module for tail side
<code>/usr/kernel/strmod/sparcv9/u8euc</code>	UTF-8 STREAMS module for head side
<code>/usr/kernel/strmod/sparcv9/u8lat1</code>	Code conversion STREAMS module between UTF-8 and ISO 8859-1 (Western European)

TABLE 4-2 64-bit STREAMS Modules Supported by en_US.UTF-8 (continued)

64-bit STREAMS module	Description
/usr/kernel/strmod/sparcv9/u8lat2	Code conversion STREAMS module between UTF-8 and ISO 8859-2 (Eastern European)
/usr/kernel/strmod/sparcv9/u8koi8	Code conversion STREAMS module between UTF-8 and KOI8-R (Cyrillic)

Loading a STREAMS Module at Kernel

To load a STREAMS module at kernel, first become root:

```
system% su
Password:
system#
```

To determine whether you are running a 32-bit Solaris or 64-bit Solaris system, use the `isainfo(1)` utility as follows:

```
system# isainfo -v
64-bit sparcv9 applications
32-bit sparc applications
system#
```

If the command returns this information, you are running the 64-bit Solaris system. If you are running the 32-bit Solaris system, the utility shows the following:

```
system# isainfo -v
32-bit sparc applications
system#
```

Use `modinfo(1M)` to be certain that your system has not already loaded the STREAMS module:

```
system# modinfo | grep eucu8modulename
```

If the STREAMS module, such as `eucu8`, is already installed, the output will look as follows:

```
system# modinfo | grep eucu8
89 ff798000 4b13 18 1 eucu8 (eucu8 module)
```

(Continuation)

```
system#
```

If the module is already installed, you don't need to load it. However, if the module has not yet been loaded, use `modload(1M)` as follows:

```
system# modload /usr/kernel/strmod/eucu8modulename
```

This loads the 32-bit `eucu8` STREAMS module at the kernel, so you can push it onto a Stream. If you are running the 64-bit Solaris product, use `modload(1M)` as follows:

```
system# modload /usr/kernel/strmod/sparcv9/eucu8
```

The STREAMS module is installed at the kernel, and you can now push it onto a Stream.

To unload a module from the kernel, use `modunload(1M)`, as shown below. In this example, the `eucu8` module is being unloaded.

```
system# modinfo | grep eucu8
89 ff798000 4b13 18 1 eucu8 (eucu8 module)
system# modunload -i 89
```

dtterm and Terminals Capable of Input and Output of UTF-8 Characters

The `dtterm(1)` and any terminal that supports input and output of the UTF-8 codeset should have the following STREAMS configuration:

```
head <-> ttcompat <-> u8euc <-> ldterm <-> eucu8 <-> pseudo-TTY
```

In this example, `u8euc` and `eucu8` are the modules supported by the `en_US.UTF-8` locale. Make sure you already loaded the STREAMS modules into the kernel as specified in the previous section.

To set up the above STREAMS configuration, use, `strchg(1)` as shown below:


```
system% cat > /tmp/mystreams
ttcompat
u8euc
ldterm
eucu8
ptem
^D
system% strchg -f /tmp/mystreams
```

When using `strchg(1)`, be sure you are either root or the owner of the device. To see the current configuration of STREAMS, use `strconf(1)` as shown below:

```
system% strconf
ttcompat
u8euc
ldterm
eucu8
ptem
pts
system%
```

To revert to the original configuration, set the STREAMS configuration again as shown below:

```
system% cat > /tmp/orgstreams
ttcompat
ldterm
ptem
^D
system% strchg -f /tmp/orgstreams
```

Terminal Support for Latin-1, Latin-2, or KOI8-R

For terminals that support only Latin-1 (ISO 8859-1), Latin-2 (ISO 8859-2), or KOI8-R, you should have the following STREAMS configuration:

```
head <-> ttcompat <-> u8euc <-> ldterm <-> eucu8 <-> u8lat1 <-> TTY
```

Note - This configuration is only for terminals that support Latin-1. For Latin-2 terminals, replace the STREAMS module `u8lat1` with `u8lat2`. For KOI8-R terminals, replace the module with `u8koi8`.

To set up the STREAMS configuration shown above, use `strchg(1)`, as follows:

```
system% cat > tmp/mystreams
ttcompat
```

(Continuation)

```
u8euc
ldterm
eucu8
u8lat1
ptem
^D
system% strchg -f /tmp/mystreams
```

Be sure that you are either root or the owner of the device when you use `strchg(1)`. To see the current configuration, use `strchg(1)`, as follows:

```
system% strconf
ttcompat
u8euc
ldterm
eucu8
u8lat1
ptem
pts
system%
```

To revert to the original configuration, set the STREAMS configuration as follows:

```
system% cat > /tmp/orgstreams
ttcompat
ldterm
ptem
^D
system% strchg -f /tmp/orgstreams
```

Setting Terminal Options

To set up the UTF-8 text edit behavior on TTY, you must first set some terminal options using `stty(1)` as follows:

```
system% /bin/stty cs8 -istrip defeucw
```

Note - Since `/usr/ucb/stty` is not yet internationalized, you should use `/bin/stty` instead.

You can also query the current settings using `stty(1)` with the `-a` option, as shown below:

```
system% /bin/stty -a
```

Saving the Settings in ~/.cshrc

Assuming the necessary STREAMS modules are already loaded with the kernel, you can save the following lines in your `.cshrc` file (C shell example) for convenience:

```
setenv LANG en_US.UTF-8
if ($?USER != 0 && $?prompt != 0) then
    cat >! /tmp/mystreams$$ << _EOF
        ttcompat
        u8euc
        ldtterm
        eucu8
        ptem
    _EOF
    /bin/strchg -f /tmp/mystream$$
    /bin/rm -f /tmp/mystream$$
    /bin/stty cs8 -istrip defeucw
endif
```

With these lines in your `.cshrc` file, you do not have to type all of the commands each time. Note that the second `_EOF` should be in the first column of the file. You can also create a file called `mystreams` and save it so the `.cshrc` references to `mystreams` instead of creating it whenever you start a C shell.

Code Conversions

The `en_US.UTF-8` locale supports various code conversions among major codesets of several countries through `iconv(1)` and `iconv(1)`.

The available `fromcode` and `to code` names that can be applied to `iconv(1)` and `iconv_open(3)` are shown in Table 4-3.

TABLE 4-3 Available Code Conversions in `en_US.UTF-8`

From Code	To Code	Description
646	UTF-8	ISO 646 (US-ASCII) to UTF-8
UTF-8	646	UTF-8 to ISO 646 (US-ASCII)
UTF-8	8859-1	UTF-8 to ISO 8859-1
UTF-8	8859-2	UTF-8 to ISO 8859-2

TABLE 4-3 Available Code Conversions in `en_US.UTF-8` (continued)

From Code	To Code	Description
UTF-8	8859-3	UTF-8 to ISO 8859-3
UTF-8	8859-4	UTF-8 to ISO 8859-4
UTF-8	8859-5	UTF-8 to ISO 8859-5 (Cyrillic)
UTF-8	8859-6	UTF-8 to ISO 8859-6 (Arabic)
UTF-8	8859-7	UTF-8 to ISO 8859-7 (Greek)
UTF-8	8859-8	UTF-8 to ISO 8859-8 (Hebrew)
UTF-8	8859-9	UTF-8 to ISO 8859-9
UTF-8	8859-10	UTF-8 to ISO 8859-10
UTF-8	8859-11	UTF-8 to TIS 620.2533 (Thai)
UTF-8	8859-15	UTF-8 to ISO 8859-15
8859-1	UTF-8	ISO 8859-1 to UTF-8
8859-2	UTF-8	ISO 8859-2 to UTF-8
8859-3	UTF-8	ISO 8859-3 to UTF-8
8859-4	UTF-8	ISO 8859-4 to UTF-8
8859-5	UTF-8	ISO 8859-5 (Cyrillic) to UTF-8
8859-6	UTF-8	ISO 8859-6 (Arabic) to UTF-8
8859-7	UTF-8	ISO 8859-7 (Greek) to UTF-8
8859-8	UTF-8	ISO 8859-8 (Hebrew) to UTF-8
8859-9	UTF-8	ISO 8859-9 to UTF-8
8859-10	UTF-8	ISO 8859-10 to UTF-8
8859-11	UTF-8	TIS 620.2553 to UTF-8
8859-15	UTF-8	ISO 8859-15 to UTF-8
UTF-8	KOI8-R	UTF-8 to KOI8-R (Cyrillic)

TABLE 4-3 Available Code Conversions in `en_US.UTF-8` (continued)

From Code	To Code	Description
KOI8-R	UTF-8	KOI8-R (Cyrillic) to UTF-8
UTF-8	UCS-2	UTF-8 to UCS-2
UCS-2	UTF-8	UCS-2 to UTF-8
UTF-8	UCS-4	UTF-8 to UCS-4
UCS-4	UTF-8	UCS-4 to UTF-8
UTF-8	UTF-7	UTF-8 to UTF-7
UTF-7	UTF-8	UTF-7 to UTF-8
UTF-8	UTF-16	UTF-8 to UTF-16
UTF-16	UTF-8	UTF-16 to UTF-8
UTF-8	euCJP	UTF-8 to Japanese EUC (JIS X0201-1976, JIS X0208-1983, and JIS X0212-1990)
UTF-8	PCK	UTF-8 to Japanese PC Kanji (SJIS)
UTF-8	ISO-2022-JP	UTF-8 to Japanese MIME character set ISO-2022-JP
euCJP	UTF-8	Japanese EUC to UTF-8
PCK	UTF-8	Japanese PC Kanji (SJIS) to UTF-8
ISO-2022-JP	UTF-8	Japanese MIME character set to UTF-8
UTF-8	ko_KR-euc	UTF-8 to Korean EUC (KS C 5636 and KS C 5601-1987)
UTF-8	ko_KR-johap	UTF-8 to Korean Johap (KS C 5601-1987)
UTF-8	ko_KR-johap92	UTF-8 to Korean Johap (KS C 5601-1992)
UTF-8	ko_KR-iso2022-7	UTF-8 to ISO-2022-KR
ko_KR-euc	UTF-8	Korean EUC to UTF-8
ko_KR-johap	UTF-8	Korean Johap (KS C 5601-1987) to UTF-8

TABLE 4-3 Available Code Conversions in `en_US.UTF-8` (continued)

From Code	To Code	Description
<code>ko_KR-johap92</code>	<code>UTF-8</code>	Korean Johap (KS C 5601-1992) to UTF-8
<code>ko_KR-iso2022-7</code>	<code>UTF-8</code>	ISO-2022-KR to UTF-8
<code>ko_KR-cp933</code>	<code>UTF-8</code>	IBM MBCS CP933 to UTF-8
<code>UTF-8</code>	<code>gb2312</code>	UTF-8 to Simplified Chinese EUC (GB 1988-1980 and GB2312-1980)
<code>UTF-8</code>	<code>iso2022</code>	UTF-8 to ISimplified Chinese MIME character set (ISO-2022-cn)
<code>UTF-8</code>	<code>GBK</code>	UTF-8 to Simplified Chinese MIME character set (ISO-2022-cn)
<code>gb2312</code>	<code>UTF-8</code>	Chinese/PRC EUC (GB 2312-1980) to UTF-8
<code>iso2022</code>	<code>UTF-8</code>	ISO-2022-CN to UTF-8
<code>GBK</code>	<code>UTF-8</code>	Simplified Chinese GBK to UTF-8
<code>UTF-8</code>	<code>zh_TW-euc</code>	UTF-8 to Traditional Chinese EUC (CNS 11643-1992)
<code>UTF-8</code>	<code>zh_TW-big5</code>	UTF-8 to Traditional Chinese Big5
<code>UTF-8</code>	<code>zh_TW-iso2022-7</code>	UTF-8 to Traditional Chinese MIME character set (ISO-2022-TW)
<code>UTF-8</code>	<code>zh_TW-cp937</code>	UTF-8 to IBM MBCS CP937
<code>zh_TW-euc</code>	<code>UTF-8</code>	Traditional Chinese EUC to UTF-8
<code>zh_TW-big5</code>	<code>UTF-8</code>	Traditional Chinese Big5 to UTF-8
<code>zh_TW-iso2022-7</code>	<code>UTF-8</code>	Traditional Chinese MIME character set (ISO-2022-TW) to UTF-8
<code>zh_TW-cp937</code>	<code>UTF-8</code>	IBM MBCS CP937 to UTF-8

For more details on `iconv` code conversion, see the `, iconv(1)` and `iconv_open(3)`, `iconv(3)`, and `iconv_close(3)` man pages. For more information on available code conversions, see `iconv_en_US.UTF-8(5)`.

Script Selection and Input Modes

The `en_US.UTF-8` locale supports multiple scripts. There are a total of eight input modes in the `en_US.UTF-8` locale:

- English/European
- Cyrillic
- Greek
- Arabic
- Hebrew
- Thai
- Unicode Hexadecimal code input method
- Table lookup input method

English/European Input Mode

The English/European input mode includes not only the English alphabet but also characters with diacritical marks (for example, á, è, î, ò, and ü) and special characters (such as ¡, §, ¿) from European scripts.

This input mode is the default mode for any application. The input mode is displayed at the bottom left corner of the GUI application.

```
[ English ]
```

To insert characters with diacritical marks or special characters from Latin-1, Latin-2, Latin-4, Latin-5 and Latin-9, you must type a Compose sequence, as shown in the following examples:

- For Ä, press and release Compose, then A, and then "
- For ¿, press and release Compose, then +, and then -

When there is no <Compose> key available on your keyboard, you can substitute for the <Compose> key by simultaneously pressing the <Control> key and <shift-T> together.

For the input of the Euro currency symbol (Unicode value U+20AC) from the locale, you can use any one of following input sequences:

- <AltGraph> and <e> together
- <AltGraph> and <4> together, or
- <AltGraph> and <5> together

These input sequences mean that you press both keys simultaneously. If there is no <AltGraph> key available on your keyboard, you can substitute the <Alt> key for the <AltGraph> key.

The following tables are the most commonly used Compose sequences in Latin-1, Latin-2, Latin-4, Latin-5 and Latin-9 script input for Sparc.

Note - To start these sequences, type <Compose> key and release it.

Table 4-4 lists the Common Latin-1 Compose Sequences.

TABLE 4-4 Common Latin-1 Compose Sequences for Sparc

Press and Release	Press and Release	Result
[Spacebar]	[Spacebar]	Non-breaking space
s	1	Superscripted 1
s	2	Superscripted 2
s	3	Superscripted 3
!	!	Inverted exclamation mark
x	o	Currency symbol ¢
p	!	Paragraph symbol ¶
/	u	mu u
'	'	apostrophe '´
'	"	acute accent ´
'	,	cedilla ,´
"	"	dieresis ¨
-	^	macron ´
o	o	degree °
x	x	multiplication sign ´x´
+	-	plus-minus ±

TABLE 4-4 Common Latin-1 Compose Sequences for Sparc (continued)

Press and Release	Press and Release	Result
-	-	soft hyphen -
-	:	division sign /
-	a	ordinal (feminine) a ā
a	-	ordinal (feminine) a ā
-	o	ordinal (masculine) o ō
o	-	ordinal (masculine) o ō
-	,	not sign ¬
.	.	middle dot '
1	2	vulgar fraction 1/2
1	4	vulgar fraction 1/4
3	4	vulgar fraction 3/4
<	<	left double angle quotation mark «
>	>	right double angle quotation mark »
?	?	inverted question mark ¿
A	`	A grave À
A	'	A acute Á
A	*	A ring above Â
A	"	A dieresis Ä
A	^	A circumflex Ã
A	~	A tilde ã
A	E	AE diphthong Æ
C	,	C cedilla Ç

TABLE 4-4 Common Latin-1 Compose Sequences for Sparc *(continued)*

Press and Release	Press and Release	Result
C	o	copyright sign ©
D	-	Capital eth D
E	‘	E grave È
E	’	E acute É
E	"	E dieresis Ê
E	^	E circumflex Ê
I	‘	I grave Ì
I	’	I acute Í
I	"	I dieresis Î
I	^	I circumflex Î
L	-	pound sign \xa3
N	~	N tilde Ñ
O	‘	O grave Ò
O	’	O acute Ó
O	/	O slash Ø
O	"	O dieresis Ö
O	^	O circumflex Ô
O	~	O tilde Õ
R	O	registered mark ®
T	H	Thorn P
U	‘	U grave Û

TABLE 4-4 Common Latin-1 Compose Sequences for Sparc *(continued)*

Press and Release	Press and Release	Result
U	'	U acute Ú
U	"	U dieresis Ü
U	^	U circumflex Û
Y	'	Y acute Ý
Y	-	yen sign ¥
a	`	a grave à
a	'	a acute á
a	*	a ring above ä
a	"	a dieresis ä
a	^	a circumflex â
a	~	a tilde ã
a	^	a circumflex â
a	e	ae diphthong æ
c	,	c cedilla ç
c	/	cent sign ¢
c	o	copyright sign ©
d	-	eth d
e	`	e grave è
e	'	e acute é
e	"	e dieresis ë
e	^	e circumflex ê
i	`	i grave ì

TABLE 4-4 Common Latin-1 Compose Sequences for Sparc (continued)

Press and Release	Press and Release	Result
i	'	i acute í
i	"	i dieresis ï
i	^	i circumflex î
n	~	n tilde ñ
o	`	o grave ò
o	'	o acute ó
o	/	o slash ø
o	"	o dieresis ö
o	^	o circumflex ô
o	~	o tilde õ
s	s	German double s ß
t	h	thorn þ
u	`	u grave ù
u	'	u acute ú
u	"	u dieresis ü
u	^	u circumflex û
y	'	y acute ý
y	"	y dieresis ÿ
		broken bar

Note - Compose sequences defined in Table 4-3 are not included in Table 4-4..

Table 4-5 lists the Common Latin-2 Compose Sequences.

TABLE 4-5 Common Latin-2 Compose Sequences

Press and Release	Press and Release	Result
a	'	ogonek á
u	''	breve ü
v	''	caron
"	''	double acute "
A	a	A ogonek a
A	u	A breve
C	'	C acute
C	v	C caron
D	v	D caron
-	D	D stroke
E	v	E caron
E	a	E ogonek
L	'	L acute
L	-	L stroke
L	>	L caron
N	'	N acute
N	v	N caron
O	>	O double acute
S	'	S acute
S	v	S caron
S	,	S cedilla
R	'	R acute

TABLE 4-5 Common Latin-2 Compose Sequences *(continued)*

Press and Release	Press and Release	Result
R	v	R caron
T	v	T caron
T	,	T cedilla
U	*	U ring above
U	>	U double acute
Z	'	Z acute
Z	v	Z caron
Z	.	Z dot above

Table 4-5 contains the Latin-2 compose sequences.

Note - Compose sequences defined in Table 4-3 or Table 4-4 are not included in Table 4-5.

Table 4-6 lists the Common Latin-4 Compose Sequences.

TABLE 4-6 Common Latin-4 Compose Sequences

Press and Release	Press and Release	Result
k	k	kra
A	_	A macron
E	_	E macron
E	.	E dot above
G	,	G cedilla
I	_	I macron
I	~	I tilde

TABLE 4-6 Common Latin-4 Compose Sequences *(continued)*

Press and Release	Press and Release	Result
I	a	I ogonek
K	,	K cedilla
L	,	L cedilla
N	,	N cedilla
O	_	O macron
R	,	R cedilla
T		T stroke
U	~	U tilde
U	a	U ogonek
U	_	U macron
N	N	Eng
a	_	a macron
e	_	e macron
e	.	e dot above
g	,	g cedilla
i	_	i macron
i	~	i tilde
i	a	i ogonek
k	,	k cedilla
l	,	l cedilla
n	,	n cedilla
o	_	o macron

TABLE 4-6 Common Latin-4 Compose Sequences (continued)

Press and Release	Press and Release	Result
r	,	r cedilla
t		t stroke
u	~	u tilde
u	a	u ogonek
u	–	u macron
n	n	eng

Note - Compose sequences defined in Table 4-3 or Table 4-4 or Table 4-5 are not included in Table 4-6.

Table 4-7 lists the Common Latin-5 Compose Sequences.

TABLE 4-7 Common Latin-5 Compose Sequences

Press and Release	Press and Release	Result
G	u	G breve
I	.	I dot above
g	u	g breve
i	.	i dotless

Any compose sequences already described do not re-appear in this table.

Table 4-8 lists the Common Latin-9 Compose Sequences.

TABLE 4-8 Common Latin-9 Compose Sequences

Press and Release	Press and Release	Result
o	e	Diphthong oe
O	E	Diphthong OE
Y	"	Y diaeresis

Cyrillic Input Mode

To switch to Cyrillic input mode from English input mode, press Compose c c. If you are currently in Greek input mode, first return to English input mode, then switch to Cyrillic mode.

The input mode is displayed at the bottom left corner of your GUI application.

[Cvrillic]

After you switch to Cyrillic input mode, you cannot enter English text. To switch back to English input mode, type Control-Space. The Russian keyboard layout appears in Figure 4-1.

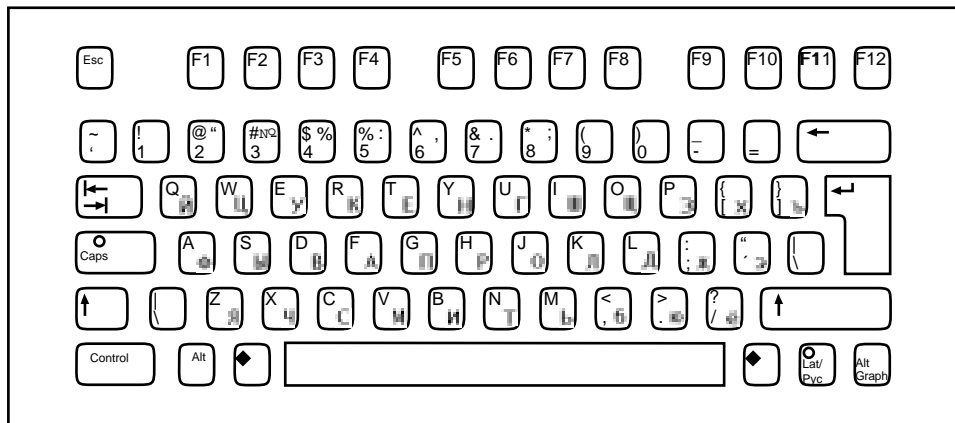


Figure 4-1 Cyrillic Keyboard

Greek Input Mode

To switch to Greek input mode from English input mode, press Compose g g. If you are currently in Cyrillic input mode, first return to English input mode and then switch to Greek mode.

The input mode is displayed at the left bottom corner of your GUI application .

[Greek]

After you switch to Greek input mode, you cannot enter English text. To switch back to English input mode, type Control-Space. The Greek keyboard layouts appear in Figures 4-2 and 4-3.

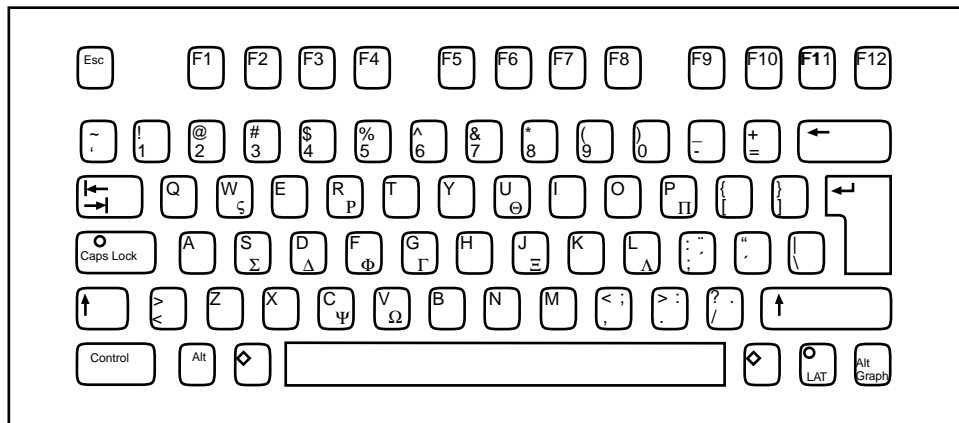


Figure 4-2 Greek Euro Keyboard

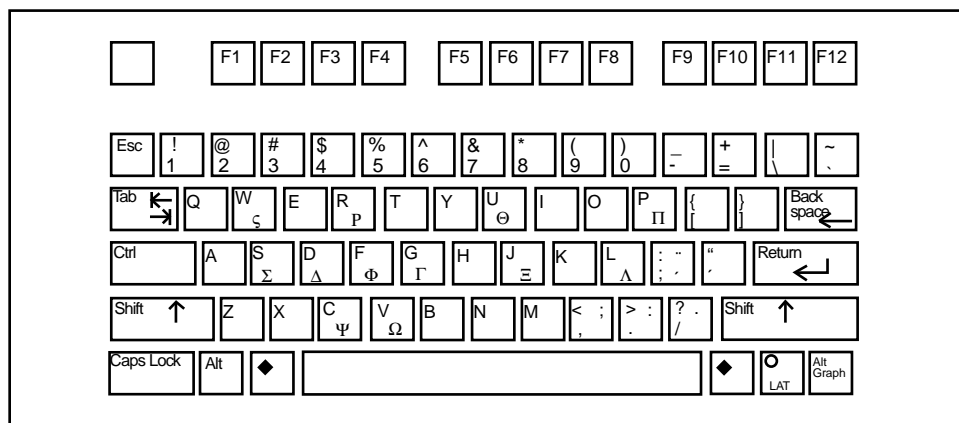


Figure 4-3 Greek UNIX Keyboard

Arabic Input Mode

To switch to Arabic input mode, type <Compose> + <g> + <g> from your current input mode. The input mode is displayed at the left bottom corner of your GUI application. Once you switch to the Arabic input mode, you have to switch back to English/European input mode to enter English/European characters by typing <Control> and <Space> together.

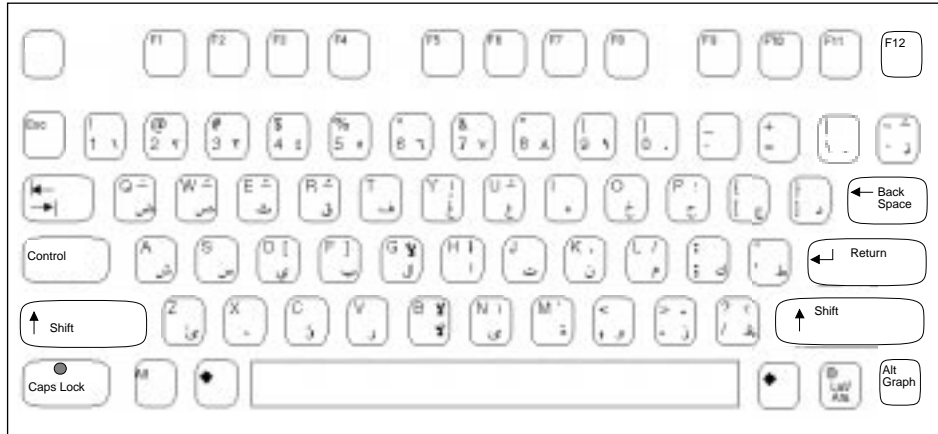


Figure 4-4 Arabic Keyboard

Hebrew Input Mode

To switch into Hebrew input mode, type <Compose> <h> <h> from your current input mode. The input mode is displayed at the left bottom corner of your GUI application.

Once you switched into the Hebrew input mode, you have to switch back to the English/European input mode to enter English/European characters by typing <Control> and <Space> and together. You can also switch into other input modes by typing the corresponding input mode switch key sequence. The Hebrew keyboard layout is shown at following figure:

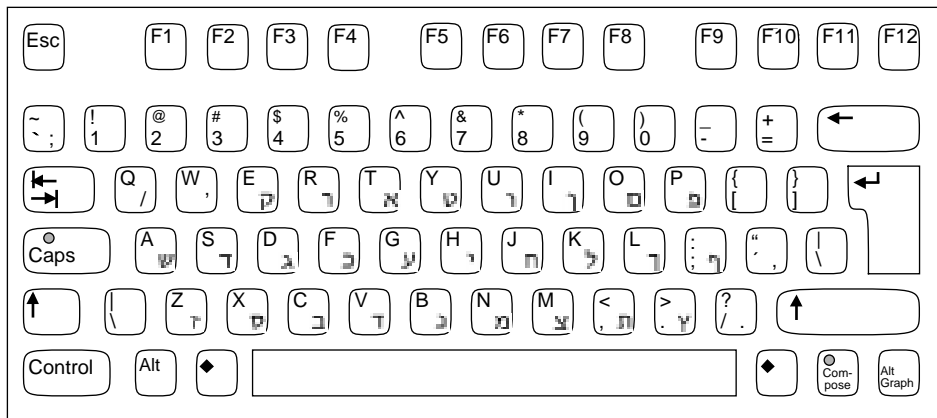


Figure 4-5 Hebrew Keyboard

Thai Input Mode

To switch into Thai input mode, type <Compose> <t> <t> from your current input mode. The input mode will be displayed at the left bottom corner of your GUI application.



Once you have switched into the Thai input mode, you have to switch back to English/European input mode to enter English/European characters by typing <Control> and <Space> together. You can also switch into other input modes by typing the corresponding input mode switch key sequence. The Thai keyboard layout is shown at following figure:

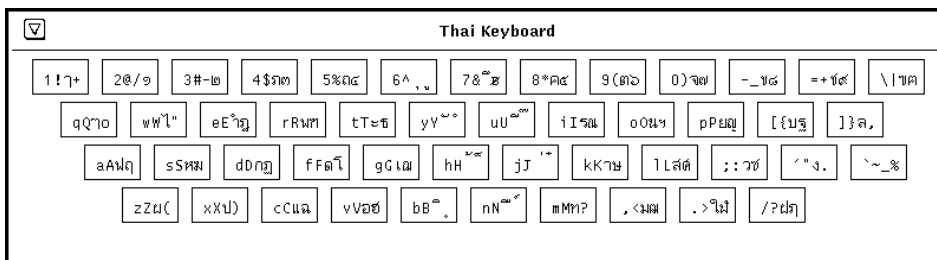


Figure 4-6 Thai Keyboard

Unicode Hexadecimal Code Input Method Input Mode

To switch into the Unicode hexadecimal code input method input mode, type `<Compose> <l> <l>` from your current input mode. The input mode is displayed at the left bottom corner of your GUI application:

To use this input mode, you need to know about the hexadecimal code point values of the characters. Refer to *The Unicode Standard, Version 2.0* for the mapping between code point values and characters. To input a character, type four hexadecimal digits, for instance, 00a1 for Inverted Exclamation Mark, 03b2 for Greek Small Letter Beta, ac00 for a Korean Hangeul Syllable KA, 30a2 for Japanese Katakana Letter A, 4e58 for a Unified Han character and so on. Users can use both uppercase and lowercase letters of A, B, C, D, E, and, F for hexadecimal digits. If you mistype a digit or two, you can delete the digits by using the `<Delete>` key or the `<Backspace>` key.

Table Lookup Input Method Input Mode

To switch into table lookup input method input mode, type `<Compose> <l> <l>` from your current input mode. The input mode is displayed at the left bottom corner of your GUI application.

Once you turn on the input mode, there is a lookup window showing multiple candidates of Unicode characters. You can choose any one of the candidates by moving your pointer and clicking the left button on your mouse. You can also select any one of the candidates by choosing a left-hand- side letter associated with each of the candidates.

Once you are finished using the current input mode, you can switch into other input mode by typing corresponding input mode switch key sequence.

Input Mode Switch Key Sequence Summary

Starting in the Solaris 7 environment, , users can switch from one input mode to another without any restrictions. The following table shows the input mode switch key sequences for each input mode.

TABLE 4-9 Input Mode Switch Key Sequences

Input Mode	Key Sequences
English/European	<code><Control> + <Space></code>
Cyrillic	<code><Compose> <c> <c></code>

TABLE 4-9 Input Mode Switch Key Sequences (continued)

Input Mode	Key Sequences
Greek	<Compose> <g> <g>
Arabic	<Compose> <a> <r>
Hebrew	<Compose> <h> <h>
Thai	<Compose> <t> <t>
Unicode hexadecimal code input method	<Compose> <u> <h>
Table lookup input method	<Compose> <l> <l>

Printing

The `en_US.UTF-8` locale provides a printing utility, `xutops(1)`. This utility can print flat text files written in UTF-8 using X11 bitmap fonts available on the system. Because the output from the utility is standard PostScript, the output can be sent to any PostScript printer.

To use the utility, type the following:

```
system% xutops filename | lp
```

You can also use the utility as a filter since the utility accepts `stdin` stream:

```
system% lpr filename | xutops | lp
```

You can set the utility as a printing filter for a line printer. For example, the following command sequence tells the printer service LP that the printer `lp1` accepts only `xutops` format files. This command line also installs the printer `lp1` on port `/dev/ttya`. See the `lpadmin(1M)` man page for more details.

```
system# lpadmin -p lp1 -v /dev/ttya -I XUTOPS
system# accept lp1
system# enable lp1
```

Using `lpfilter(1M)`, you can add the utility as a filter as follows:

```
system# lpfilter -f filtername -F pathname
```

The command tells the printer that a converter (in this case, `xutops`) is available through the filter description file named *pathname*. *Pathname* can be as follows:

```
Input types: simple
Output types: XUTOPS
Command: /usr/openwin/bin/xutops
```

The filter converts the default type file input to PostScript output using `/usr/openwin/bin/xutops`.

To print a UTF-8 text file, use the following command:

```
system% lp -T XUTOPS UTF-8-file
```

DtMail

As a result of increased coverage in scripts, Solaris 7 DtMail running in the `en_US.UTF-8` locale supports various MIME character sets shown below.

- US-ASCII (7-bit US ASCII)
- UTF-8 (UCS Transmission Format 8 of Unicode)
- UTF-7 (UCS Transmission Format 7 of Unicode)
- ISO-8859-1 (Latin-1)
- ISO-8859-2 (Latin-2)
- ISO-8859-3 (Latin-3)
- ISO-8859-4 (Latin-4)
- ISO-8859-5 (Latin/Cyrillic)
- ISO-8859-6 (Latin/Arabic)
- ISO-8859-7 (Latin/Greek)
- ISO-8859-8 (Latin/Hebrew)
- ISO-8859-9 (Latin-5)
- ISO-8859-10 (Latin-6)
- ISO-8859-15 (Latin-9)
- KOI8-R (Cyrillic)
- ISO-2022-JP (Japanese)
- ISO-2022-KR and EUC-KR (Korean)

- ISO-2022-CN (Simplified Chinese)
- ISO-2022-TW (Traditional Chinese)

This support allows users to view virtually any kind of email encoded in various MIME character sets from any region of the world in a single instance of DtMail. The decoding of received email is done by DtMail, which looks at the MIME character set and content transfer encoding provided with the email. However, in case of sending, you need to specify a MIME character set that is understood by the recipient mail user agent (in other words, mail client), unless you want to use the default MIME character set provided by the `en_US.UTF-8` locale. To switch the character set of out-going email, at the 'New Message' window, type either `<CONTROL> + <Y>` or click the "Format" menu button and then again click on the "Change Char Set" button by using your mouse. The next available character set name will be displayed at left bottom corner on top of the Send button. If your email message header or message body contains characters that cannot be represented by the MIME charset specified, the system automatically switches the MIME character set to the `UTF-8` that can represent any characters.

If your message contains characters from the 7-bit US-ASCII character set only, your email's default MIME character set is `US-ASCII`. Any mail user agent can interpret such email message without any loss of characters or information.

If your message contains characters from a mixture of scripts, your email's default MIME character set is `UTF-8` and any 8-bit characters of `UTF-8` is encoded with Quoted-Printable encoding. For more detail on MIME, registered MIME charsets and Quoted-Printable encoding, refer to RFC 2045, 2046, 2047, 2048, 2049, 2279, 2152, 2237, 1922, 1557, 1555, and, 1489.

Programming Environment

Appropriately, internationalized applications should automatically enable the `en_US.UTF-8` locale, but proper `FontSet/XmFontList` definitions in the application's resource file are required.

For information on internationalized applications, see *Creating Worldwide Software: Solaris International Developer's Guide*, 2nd edition.

Font Set Used with X Applications

The `en_US.UTF-8` locale in the Solaris 7 environment supports fonts for the following character sets.

- ISO 8859-1

- ISO 8859-2
- ISO 8859-4
- ISO 8859-5
- ISO 8859-7
- ISO 8859-9
- ISO 8859-15
- BIG5
- GB 2312-1980
- JISX 0201.1976
- JISX 0208.1983
- KSC 5601.1992-3
- ISO 8859-6-1
- ISO 8859-8
- TIS 620.2533-1

Because the Solaris 7 environment supports the CDE desktop environment, each character set has a guaranteed sets of fonts.

The following is a list of the Latin-1 fonts that are supported in the Solaris 7 product.

```
-dt-interface system-medium-r-normal-xxs sans
  utf-10-100-72-72-p-59-iso8859-1
-dt-interface system-medium-r-normal-xs sans
  utf-12-120-72-72-p-71-iso8859-1
-dt-interface system-medium-r-normal-s sans
  utf-14-140-72-72-p-82-iso8859-1
-dt-interface system-medium-r-normal-m sans
  utf-17-170-72-72-p-97-iso8859-1
-dt-interface system-medium-r-normal-l sans
  utf-18-180-72-72-p-106-iso8859-1
-dt-interface system-medium-r-normal-xl sans
  utf-20-200-72-72-p-114-iso8859-1
-dt-interface system-medium-r-normal-xxl sans
  utf-24-240-72-72-p-137-iso8859-1
```

For information on CDE common font aliases, including `-dt-interface user-*` and `-dt-application-*` aliases, see *Common Desktop Environment: Internationalization Programmer's Guide*.

In the `en_US.UTF-8` locale, `utf` is also supported as a common font alias. A font set for an application should have a collection of fonts that contains each of the character sets, as in the following example.

```
fs = XCreateFontSet(display,
"-dt-interface system-medium-r-normal-s*utf-**-**-**-**-iso8859-1,
```

(Continuation)

```
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-2,  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-5,  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-6,  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-7,  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-8,  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-9",  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-15",  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*big5-1",  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*gb2312.1980-0",  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*jisx0201.1976-0",  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*jisx0208.1983-0",  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*kcs5601.1992-3",  
-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*tis620.2533-0",  
    &missing_ptr, &missing_count, &def_string);
```

XmFontList Definition as CDE/Motif Applications

As with FontSet definition, the XmFontList resource definition of an application should also include each font of the character sets that the locale supports.

CODE EXAMPLE 4-1 XmnFontList definition for the en_US.UTF-8 locale

```
*fontList:\n-dt-interface system-medium-r-normal-s*-*-*-*-*-*-*iso8859-1;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-2;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-4;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-5;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-7;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-8;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-9;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*iso8859-15;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*big5-1;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*gb2312.1980-0;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*jisx0201.1976-0;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*jisx0208.1983-0;\n-dt-interface system-medium-r-normal-s*utf-*-*-*-*-*-*-*tis620.2533-0:
```

For more details on the XmFontList and the XmnFontList, refer to the *XmFontList(3X)* man page, *OSF/Motif Programmer's Guide* and the resource section of each Motif widget in the *OSF/Motif Programmer's Reference Manual*.

Installation

The Solaris 7 product allows you to install more than one locale on a machine. This allows the developer to test different locales or to work in different locales for different projects. This chapter describes how to add additional locales on the machine.

Adding Packages

This section describes how to install packages with the `pkgadd` command.

▼ How to Add Packages to a Standalone System

1. **Log in as root.**
2. **Remove any packages with the same name as the ones you are adding.**

This ensures that the system keeps a proper record of software that has been added and removed. There may be times when you want to maintain multiple versions of the same application on the system. For strategies on how to do this, see “Guidelines for Removing Packages,” and for task information, see “How to Remove a Package.” Both of these can be found in the *System Administration Guide*
3. **Add one or more software packages to the system.**

```
# pkgadd -a admin-file -d device-name pkgid...
```

In this command,

<code>-a admin-file</code>	(Optional) Specifies an administration file that <code>pkgadd</code> should consult during the installation. (For details about using an administration file, see the <i>System Administration Guide</i> .)
<code>-d device-name</code>	Specifies the absolute path to the software packages. <i>Device-name</i> can be a path to a device, a directory, or a spool directory. If you do not specify the path where the package resides, the <code>pkgadd</code> command checks the default spool directory (<code>/var/spool/pkg</code>). If the package is not there, the package installation fails.
<code>pkgid</code>	(Optional) Is the name of one or more packages (separated by spaces) to be installed. If spaces are omitted, the <code>pkgadd</code> command installs all available packages.

If `pkgadd` encounters a problem during installation of the package, it displays a message related to the problem, followed by this prompt:

```
Do you want to continue with this installation?
```

Respond with `yes`, `no`, or `quit`. If more than one package has been specified, type `no` to stop the installation of the package being installed. `pkgadd` continues to install the other packages. Type `quit` to stop the installation.

4. Verify that the package has been installed successfully, using the `pkgchk` command.

```
# pkgchk -v pkgid
```

If `pkgchk` determines there are no errors, it returns a list of installed files. Otherwise, it reports the error.

Installing Software From a Mounted CD

The following example shows a command to install the `SUNWaudio` package from a CD mounted on the Solaris 2.6 operating environment or compatible versions. The example also shows use of the `pkgchk` command to verify that the package files were installed properly.

```
# pkgadd -d /cdrom/cdrom0/s0/Solaris_2.7/Product SUNWaudio
.
.
.
Installation of SUNWaudio> complete.
```

(continued)

(Continuation)

```
# pkgchk -v SUNWaudio
/usr
/usr/bin
/usr/bin/audioconvert
/usr/bin/audioplay
/usr/bin/audiorecord
```

Installing Software From a Remote Package Server

If the packages you want to install are available from a remote system, you can mount the directory containing the packages (in package format) manually and install packages on the local system. The following example shows the commands to do this. In this example, assume the remote system named `package-server` has software packages in the `/latest-packages` directory. The mount command mounts the packages locally on `/mnt`, and the `pkgadd` command installs the `SUNWaudio` package.

```
# mount -F nfs -o ro package-server:/latest-packages /mnt
# pkgadd -d /mnt SUNWaudio
.
.
.
Installation of SUNWaudio> was successful.
```

If the automounter is running at your site, you do not need to mount the remote package server manually. Instead, use the automounter path (in this case, `/net/package-server/latest-packages`) as the argument to the `-d` option.

```
# pkgadd -d /net/package-server/latest-packages SUNWaudio
.
.
.
Installation of SUNWaudio> was successful.
```

The following example is similar to the previous one, except it uses the `-a` option and specifies an administration file named `noask-pkgadd`. In this example, assume the `noask-pkgadd` administration file is in the default location, `/var/sadm/install/admin`.

```
# pkgadd -a noask-pkgadd -d /net/package-server/latest-packages SUNWaudio
.  
.  
.  
Installation of SUNWaudio> was successful.
```

Installing the Localization Product

Table 5-1 contains the list of common packages for the operating system localization and the window system localization.

European Packages

TABLE 5-1 Pan-European Files for Localization and Windowing

Locale	OS Common Packages	Win Common Packages
All Euro	SUNWploc	SUNWplow
	SUNWploc1	SUNWplow1
	SUNWenise	SUNWpldte
	SUNWeuise	

French Files

TABLE 5-2 French Files for Localization and Windowing

Locale	OS Common Packages	Win Common Packages	OS Packages	Desktop Packages
fr			SUNwfros	SUNwfoaud SUNwfobk SUNwfodcv SUNwfodem SUNwfordst SUNwfordte SUNwfoimt SUNwforte SUNwfrbas SUNwfrdst SUNwfrdte SUNwfrhe SUNwfrhed SUNwfrim SUNwfris SUNwfrwm SUNwftltk SUNfwacx SUNwfxplt

German Files

TABLE 5-3 German Files for Localization and Windowing

Locale	OS Common Packages	Win Common Packages	OS Packages	Desktop Packages
de			SUNWdeos	SUNWdoaud SUNWdobk SUNWdodcv SUNWdodem SUNWdodst SUNWdodte SUNWdoimt SUNWdorte SUNWdebas SUNWdedst SUNWdedte SUNWdehe SUNWdehed SUNWdeim SUNWdeis SUNWdewm SUNWdtltk SUNWdwacx SUNWdxplt

Italian Files

TABLE 5-4 Italian Files for Localization and Windowing

Locale	OS Common Packages	Win Common Packages	OS Packages	Desktop Packages
it			SUNWitos	SUNWioaud SUNWiobk SUNWiodcv SUNWiodem SUNWiodst SUNWiodte SUNWioimt SUNWiorte SUNWitbas SUNWitdst SUNWitdte SUNWithe SUNWithed SUNWitim SUNWitis SUNWitwm SUNWitltk SUNWiwacx SUNWixplt

Spanish Files

TABLE 5-5 Spanish Files for Localization and Windowing

Locale	OS Common Packages	Win Common Packages	OS Packages	Desktop Packages
es			SUNWesos	SUNWeoaud SUNWeobk SUNWeodcv SUNWeodem SUNWeodst SUNWeodte SUNWeoimt SUNWeorte SUNWesbas SUNWesdst SUNWesdte SUNWeshe SUNWeshed SUNWesim SUNWesis SUNWeswm SUNWetltk SUNWewacx SUNWexplt

Swedish Files

TABLE 5-6 Swedish Files for Localization and Windowing

Locale	OS Common Packages	Win Common Packages	OS Packages	Desktop Packages
sv			SUNWsvos	SUNWsoaud SUNWsobk SUNWsodcv SUNWsodem SUNWsodst SUNWsodte SUNWsoimt SUNWsorte SUNWsvbas SUNWsvdst SUNWsvdte SUNWsvhe SUNWsvhed SUNWsvim SUNWsvis SUNWsvwm SUNWstltk SUNWswacx SUNWsxplt

Detailed Descriptions of European Files

TABLE 5-7 European Package Descriptions

Package Name	Package Description
SUNWerdm	European OILBN ReadMe Directory
SUNWi1of	ISO-8859-1 (Latin-1) Optional Fonts
SUNWi1of	ISO-8859-1 (Latin-1) Optional Fonts
SUNWi2of	X11 fonts for ISO-8859-2 character set (optional fonts)
SUNWi2rf	X11 fonts for ISO-8859-2 character set (required fonts)
SUNWi4of	X11 fonts for ISO-8859-4 character set (optional fonts)
SUNWi4rf	X11 fonts for ISO-8859-4 character set (required fonts)
SUNWi5of	X11 fonts for ISO-8859-5 character set (optional fonts)
SUNWi5rf	X11 fonts for ISO-8859-5 character set (required fonts)
SUNWi7of	X11 fonts for ISO-8859-7 character set (optional fonts)
SUNWi7rf	X11 fonts for ISO-8859-7 character set (required fonts)
SUNWi9of	X11 fonts for ISO-8859-9 character set (optional fonts)
SUNWi9rf	X11 fonts for ISO-8859-9 character set (required fonts)
SUNWioaud	Italian OPEN LOOK Audio applications
SUNWiobk	Italian OpenWindows online handbooks
SUNWiodcv	Italian OPEN LOOK document and help viewer applications
SUNWiodem	Italian OPEN LOOK demo programs
SUNWiodst	Italian OPEN LOOK deskset tools

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWiodte	Italian OPEN LOOK desktop environment
SUNWioimt	Italian OPEN LOOK imagetool
SUNWiorte	Italian OPEN LOOK toolkits runtime environment
SUNWislcc	XSH4 conversion for Eastern European locales
SUNWisolc	XSH4 conversion for ISO Latin character sets
SUNWitbas	Base L10N it CDE functionality to run a CDE application
SUNWitdst	Italian CDE Desktop Applications messages
SUNWitdte	Italian CDE Desktop Environment
SUNWithe	Italian CDE Help Runtime Environment
SUNWithed	Italian CDE Help Developer Environment
SUNWithev	Italian CDE Online Help
SUNWitim	Italian CDE Imageviewer
SUNWitis	Italian install software localization
SUNWitltk	Italian ToolTalk binaries and shared libraries
SUNWitos	Italian OS localization
SUNWitpmw	Italian (EUC) Localizations for Power Management OW Utilities
SUNWitreg	Italian Solaris User Registration prompts at desktop login for user registration
SUNWitwm	Italian CDE Desktop Window Manages Messages
SUNWiwacx	Italian OPEN LOOK AccessX

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWiwbcp	Italian OpenWindows Binary Compatibility Package
SUNWixplt	Italian X Windows platform software
SUNWeoaud	Spanish OPEN LOOK Audio applications
SUNWeobk	Spanish OpenWindows online handbooks
SUNWeodcv	Spanish OPEN LOOK document and help viewer applications
SUNWeodem	Spanish OPEN LOOK demo programs
SUNWeodst	Spanish OPEN LOOK deskset tools
SUNWeodte	Spanish OPEN LOOK desktop environment
SUNWeoimt	Spanish OPEN LOOK imagetool
SUNWeorte	Spanish OPEN LOOK toolkits runtime environment
SUNWesbas	Base L10N fr CDE functionality to run a CDE application
SUNWesdst	Spanish CDE Desktop Applications
SUNWesdte	Spanish CDE Desktop Environment
SUNWeshe	Spanish CDE Help Runtime Environment
SUNWeshed	Spanish CDE Help Developer Environment
SUNWeshev	Spanish CDE Online Help
SUNWesim	Spanish CDE Desktop apps
SUNWesis	Spanish install software localization
SUNWesos	Spanish OS localization

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWespmw	Spanish (EUC) Localizations for Power Management OW Utilities
SUNWesreg	Solaris User Registration prompts at desktop login for user registration
SUNWeswm	Spanish CDE Desktop window manages messages
SUNWetltk	Spanish ToolTalk binaries and shared libraries
SUNWenise	English partial locales enabling during install
SUNWeuise	European partial locales enabling during install
SUNWewacx	Spanish OPEN LOOK AccessX
SUNWexplt	Spanish X Windows platform software
SUNWfbcp	French OS Binary Compatibility Package
SUNWfoaud	French OPEN LOOK Audio applications
SUNWfobk	French OpenWindows online handbooks
SUNWfodcv	French OPEN LOOK document and help viewer applications
SUNWfodem	French OPEN LOOK demo programs
SUNWfodst	French OPEN LOOK deskset tools
SUNWfodte	French OPEN LOOK desktop environment
SUNWfoimt	French OPEN LOOK imagetool
SUNWforte	French OPEN LOOK toolkits runtime environment
SUNWfrbas	Base L10N fr CDE functionality to run a CDE application
SUNWfrdst	French CDE Desktop Applications

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWfrdte	French CDE Desktop Environment
SUNWfrhe	French CDE Help Runtime Environment
SUNWfrhed	French CDE Help Developer Environment
SUNWfrhev	French CDE Online Help
SUNWfrim	French CDE ImageViewer
SUNWfris	French install software localization
SUNWfros	French OS localization
SUNWfrpmw	French (EUC) Localizations for Power Management OW Utilities
SUNWfrwm	French CDE Desktop Window Manages Messages
SUNWftltk	French ToolTalk binaries and shared libraries
SUNWfwacx	French OPEN LOOK AccessX
SUNWfwbcp	French OpenWindows Binary Compatibility Package
SUNWfxplt	French X Windows platform software
SUNWf8bas	Base L10N fr CDE functionality to run a CDE application
SUNWf8dst	CDE Desktop Applications
SUNWf8dte	CDE Desktop Environment
lSUNWf8he	CDE Help L10N fr Runtime Environment
SUNWf8im	CDE Desktop Applications
SUNWf8wm	French UTF-8 CDE Desktop Window Manages Messages

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWd8bas	Base L10N German UTF-8 CDE functionality to run a CDE application
SUNWd8dst	CDE Desktop Applications
SUNWd8dte	CDE Desktop Login Environment
SUNWd8he	CDE Help L10N German UTF-8 Runtime Environment
SUNWd8im	CDE Desktop Applications
SUNWd8wm	German UTF-8 CDE Desktop Window Manages Messages
SUNWdbcp	German OS Binary Compatibility Package
SUNWdebas	Base L10N German CDE functionality to run a CDE application
SUNWe8bas	Base L10N Spanish CDE functionality to run a CDE application
SUNWe8dst	CDE Desktop Applications
SUNWe8dte	CDE Desktop Login Environment
SUNWe8he	CDE Help L10N es Runtime Environment
SUNWe8im	CDE Desktop applications
SUNWe8wm	Spanish UTF-8 CDE Desktop Window Manages Messages
SUNWsoaud	Swedish OPEN LOOK Audio applications
SUNWsobk	Swedish OpenWindows online handbooks
SUNWsodcv	Swedish OPEN LOOK document and help viewer applications
SUNWsodem	Swedish OPEN LOOK demo programs
SUNWsodst	Swedish OPEN LOOK deskset tools

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWsdte	Swedish OPEN LOOK desktop environment
SUNWsoimt	Swedish OPEN LOOK imagetool
SUNWsorte	Swedish OPEN LOOK toolkits runtime environment
SUNWstltk	Swedish ToolTalk binaries and shared libraries
SUNWsvbas	Base Swedish CDE functionality messages
SUNWsvdst	Swedish CDE Desktop Applications messages
SUNWsvdte	Swedish CDE Desktop Environment messages
SUNWsvhe	Swedish CDE Help Runtime Environment
SUNWsvhed	Swedish CDE Help Developer Environment messages
SUNWsvhev	Swedish CDE Online Help
SUNWsvim	Swedish CDE Image editor messages
SUNWsvis	Swedish install software localization
SUNWsvos	Swedish OS localization
SUNWsvpmw	Swedish (EUC) Localizations for Power Management OW Utilities
SUNWsvreg	Swedish Solaris User Registration prompts at desktop login for user registration
SUNWsvwm	Swedish CDE Desktop Window Manages Messages
SUNWswacx	Swedish OPEN LOOK AccessX
SUNWsxplt	Swedish X Windows platform software
SUNWdbcp	German OS Binary Compatibility Package

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWdebas	Base L10N German CDE functionality to run a CDE application
SUNWdedst	German CDE Desktop Applications
SUNWdedte	German CDE Desktop Login Environment
SUNWdehe	German CDE Help Runtime Environment
SUNWdehed	German CDE Help Developer Environment
SUNWdehev	German CDE Online Help
SUNWdeim	German CDE Imageviewer
SUNWdeis	German install software localization
SUNWdeos	German message files for the OS-Networking consolidation
SUNWdepmw	German (EUC) Localizations for Power Management OW Utilities
SUNWdereg	German Solaris User Registration prompts at desktop login for user registration
SUNWdewm	German CDE Desktop Window Manages Messages
SUNWdoaud	German OPEN LOOK Audio applications
SUNWdobk	German OpenWindows online handbooks
SUNWdodcv	German OPEN LOOK document and help viewer applications
SUNWdodem	German OPEN LOOK demo programs
SUNWdodst	German OPEN LOOK deskset tools
SUNWdodte	German OPEN LOOK desktop environment
SUNWdoimt	German OPEN LOOK imagetool

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWdorte	German OPEN LOOK toolkits runtime environment
SUNWdwacx	German OPEN LOOK AccessX
SUNWdwbcp	German OpenWindows Binary Compatibility Package
SUNWpldte	CDE Eastern European locale support
SUNWploc	European Partial Locales
SUNWploc1	Supplementary Partial Locales
SUNWplow	OpenWindows enabling for Partial Locales
SUNWplow1	OpenWindows enabling for Supplementary Partial Locales
SUNWfrreg	Localized e-reg software messages in the End-User cluster and above
SUNWitreg	
SUNWsvreg	
SUNWesreg	
SUNWdereg	

TABLE 5-7 European Package Descriptions *(continued)*

Package Name	Package Description
SUNWfrpmw	Localized Power Management software in the End-User cluster and above
SUNWitpmw	
SUNWsvpmw	
SUNWespmw	
SUNWdepmw	
SUNWfwbcp	Localized Binary Compatibility Packages
SUNWiwbcp	
SUNWswbcp	
SUNWewbcp	
SUNWdwbc	

European Codesets

In the Solaris 7 product, several fonts display characters that are encoded in the following codesets:

- Latin-1
- Latin-2
- Latin-4
- Cyrillic
- Greek
- Latin-5

European Font Packages

There are a number of font packages in the Solaris 7 product, as shown in Table 5-8

TABLE 5-8 Font Packages in the Solaris 7 Product

Font Package	Description
SUNWi2of	Latin-2 Optional fonts
SUNWi2rf	Latin-2 Required fonts
SUNWi4of	Latin-4 Optional fonts
SUNWi4rf	Latin-4 Required fonts
SUNWi5of	Cyrillic Optional fonts
SUNWi5rf	Cyrillic Required fonts
SUNWi7of	Greek Optional fonts
SUNWi7rf	Greek Required fonts
SUNWi9of	Latin-5 Optional fonts
SUNWi9rf	Latin-5 Required fonts

- All required font packages are in the developer cluster.
- All fonts (both required and optional) are in the entire cluster.

Asian Packages

The remainder of this chapter covers the Asian packages. Table 5-9 is shown below.

TABLE 5-9 Common Asian Packages for Localization and Windowing

OS Common Packages	Win dow Packages	64-bit OS Packages	64-bit Windowing Packages
SUNWale	SUNWxi18n	SUNWalex	
SUNWaled	SUNWxim		

TABLE 5-10 Korean Packages for Localization and Windowing

Locale	OS Packages	Windowing Packages	64-bit OS Packages	64-bit Windowing Packages
ko	SUNWkler	SUNWkoaud	SUNWklerx	
	SUNWkleu	SUNWkodcv	SUNWkleux	
	SUNWkbcp	SUNWkodem		
		SUNWkadis		
		SUNWkadma		
		SUNWsadl		
		SUNWkervl		
		SUNWkoimt		
		SUNWkxfnt		
		SUNWkexir		
		SUNWkoman		
		SUNWkxman		
		SUNWkkcsr		
		SUNWkodst		
		SUNWkorte		
		SUNWkxoft		
		SUNWkepmw		
		SUNWkodte		
		SUNWkltl		
		SUNWkxplt		
	SUNWkwbcp			

TABLE 5-10 Korean Packages for Localization and Windowing *(continued)*

Locale	OS Packages	Windowing Packages	64-bit OS Packages	64-bit Windowing Packages
ko.UTF-8	SUNWkiu8		SUNWkiu8x	
	SUNWkuleu		SUNWkulex	
			SUNWkuadm	
			SUNWkusal	
			SUNWkuadi	
			SUNWkcoft	
			SUNWkupmw	
			SUNWkuxpl	
			SUNWkuodf	
			SUNWkuxft	

TABLE 5-11 Simplified Chinese Packages for Localization and Windowing

Locale	OS Packages	Windowing Packages	64-bit OS Packages	64-bit Windowing Packages
zh	SUNWcleu	SUNWcadis	SUNWciu8x	
	SUNWcler	SUNWcsadl	SUNWcleux	
	SUNWciu8	SUNWcadma		
	SUNWcbcp	SUNWcervl		
		SUNWcodcv		
		SUNWcoimt		
		SUNWcttf		
		SUNWcxplt		
		SUNWcexir		
		SUNWcodem		
		SUNWcoman		
		SUNWcxfont		
		SUNWckcsr		
		SUNWcodst		
		SUNWcorte		
		SUNWcxman		
		SUNWcepmw		
		SUNWcoaud		
		SUNWcodte		
		SUNWcltk		
	SUNWcxoft			

TABLE 5-11 Simplified Chinese Packages for Localization and Windowing *(continued)*

Locale	OS Packages	Windowing Packages	64-bit OS Packages	64-bit Windowing Packages
		SUNWcttf		
		SUNWcwbc		
		SUNWcxmft		
		SUNWgleux		
		SUNWgxplx		
zh.GBK	SUNWgleu	SUNWgxplx		
		SUNWgxfnt		
		SUNWgxman		
		SUNWgxplt		
		SUNWgadis		
		SUNWgodte		
		SUNWgadma		
		SUNWgpmw		
		SUNWgsadl		
		SUNWgttf		

TABLE 5-12 Traditional Chinese Packages for Localization and Windowing

Locale	OS Packages	Windowing Packages	64-bit OS Packages	64-bit Windowing Packages
zh_TW	SUNWhler	SUNWhadma	SUNWhiu8x	
	SUNWhleu	SUNWhsadl	SUNWhlerx	
	SUNWhiu8	SUNWhuccd	SUNWhleux	
	SUNWhbcp	SUNWhadis		
		SUNWhepmw		
		SUNWhoaud		
		SUNWhodte		
		SUNWhlttk		
		SUNWhxoft		
		SUNWhervl		
		SUNWhodcv		
		SUNWhoimt		
		SUNWhttf		
		SUNWhxplt		
		SUNWhexir		
	SUNWhodem			
	SUNWhoman			

TABLE 5-12 Traditional Chinese Packages for Localization and Windowing *(continued)*

Locale	OS Packages	Windowing Packages	64-bit OS Packages	64-bit Windowing Packages
		SUNWhxfnt		
		SUNWhkcsr		
		SUNWhodst		
		SUNWhorte		
		SUNWhwbcpl		
		SUNWhxman		
zh_TW.BIG5				
	SUNW5leu	SUNW5adi	SUNW5leux	SUNW5xplx
		SUNW5adma		
		SUNW5ttf		
		SUNW5sadm		
		SUNW5odte		
		SUNW5xfnt		
		SUNW5xpl t		
		SUNW5pmw		
		SUNW5xmft		

TABLE 5-13 Simplified Chinese Packages

Package Name	Package Description
SUNWcadis	Simplified Chinese (EUC) Localizations for admintool and GUI install
SUNWcadma	Simplified Chinese (EUC) Localizations for Software used to perform system administration tasks

TABLE 5-13 Simplified Chinese Packages *(continued)*

Package Name	Package Description
SUNWcbcp	Simplified Chinese (EUC) Language Environment binary compatibility files.
SUNWcdhj	Simplified Chinese (EUC) Localizations for HotJava Browser for Solaris
SUNWcepmw	Simplified Chinese (EUC) Localization for Power Management OW Utilities
SUNWcervl	Simplified Chinese (EUC) SunVideo Runtime Support Software
SUNWcexir	Simplified Chinese (EUC) XIL Runtime Environment
SUNWckcsr	Simplified Chinese (EUC) KCMS Runtime Environment
SUNWcleu	Simplified Chinese (EUC) Language Environment specific files
SUNWcoaud	Simplified Chinese (EUC) OPENLOOK Audio Applications Package
SUNWcodcv	Simplified Chinese (EUC) OPENLOOK Document and Help Viewer Applications Package
SUNWcodem	Simplified Chinese (EUC) OPENLOOK Demo Programs Package
SUNWcodst	Simplified Chinese (EUC) OPENLOOK Deskset Tools Package
SUNWcodte	Simplified Chinese (EUC) Core OPENLOOK Desktop Package
SUNWcoimt	Simplified Chinese (EUC) OPENLOOK Imagetool Package
SUNWcoman	Simplified Chinese (EUC) OPENLOOK Toolkit/Desktop Users Man Pages Package
SUNWcorte	Simplified Chinese OPENLOOK Toolkits Runtime Environment Package
SUNWcreg	Simplified Chinese Localizations for Solaris User Registration
SUNWcsadl	Simplified Chinese (EUC) Localizations for Solstice Admintool launcher and associated libraries
SUNWctltk	Simplified Chinese T(EUC) oolTalk Runtime Package

TABLE 5-13 Simplified Chinese Packages *(continued)*

Package Name	Package Description
SUNWcxfnt	Simplified Chinese (EUC) X Windows Platform Required Fonts
SUNWcxman	Simplified Chinese (EUC) X Windows Online User Man Pages Package
SUNWcxplt	Simplified Chinese (EUC) X Windows Platform Software Package
SUNWgadis	Simplified Chinese (zh.GBK) Localizations for admintool and GUI install
SUNWgadma	Simplified Chinese (GBK) Localizations for Software used to perform system administration tasks
SUNWgdhj	Simplified Chinese (GBK) Localizations for HotJava Browser for Solaris
SUNWgleu	Simplified Chinese (GBK) Language Environment specific file
SUNWgodte	Simplified Chinese (GBK) Core OPENLOOK Desktop Package
SUNWgpmw	Simplified Chinese (GBK) Localization for Power Management OW Utilities
SUNWgsadl	Simplified Chinese (GBK) Localizations for Solstice Admintool launcher and associated libraries
SUNWgttf	Simplified Chinese (GBK) True Type Fonts
SUNWgxfont	Simplified Chinese (GBK) X Windows Platform required Fonts
SUNWgxman	Simplified Chinese (GBK) X Windows Online User Man Pages Package
SUNWgxplt	Simplified Chinese (GBK) X Windows Platform Software Package
SUNWcxplt	Simplified Chinese (GBK) X Windows Platform Software Package
SUNWciu8x	Simplified Chinese (EUC) icon modules for UTF-8
SUNWcleux	Simplified Chinese (EUC) Language Environment specific files

TABLE 5-14 Japanese Packages for Localization and Windowing

Locale	OS Common Packages	Win Common Packages	OS Packages	Desktop Packages
ja/ja_JP.PCK common			SUNWjfp SUNWjfp SUNWjc0d SUNWjc0r SUNWjc0u SUNWjwncr SUNWjwncu SUNWjwnsr SUNWjwnsu SUNWjiu8 SUNWjman SUNWjxf3 SUNWjxfa SUNWxgljf	SUNWJSat8xw SUNWjc0w SUNWjwncx SUNWjwndt SUNWjreg SUNWjxcft SUNWjxfnt SUNWjxoft SUNWjfxmn SUNWjbf SUNWjcs3f
ja packages (Japanese)			SUNWjbc SUNWjrdm SUNWjeman SUNWjeudc SUNWjexir SUNWjmfrn SUNWjoaud SUNWjodcv SUNWjodst SUNWjodte SUNWjoimt SUNWjorte SUNWjxgld SUNWjxgle SUNWjtltk SUNWjwbcp SUNWjwbk SUNWjxplt SUNWjkcsr SUNWjourn SUNWjxumn SUNWjxpmn SUNWjervl SUNWjffb SUNWjleo SUNWjodem SUNWjsadl SUNWjsxgl SUNWjwacx SUNWjexfa	SUNWjadis SUNWjadma SUNWjepmw

TABLE 5-14 Japanese Packages for Localization and Windowing *(continued)*

Locale	OS Common Packages	Win Common Packages	OS Packages	Desktop Packages
ja_JP.PCK pkgs (Japanese)			SUNWjpwnu SUNWjprdm SUNWjpman SUNWjpadi SUNWjppmw SUNWjpudc SUNWjpxir SUNWjpmfr SUNWjptlt SUNWjpxge SUNWjpxpl SUNWudct SUNWjpkcs SUNWjptlm SUNWjpxpm SUNWjpxum SUNWjprvl SUNWjpf fb SUNWjpleo SUNWjpsal SUNWjpsxg SUNWjpacx SUNWjpxfa	SUNWjpxgd SUNWjpadm SUNWjpadi

TABLE 5-15 Thai Packages for Localization and Windowing

Locale	OS Packages	Window Packages	64-bit OS Packages	64-bit Win Packages
th_TH	SUNWtiu8 SUNWtleu	SUNWtxplt SUNWtxfnt SUNWtxodt	SUNWtleux SUNWtiu8x	

Description of General Packages

TABLE 5-16 General Packages

Package Name	Package Description
SUNWale SUNWaled	Asian Language Environment Common Files Asian Language Environment Common Man Pages
SUNWxi18n SUNWxim	X Windows Internationalization Common Package X Windows X Input Method Server Package

Description of Korean Packages

TABLE 5-17 Korean Packages

Package Name	Package Description
SUNWkoaud	Korean (EUC) OpenLook Audio Applications Package
SUNWkodcv	Korean (EUC) OpenLook Document and Help Viewer Applications Package
SUNWkodem	Korean (EUC) OpenLook Demo Programs Package
SUNWkodst	Korean (EUC) OpenLook Deskset Tools Package
SUNWkodte	Korean (EUC) Core OpenLook Desktop Package
SUNWkoimt	Korean (EUC) OpenLook Imagetool Package
SUNWkoman	Korean (EUC) OpenLook Toolkit/Desktop Users Man Pages Package
SUNWktltk	Korean (EUC) ToolTalk Runtime Package
SUNWkxman	Korean (EUC) X Windows Online User Man Pages Package
SUNWkxplt	Korean (EUC) X Windows Platform Software Package
SUNWkxfnt	Korean (EUC) X Windows Platform required Font Package
SUNWkwbcpc	Korean OpenWindows Binary Compatibility Package

TABLE 5-17 Korean Packages *(continued)*

Package Name	Package Description
SUNWkepmw	Korean (EUC) Power Management OW Utilities
SUNWkcsr	Korean (EUC) Localizations for Kodak Color Management System Runtime
SUNWkervl	Korean (EUC) Localizations for SunVideo™ Runtime Support Software
SUNWkexir	Korean (EUC) Localizations for XIL Runtime Environment
SUNWkdest	Korean (EUC) Localized Tools
SUNWkiu8	Korean (EUC) UTF-8 iconv modules for UTF-8
SUNWkuleu	Korean UTF-8 Language Environment user files
SUNWkcoft	Korean UTF-8 common optional font package
SUNWkuodf	Korean UTF-8 Core OPENLOOK Desktop Package
SUNWkupmw	Korean UTF-8 Power Management OW Utilities
SUNWkxwft	Korean UTF-8 X Windows Platform Required Fonts
SUNWkuxpl	Korean UTF-8 X Windows Platform Software Package
SUNWklerx	64-bit Korean (Language Environment root files
SUNWkleux	64-bit Korean Language Environment user files
SUNWkiu8x	64-bit Korean UTF-8)iconv modules
SUNWkulex	64-bit Korean (UTF-8) Language Environment user files

Description of Traditional Chinese Packages

TABLE 5-18 Traditional Chinese Packages

Package Name	Package Description
SUNWcbcp	Traditional Chinese (EUC) Language Environment Binary Compatibility Package
SUNWcleu	Traditional Chinese (EUC) Language Environment user files
SUNWcoaud	Traditional Chinese (EUC) OpenLook Audio Applications Package
SUNWcodcv	Traditional Chinese (EUC) OpenLook Doc and Help Viewer Applications Package
SUNWcodem	Traditional Chinese (EUC) OpenLook Demo Programs Package
SUNWcodst	Traditional Chinese (EUC) OpenLook Deskset Tools Package
SUNWcodte	Traditional Chinese (EUC) Core OpenLook Desktop Package
SUNWcoimt	Traditional Chinese (EUC) OpenLook Imagetool Package
SUNWcoman	Traditional Chinese (EUC) OpenLook Toolkit/Desktop Users Man Pages Package
SUNWcorte	Traditional Chinese (EUC) OpenLook Toolkits Runtime Environment Package
SUNWctltk	Traditional Chinese (EUC) ToolTalk Runtime Package
SUNWcwbc	Traditional Chinese (EUC) OpenWindows Binary Compatibility Package
SUNWcxman	Traditional Chinese (EUC) X Windows Online User Man Pages Package
SUNWcxoft	Traditional Chinese (EUC) X Windows Optional Fonts Package
SUNWcxplt	Traditional Chinese X(EUC) Windows Platform Software Package
SUNWcxfont	Traditional Chinese (EUC) X Windows Platform required Font Package
SUNWcepwm	Traditional Chinese (EUC) Power Management OW Utilities
SUNWckcsr	Traditional Chinese (EUC) for Kodak Color Management System Runtime

TABLE 5-18 Traditional Chinese Packages *(continued)*

Package Name	Package Description
SUNWcervl	Traditional Chinese (EUC) Localizations for SunVideo Runtime Support Software
SUNWcexir	Traditional Chinese (EUC) Localizations for XIL Runtime Environment
SUNWhbcp	Traditional Chinese (EUC) Language Environment Binary Compatibility Package
SUNWhler	Traditional Chinese (EUC) Language Environment root files
SUNWhleu	Traditional Chinese (EUC) Language Environment user files
SUNWhsadl	Traditional Chinese (EUC) Localization for Solstice Admintool launcher and associated libraries
SUNWhttf	Traditional Chinese (EUC) True Type Fonts Package
SUNWhadis	Traditional Chinese (EUC) Localization for Admintool and GUI install
SUNWhadma	Traditional Chinese (EUC) Localization for Software used to perform system administration tasks
SUNWhiu8	Traditional Chinese (EUC) iconv modules for UTF-8
SUNWhiu8x	Traditional Chinese (EUC) iconv modules for UTF-8
SUNWhlerx	Traditional Chinese (EUC) language environment streams modules
SUNWhleux	Traditional Chinese (EUC) language environment specific files
SUNWhuccd	Traditional Chinese (EUC) User based Chinese Console Display package
SUNWhoaud	Traditional Chinese (EUC) OpenLook Audio Applications Package
SUNWhodcv	Traditional Chinese (EUC) OpenLook Doc and Help Viewer Applications Package
SUNWhodem	Traditional Chinese (EUC) OpenLook Demo Programs Package
SUNWhodst	Traditional Chinese (EUC) OpenLook Deskset Tools Package

TABLE 5-18 Traditional Chinese Packages *(continued)*

Package Name	Package Description
SUNWhodte	Traditional Chinese (EUC) Core OpenLook Desktop Package
SUNWhoimt	Traditional Chinese (EUC) OpenLook Imagetool Package
SUNWhoman	Traditional Chinese (EUC) OpenLook Toolkit/Desktop Users Man Pages Package
SUNWhorte	Traditional Chinese (EUC) OpenLook Toolkits Runtime Environment Package
SUNWhl1tk	Traditional Chinese (EUC) ToolTalk Runtime Package
SUNWhwbcp	Traditional Chinese (EUC) OpenWindows Binary Compatibility Package
SUNWhxman	Traditional Chinese (EUC) X Windows Online User Man Pages Package
SUNWhxoft	Traditional Chinese (EUC) X Windows Optional Fonts Package
SUNWhxplt	Traditional Chinese (EUC) X Windows Platform Software Package
SUNWhxfnt	Traditional Chinese (EUC) X Windows Platform required Font Package
SUNWhepmw	Traditional Chinese (EUC) Power Management OW Utilities
SUNWhkcsr	Traditional Chinese (EUC) Localize for Kodak Color Management System Runtime
SUNhervl	Traditional Chinese (EUC) Localizations for SunVideo Runtime Support Software
SUNWhexir	Traditional Chinese (EUC) Localizations for XIL Runtime Environment
SUNW51eu	Traditional Chinese BIG5 Language Environment user files
SUNW5odte	Traditional Chinese BIG5 Core OPENLOOK Desktop Package
SUNW5pmw	Traditional Chinese BIG5 Power Management OW Utilities
SUNW5xfnt	Traditional Chinese BIG5 X Windows Platform required Fonts Package
SUNW5xoft	Traditional Chinese BIG5 X Windows Optional Fonts Package

TABLE 5-18 Traditional Chinese Packages *(continued)*

Package Name	Package Description
SUNW5xplt	Traditional Chinese BIG5 X Windows Platform Software Package
SUNWgsadl	GBK Solstice Admintool launcher
SUNWgttf	GBK True Type Fonts
SUNWgxfntr	GBK X Windows Platform Required Fonts Package
SUNWgxman	GBK X Windows Online User Man Pages Package
SUNWgxplt	GBK X Windows Platform Software Package
SUNWgreg	GBK L10N for Solaris User Registration
SUNWgdhj	GBK HotJava Browser for Solaris
SUNWgdezt	GBK) Localizations for Desktop Power Pack Applications
SUNWgdhez	Localizations for Desktop Power Pack Help Volumes
SUNWgleux	GBK) 64 Bits Language Environment user files
SUNWgxplx	GBK) 64-bits X Windows Platform Software Package

Description of zh.GBK Packages

TABLE 5-19 zh.GBK Packages

Package Name	Package Description
SUNWgadis	GBK admintool and install software
SUNWgadma	GBK system administration applications
SUNWgdab	GBK L10N for CDE DTBUILDER
SUNWgdbas	GBK L10N for CDE Base

TABLE 5-19 zh.GBK Packages *(continued)*

Package Name	Package Description
SUNWgdst	GBK L10N for CDE Desktop Applications
SUNWddte	GBK L10N for CDE Desktop Login Environment
SUNWgdft	GBK L10N for CDE Fonts
SUNWgdhe	GBK L10N for CDE Help Runtime
SUNWgdhev	GBK L10N CDE Help Volumes
SUNWgdicn	GBK L10N CDE Icons
SUNWgdim	GBK L10N CDE Desktop Imagetool
SUNWgdwm	GBK L10N CDE desktop Window Manager
SUNWgleu	GBK Language Environment user files
SUNWgodte	GBK Core OPENLOOK Desktop Package
SUNWgpmw	GBK Power Management OW Utilities
SUNWgsadl	GBK Solstice Admintool Launcher
SUNWgttf	GBK True Type Fonts
SUNWgxfont	GBK X Windows Platform Required Fonts Package
SUNWgxman	GBK X Windows Online User Man Pages Package
SUNWgxplt	GBK X Windows Platform Software Package
SUNWgreg	GBK L10N for Solaris User Registration
SUNWgdhj	GBK Hot Java Browser for Solaris
SUNWgdezt	Simplified Chinese (GBK) Localizations for Desktop Power Pack Applications
SUNWgdhez	Simplified Chinese (Common) Localizations for Desktop Power Pack Help Volumes
SUNWgleux	64-bit Chinese (GBK) Language Environment user files
SUNWgxplx	64-bit Chinese/PRC (GBK) X Windows Platform Software Package

Description of Thai Packages

TABLE 5-20 Thai Language Packages

Package Name	Package Description
SUNWtiu8	This package contains Thai UTF-8 iconv modules for UTF-8
SUNWtleu	Thai Language Environment specific files
SUNWxtfnt	Thai X Windows Platform required Fonts Package
SUNWxplt	Thai X Windows Platform Software Package
SUNWtiu8x	Thai 64 Bits UTF-8 iconv modules for UTF-8
SUNWtleux	Thai 64 Bits Language Environment user files
SUNWtxodt	Thai Core OPENLOOK Desktop Package

Description of Japanese Packages

TABLE 5-21 Japanese Packages

Package Name	Package Description
SUNWjfpr	Japanese Feature Package root files
SUNWjfpu	Japanese Feature Package usr files
SUNWjeuc	Japanese (EUC) Feature Package usr files
SUNWjpck	Japanese (PCK) Feature Package usr files
JSat8xw	Japanese Input System - ATOK8
SUNWjc0d	Japanese cs00 user dictionary maintenance tool for CDE Motif
SUNWjc0r	Japanese Kana-Kanji Conversion Server cs00 Root Files

TABLE 5-21 Japanese Packages *(continued)*

Package Name	Package Description
SUNWjc0u	Japanese Kana-Kanji Conversion Server cs00 User Files
SUNWjc0w	Japanese cs00 user dictionary maintenance tool for OPEN LOOK
SUNWjdbas	Japanese CDE base
SUNWjdhev	Japanese CDE HELP VOLUMES
SUNWjdhj	Japanese HotJava Browser for Solaris
SUNWjwncr	Wnn6 Client Root Files
SUNWjwncu	Wnn6 Client Usr Files (common)
SUNWjwncx	Wnn6 Client X Window System Files
SUNWjwndt	Wnn6 Client User Files for CDE
SUNWjwnsr	Wnn6 Server Root Files
SUNWjwnsu	Wnn6 Server Usr Files
SUNWjreg	Japanese Solaris User Registration
SUNWjxcft	Japanese X Window System common (not required) fonts
SUNWjxfnt	Japanese X Window System required fonts
SUNWjadis	Japanese (EUC) admintool and install software
SUNWjadma	Japanese (EUC) System administration applications
SUNWjbcp	Japanese SunOS 4.x Binary Compatibility
SUNWjebas	Japanese (EUC) CDE base
SUNWjddst	Japanese (EUC) CDE DESKTOP APPS

TABLE 5-21 Japanese Packages *(continued)*

Package Name	Package Description
SUNWjddte	Japanese (EUC) CDE DESKTOP LOGIN ENVIRONMENT
SUNWjdhe	Japanese (EUC) CDE HELP RUNTIME
SUNWjehev	Japanese (EUC) CDE HELP VOLUMES
SUNWjdim	Japanese (EUC) Solaris CDE Image Viewer
SUNWjdrme	Japanese (EUC) CDE README FILES
SUNWjdwmm	Japanese (EUC) CDE DESKTOP WINDOW MANAGER
SUNWjepmw	Japanese (EUC) Power Management OW Utilities
SUNWjeudc	Japanese (EUC) User Defined Character tool for Solaris CDE
SUNWjexir	Japanese (EUC) XIL Runtime Environment
SUNWjmfrn	Japanese (EUC) Motif RunTime Kit
SUNWjoaud	Japanese (EUC) OPEN LOOK Audio applications
SUNWjodcv	Japanese(EUC) OPEN LOOK document and help viewer applications
SUNWjodst	Japanese (EUC) OPEN LOOK deskset tools
SUNWjodte	Japanese (EUC) OPEN LOOK Desktop Environment
SUNWjoint	Japanese (EUC) OPEN LOOK imagetool
SUNWjorte	Japanese (EUC) OPEN LOOK toolkits runtime environment
SUNWjrdr	Japanese (EUC) On-Line Open Issues ReadMe
SUNWjxgld	Japanese (EUC) XGL Generic Loadable Libraries
SUNWjpxgd	Japanese (PCK) XGL Generic Loadable Libraries

TABLE 5-21 Japanese Packages *(continued)*

Package Name	Package Description
SUNWjxgle	Japanese (EUC) XGL Runtime Environment
SUNWjtltk	Japanese (EUC) ToolTalk runtime
SUNWjwbcp	Japanese (EUC) OpenWindows binary compatibility
SUNWjwbk	Japanese (EUC) OpenWindows online handbooks
SUNWjxplt	Japanese (EUC) X Window System platform software
SUNWjpadm	Japanese (PCK) System administration applications
SUNWjpadi	Japanese (PCK) admintool and install software
SUNWjpbas	Japanese (PCK) CDE base
SUNWjpdst	Japanese (PCK) CDE DESKTOP APPS
SUNWjpdte	Japanese (PCK) CDE DESKTOP LOGIN ENVIRONMENT
SUNWjphe	Japanese (PCK) CDE HELP RUNTIME
SUNWjphev	Japanese (PCK) CDE HELP VOLUMES
SUNWjpim	Japanese (PCK) Solaris CDE Image Viewer
SUNWjprme	Japanese (PCK) CDE README FILES
SUNWjppwm	Japanese (PCK) CDE DESKTOP WINDOW MANAGER
SUNWjppmw	Japanese (PCK) Power Management OW Utilities
SUNWjpudc	Japanese (PCK) User Defined Character tool for Solaris CDE
SUNWjpwnu	Wnn6 Client Usr Files (PCK)
SUNWjpxir	Japanese (PCK) XIL Runtime Environment

TABLE 5-21 Japanese Packages *(continued)*

Package Name	Package Description
SUNWjpmfr	Japanese (PCK) Motif RunTime Kit
SUNWjprdm	Japanese (PCK) On-Line Open Issues ReadMe
SUNWjptlt	Japanese (PCK) ToolTalk runtime
SUNWjpxge	Japanese (PCK) XGL Runtime Environment
SUNWjpxpl	Japanese (PCK) X Window System platform software
SUNWudct	User Defined Character tool for Solaris CDE environment
SUNWjdab	Japanese CDE DTBUILDER
SUNWjfxmn	Japanese Feature English Man Pages for X Window System
SUNWjiu8	Japanese iconv modules for UTF-8
SUNWjman	Japanese Feature Package Man Pages (English)
SUNWjxoft	Japanese X Window System optional fonts
SUNWjeab	Japanese (EUC) CDE DTBUILDER
SUNWjdhed	Japanese (EUC) CDE HELP DEVELOPER ENVIRONMENT
SUNWjedev	Japanese (EUC) Development Environment Package
SUNWjeman	Japanese (EUC) Feature Package Man Pages
SUNWjkcsr	Japanese (EUC) KCMS Runtime Environment
SUNWjourn	Japanese (EUC) OPEN LOOK toolkit/desktop users Man Pages
SUNWjxumn	Japanese (EUC) X Window System online user Man Pages
SUNWjxpmn	Japanese (EUC) X Window System online programmers Man Pages

TABLE 5-21 Japanese Packages *(continued)*

Package Name	Package Description
SUNWjpab	Japanese (PCK) CDE DTBUILDER
SUNWjphed	Japanese (PCK) CDE HELP DEVELOPER ENVIRONMENT
SUNWjpman	Japanese (PCK) Development Environment Package
SUNWjpkcs	Japanese (PCK) Feature Package Man Pages
SUNWjptlm	Japanese (PCK) ToolTalk Man Pages
SUNWjpxpm	Japanese (PCK) X Window System online programmers Man Pages
SUNWjpxum	Japanese (PCK) X Window System online user Man Pages
SUNWjbdf	Japanese BDF font source
SUNWjcs3f	Japanese JIS X0212 Type1 fonts for printing
SUNWjxf3	Japanese X Window System hinted F3 fonts
SUNWjxfa	Japanese X Window System Font Administrator
SUNWxgljf	Japanese XGL Stroke Font
SUNWjervl	Japanese (EUC) SunVideo Runtime Support Software
SUNWjffb	Japanese (EUC) Creator Graphics (FFB) XGL Support
SUNWjleo	Japanese (EUC) ZX XGL support
SUNWjodem	Japanese (EUC) OPEN LOOK demo programs
SUNWjsadl	Japanese (EUC) Solstice Admintool launch
SUNWjsxgl	Japanese (EUC) SX XGL Support
SUNWjwacx	Japanese (EUC) AccessX client program

TABLE 5-21 Japanese Packages *(continued)*

Package Name	Package Description
SUNWjexfa	Japanese (EUC) X Window System Font Administor
SUNWjprv1	Japanese (PCK) SunVideo Runtime Support Software
SUNWjpf fb	Japanese (PCK) Creator Graphics (FFB) XGL Support
SUNWjpleo	Japanese (PCK) ZX XGL support
SUNWjpsal	Japanese (PCK) Solstice Admintool launcher
SUNWjpsxg	Japanese (PCK) SX XGL Support
SUNWjpacx	Japanese (PCK) AccessX client program
SUNWjpxfa	Japanese (PCK) X Window System Font Administrator

Table 5-22 and Table 5-23 show which Korean files will be installed for each type of installation: core, end user, developer, or the entire installation.

TABLE 5-22 ko Locale

Package Name	Core	End User	Developer	Entire
SUNWale	X	X	X	X
SUNWaled			X	X
SUNWxi18n		X	X	X
SUNWkler	X	X	X	X
SUNWkleu	X	X	X	X
SUNWkbcp		X	X	X
SUNWkoaud		X	X	X

TABLE 5-22 ko Locale *(continued)*

Package Name	Core	End User	Developer	Entire
SUNWkodcv		X	X	X
SUNWkodem				X
SUNWkodst		X	X	X
SUNWkodte		X	X	X
SUNWkoimt		X	X	X
SUNWkoman				X
SUNWkorte		X	X	X
SUNWktltk		X	X	X
SUNWkwbcp		X	X	X
SUNWkxman			X	X
SUNWkxplt			X	X
SUNWkxfnt		X	X	X
SUNWkepmw		X	X	X
SUNWkkcsr		X	X	X
SUNWkervl				X
SUNWkexir		X	X	X
SUNWkdest			X	X

TABLE 5-22 ko Locale (continued)

Package Name	Core	End User	Developer	Entire
SUNWklerx		X	X	X
SUNWkleux		X	X	X

TABLE 5-23 ko.UTF-8 Locale

Package Name	Core	End User	Developer	Entire
SUNWale	X	X	X	X
SUNWaled			X	X
SUNWxi18n		X	X	X
SUNWxim		X	X	X
SUNWkiu8	X	X	X	X
SUNWkuleu	X	X	X	X
SUNWkcoft		X	X	X
SUNWkuodf		X	X	X
SUNWkuxft			X	X
SUNWkupmw		X	X	X
SUNWkuxpl		X	X	X
SUNWkxmft		X		
SUNWkuadi		X	X	X
SUNWkuadn		X	X	X
SUNWkusal		X	X	X

TABLE 5-23 ko.UTF-8 Locale *(continued)*

Package Name	Core	End User	Developer	Entire
SUNWkiu8x	X	X	X	X
SUNWkulex	X	X	X	X

Table 5-24 and Table 5-25 and Table 5-26 shows which Chinese files are installed for each type of installation: core, end user, developer, or the entire installation.

TABLE 5-24 zh Locale

Package Name	Core	End User	Developer	Entire
SUNWaled	X	X	X	X
SUNWxi18n		X	X	X
SUNWxim		X	X	X
SUNWcleu	X	X	X	X
SUNWcbcp		X	X	X
SUNWcwbc		X	X	X
SUNWcoaud		X	X	X
SUNWcodcv		X	X	X
SUNWcodem		X	X	X
SUNWcodst		X	X	X
SUNWcodte		X	X	X
SUNWcoimt		X	X	X

TABLE 5-24 zh Locale (continued)

Package Name	Core	End User	Developer	Entire
SUNWcoman		X	X	X
SUNWctltk		X	X	X
SUNWctltk		X	X	X
SUNWcxman		X	X	X
SUNWcxoft		X	X	X
SUNWcxplt		X	X	X
SUNWcxfmt		X	X	X
SUNWcepmw		X	X	X
SUNWckcsr		X	X	X
SUNWcervl			X	X
SUNWcexir		X	X	X
SUNWttf		X	X	X
SUNWcxmft		X	X	X
SUNWcin8x	X	X	X	X
SUNWcleux	X	X	X	X

TABLE 5-25 zh.GBK Locale

Package Name	Core	End User	Developer	Entire
SUNWgadis		X	X	X
SUNWgadma		X	X	X
SUNWgleu		X	X	X
SUNWgodte		X	X	X
SUNWgpmw		X	X	X
SUNWgsadl		X	X	X
SUNWgttf		X	X	X
SUNWgxfont		X	X	X
SUNWgxman		X	X	X
SUNWgxplt		X	X	X
SUNWgreg		X	X	X
SUNWgdhj		X	X	X
SUNWgdezt		X	X	X
SUNWgdhez		X	X	X
SUNWgleux	X	X	X	X
SUNWgxplx	X	X	X	X

TABLE 5-26 th_TH Locale

Package Name	Core	End User	Developer	Entire
SUNWtiu8	X	X	X	X
SUNWtleu	X	X	X	X
SUNWtxfnt		X	X	X
SUNWtxplt		X	X	X
SUNWtiu8x	X	X	X	X
SUNWtleux	X	X	X	X
SUNWtxodt		X	X	X

TABLE 5-27 zh_TW Locale

Package Name	Core	End User	Developer	Entire
SUNWale	X	X	X	X
SUNWaled			X	X
SUNWxi18n		X	X	X
SUNWxim		X	X	X
SUNWhler	X	X	X	X
SUNWhleu	X	X	X	X
SUNWhbcp		X	X	X
SUNWhuccd	X	X	X	X
SUNWhkccd		X	X	X
SUNWhoaud		X	X	X
SUNWhodcv		X	X	X

TABLE 5-27 zh_TW Locale *(continued)*

Package Name	Core	End User	Developer	Entire
SUNWhodem		X	X	X
SUNWhodst		X	X	X
SUNWhodte		X	X	X
SUNWhoimt		X	X	X
SUNWhoman		X	X	X
SUNWhorte		X	X	X
SUNWhlgtk		X	X	X
SUNWhwbcp		X	X	X
SUNWhxman		X	X	X
SUNWhxplt		X	X	X
SUNWhxfnt		X	X	X
SUNWhepmw		X	X	X
SUNWhkcsr		X	X	X
SUNWhervl		X	X	X
SUNWhexir			X	X
SUNWhiu8x		X	X	X
SUNWhlerx		X	X	X
SUNWhleux		X	X	X

TABLE 5-28 zh_TW.BIG5 Locale

Package Name	Core	End User	Developer	Entire
SUNWale	X	X	X	X
SUNWaled			X	X
SUNWxi18n		X	X	X
SUNWxim		X	X	X
SUNWhleu	X	X	X	X
SUNW5leu	X	X	X	X
SUNW5odte		X	X	X
SUNW5pmw		X	X	X
SUNW5xoft		X	X	X
SUNW5xfnt		X	X	X
SUNW5xplt			X	X
SUNW5mft		X	X	X
SUNW5ttf		X	X	X
SUNW5leux		X	X	X
SUNW5plx		X	X	X

Table 5-29, 5-30 and 5-31 show which Japanese files are installed for each type of installation: core, end user, developer, or the entire installation.

TABLE 5-29 ja/ja_JP.PCK Common Packages

Package Name	Core	End User	Developer	Entire
SUNWjfpr	X	X	X	X
SUNWjfpu	X	X	X	X
JSat8xw		X	X	X
SUNWjpadi		X	X	X
SUNWjpadm		X	X	X
SUNWjc0d		X	X	X
SUNWjc0r		X	X	X
SUNWjc0u		X	X	X
SUNWjc0w		X	X	X
SUNWjwncr		X	X	X
SUNWjwncu		X	X	X
SUNWjwncx		X	X	X
SUNWjwndt		X	X	X
SUNWjwnsr		X	X	X
SUNWjwnsu		X	X	X
SUNWjreg		X	X	X
SUNWjxcft		X	X	X
SUNWjxfnt		X	X	X
SUNWudct		X	X	X

TABLE 5-29 ja/ja_JP.PCK Common Packages *(continued)*

Package Name	Core	End User	Developer	Entire
SUNWjfxmn			X	X
SUNWjiu8			X	X
SUNWjman			X	X
SUNWjxoft			X	X
SUNWjbdf				X
SUNWjcs3f				X
SUNWjxf3				X
SUNWjxfa				X
SUNWxgljf				X

TABLE 5-30 ja Locale

Package Name	Core	End User	Developer	Entire
SUNWjeuc	X	X	X	X
SUNWjepmw		X	X	X
SUNWjeudc		X	X	X
SUNWjewnu		X	X	X
SUNWjexir		X	X	X
SUNWjadis		X	X	X
SUNWjadma		X	X	X

TABLE 5-30 ja Locale (continued)

Package Name	Core	End User	Developer	Entire
SUNWjbcpl		X	X	X
SUNWjmfrn		X	X	X
SUNWjoaud		X	X	X
SUNWjodcv		X	X	X
SUNWjodst		X	X	X
SUNWjodte		X	X	X
SUNWjoint		X	X	X
SUNWjorte		X	X	X
SUNWjrdm		X	X	X
SUNWjtltk		X	X	X
SUNWjwbcp		X	X	X
SUNWjwbk		X	X	X
SUNWjxgld		X	X	X
SUNWjxgle		X	X	X
SUNWjxplt		X	X	X
SUNWjedev			X	X
SUNWjeman			X	X
SUNWjkcsr			X	X
SUNWjourn			X	X

TABLE 5-30 ja Locale (continued)

Package Name	Core	End User	Developer	Entire
SUNWjtlmn			X	X
SUNWjxpmn			X	X
SUNWjxumn			X	X
SUNWjervl				X
SUNWjexfa				X
SUNWjffb				X
SUNWjleo				X
SUNWjodem				X
SUNWjsadl				X
SUNWjsxgl				X
SUNWjwacx				X

TABLE 5-31 ja_JP.PCK Locale

Package Name	Core	End User	Developer	Entire
SUNWjpck	X	X	X	X
SUNWjppmw		X	X	X
SUNWjpudc		X	X	X
SUNWjpwnu		X	X	X
SUNWjpxir		X	X	X

TABLE 5-31 ja_JP.PCK Locale *(continued)*

Package Name	Core	End User	Developer	Entire
SUNWjpmfr		X	X	X
SUNWjprdm		X	X	X
SUNWjptlt		X	X	X
SUNWjpxgd		X	X	X
SUNWjpxge		X	X	X
SUNWjpxpl		X	X	X
SUNWjpman			X	X
SUNWjpkcs			X	X
SUNWjptlm			X	X
SUNWjpxpm			X	X
SUNWjprvl				X
SUNWjpxum			X	X
SUNWjpf fb				X
SUNWjpleo				X
SUNWjpsal				X
SUNWjpsxg				X
SUNWjpxfa				X
SUNWjpacx				X

Table 5-32 lists the CDE localization packages.

TABLE 5-32 CDE Packages

Locale	CDE Packages	CDE Minimum	CDE End User	CDE Developers
zh_TW (Traditional Chinese)	SUNWhdab	SUNWhdbas	SUNWhdwm	SUNWhdab
	SUNWhdbas	SUNWhddte	SUNWhdhe	
	SUNWhddst	SUNWhdicn	SUNWhddst	
	SUNWhddte		SUNWhdhev	
	SUNWhdhe		SUNWhdim	
	SUNWhdhev		SUNWhreg	
	SUNWhdicn			
	SUNWhdim			
	SUNWhdwm			
	SUNWhreg			
zh_TW.BIG5 (Traditional Chinese)	SUNW5dab	SUNW5dbas	SUNW5dwm	SUNW5dab
	SUNW5dbas	SUNW5ddte	SUNW5dhe	
	SUNW5ddst	SUNW5dicn	SUNW5ddst	
	SUNW5ddte		SUNW5dim	
	SUNW5dhe		SUNWhreg	
	SUNW5dicn			

TABLE 5-32 CDE Packages (continued)

Locale	CDE Packages	CDE Minimum	CDE End User	CDE Developers
	SUNW5dim			
	SUNW5dwm			
	SUNWhreg			
	SUNW5dezt			
	SUNW5dft			
	SUNWdhed			
	SUNW5hev			
	SUNW5hez			
zh (Simplified Chinese)	SUNWcdab	SUNWcdbas	SUNWcdwm	SUNWcdab
	SUNWcdbas	SUNWcddte	SUNWcdhe	
	SUNWcddst	SUNWcdicn	SUNWcddst	
	SUNWcddte	SUNWcdft	SUNWcdhev	
	SUNWcdhe		SUNWcdim	
	SUNWcdhev		SUNWcreg	
	SUNWcdicn			
	SUNWcdim			
	SUNWcdwm			
	SUNWcreg			

TABLE 5-32 CDE Packages (continued)

Locale	CDE Packages	CDE Minimum	CDE End User	CDE Developers
	SUNWcdezt			
	SUNWcdft			
	SUNWcdhed			
	SUNWcdhex			
th_TH	SUNWtdbas	SUNWtdbas	SUNWtdde	SUNWtdft
	SUNWtddst		SUNWtddst	
	SUNWdde			
	SUNWdft			
	SUNWtdwm			
zh.GBK (Simplified Chinese)	SUNWgdab	SUNWgdab	SUNWgdab	SUNWgdab
	SUNWgdbas	SUNWgdbas	SUNWgdbas	SUNWgdbas
	SUNWgddte	SUNWgddte	SUNWgddte	SUNWgddte
	SUNWgdhev	SUNWgdhev	SUNWgdhev	SUNWgdhev
	SUNWgdhe	SUNWgdhe	SUNWgdhe	SUNWgdhe
	SUNWgdim	SUNWgdim	SUNWgdim	SUNWgdim
	SUNWgdinc	SUNWgdinc	SUNWgdinc	SUNWgdinc
	SUNWgdwm	SUNWgdwm	SUNWgdwm	SUNWgdwm
	SUNWgddst	SUNWgddst	SUNWgddst	SUNWgddst

TABLE 5-32 CDE Packages *(continued)*

Locale	CDE Packages	CDE Minimum	CDE End User	CDE Developers
	SUNWgdtf	SUNWgdtf	SUNWgdtf	SUNWgdtf
	SUNWgreg	SUNWgreg	SUNWgreg	SUNWgreg
	SUNWgttf	SUNWgttf	SUNWgttf	SUNWgttf
	SUNWgdhj	SUNWgdhj	SUNWgdhj	SUNWgdhj
	SUNWgdezt	SUNWgdezt	SUNWgdezt	SUNWgdezt
ko(Korean)	SUNWkdab	SUNWkdbas	SUNWkdwm	SUNWkdab
	SUNWkdbas	SUNWkddte	SUNWkdhe	
	SUNWkddst	SUNWkdicn	SUNWkddst	
	SUNWkddte	SUNWkdft	SUNWkdhev	
	SUNWkdhe		SUNWkdim	
	SUNWkdhev		SUNWkreg	
	SUNWkdicn			
	SUNWkdim			
	SUNWkdwm			
	SUNWkdft			
	SUNWkreg			
	SUNWdest			
	SUNWdezt			

TABLE 5-32 CDE Packages (continued)

Locale	CDE Packages	CDE Minimum	CDE End User	CDE Developers
	SUNWdhed			
	SUNWdhez			
ko.UTF-8 Korean)	SUNWkudab	SUNWkudbs	SUNWkudwm	SUNWkudab
	SUNWkudbs	SUNWkuddt	SUNWkudhr	
	SUNWkudda	SUNWkudic	SUNWkudda	
	SUNWkuddt	SUNWkudft	SUNWkudhv	
	SUNWkudhr		SUNWkudim	
	SUNWkudhv		SUNWkreg	
	SUNWkudic			
	SUNWkudim			
	SUNWkudwm			
	SUNWkudft			
	SUNWkreg			
	SUNWkudhd			
	SUNWkudhz			
	SUNWkudzt			
ja/ja_JP.PCK common	SUNWjdbas	SUNWjdbas	SUNWjdhev	SUNWjdab
	SUNWjdhev			

TABLE 5-32 CDE Packages *(continued)*

Locale	CDE Packages	CDE Minimum	CDE End User	CDE Developers
	SUNWjdab			
	SUNWjpdte			
ja package	SUNWjebas	SUNWjdbas	SUNWjdhev	SUNWjdab
	SUNWjddte			
	SUNWjddst			
	SUNWjdw			
	SUNWjdhe			
	SUNWjehev			
	SUNWjdim			
	SUNWjdrme			
	SUNWjdhed			
	SUNWjeab	SUNWjphed		
	SUNWgdicn	SUNWgdicn	SUNWgdicn	SUNWgdicn
	SUNWgdwm	SUNWgdwm	SUNWgdwm	SUNWgdwm
	SUNWgddst	SUNWgddst	SUNWgddst	SUNWgddst
	SUNWgdft	SUNWgdft	SUNWgdft	SUNWgdft
	SUNWgdft	SUNWgdft	SUNWgdft	SUNWgdft
	SUNWgreg	SUNWgreg	SUNWgreg	SUNWgreg

TABLE 5-32 CDE Packages (continued)

Locale	CDE Packages	CDE Minimum	CDE End User	CDE Developers
	SUNWgttf	SUNWgttf	SUNWgttf	SUNWgttf
	SUNWgdhj	SUNWgdhj	SUNWgdhj	SUNWgdhj
	SUNWgdezt	SUNWgdezt	SUNWgdezt	SUNWgdezt
	SUNWgdhez	SUNWgdhez	SUNWgdhez	SUNWgdhez

Asian Localization Packages Disk Space

The following tables display how much hard disk space, in Megabytes is required by the various packages.

TABLE 5-33 MB Required for Software Groups (SPARC)

Software Group	ko	zh	zh_TW	ja	ja_JP.PCK	ja and ja_JP.PCK
Core System Support	107	105	109	56	57	57
End User System Support	224	196	190	346	339	354
Developer System Support	405	307	524	617	608	632
Entire Distribution	481	385	726	798	790	813

TABLE 5-34 MB Required for Software Groups (x86)

Software Group	ko	zh	zh_TW	ja	ja_JP.PCK	a and ja_JP.PCK
Core System Support	104	105	109	64	64	64
End User System Support	183	183	217	339	339	347
Developer System Support	356	289	597	598	606	622
Entire Distribution	415	349	765	763	763	778

TABLE 5-35 MB Required for ko and ko plus ko.UTF-8 (SPARC)

Software Group	ko	ko.UTF-8	ko plus ko.UTF-8
core system	131	132	135
end user	621	636	702
developer	909	923	990
entire	966	980	1047
entire + OEM	973	988	1054

TABLE 5-36 MB Required for ko and ko plus UTF-8 (x86)

Software Group	ko	ko.UTF-8	ko plus UTF-8
core system	97	98	101
end user	448	466	528
developer	710	729	791
entire	756	774	836

TABLE 5-37 MB Required for zh_TW and zh_TW.BIG5 (SPARC)

Software Group	zh_TW	zh_TW.BIG5	zh_TW & zh_TW.BIG5
core system	137	174	174
end user	601	573	660
developer	885	857	944
entire	942	914	1001
entire + OEM	953	924	1011

TABLE 5-38 MB Required for zh_TW and zh_TW.BIG5 (x86)

Software Group	zh_TW	zh_TW.BIG5	zh_TW & zh_TW.BIG5
core user	98	137	137
end user	422	400	482
developer	678	656	738
entire	723	701	784

TABLE 5-39 MB Required for zh and zh.GBK (SPARC)

Software Group	zh	zh.GBK	zh & zh.GBK
core system	132	139	139
end user	543	531	575
developer	826	831	874
entire	883	888	931
entire + OEM	894	898	942

TABLE 5-40 MB Required for zh and zh.GBK (x86)

Software Group	zh	zh.GBK	zh & zh.GBK
core system	99	100	104
end user	368	355	399
developer	625	627	671
entire	670	672	716

Internationalization Framework in the Solaris 7 Environment

This chapter discusses several new internationalization features contained in the Solaris 7 environment.

- Codeset independence support
- Locale database
- Process code format (wide character expression)
- `libw` and `libintl`
- `ctype` macros
- `genmsg` utility

This chapter also contains information useful for developing internationalized applications such as:

- Dynamically linked applications
- Solaris 7 internationalized APIs

Codeset Independence Support

Before the release of the Solaris 7 operating system, the Sun OS and the Solaris internationalization framework supported only Extended UNIX Code (EUC) representation. This prevented support of new encodings that didn't fit the EUC model, such as PC-Kanji in Japan, Big-5 in Taiwan and GBK in the People's Republic of China.

Because a large part of the computer market demands non-EUC codeset support, Solaris 7 provides a solid framework to enable both EUC and non-EUC codeset support. This support is called *Codeset Independence*, or CSI.

The goal of CSI is to remove EUC dependencies on specific codesets or encoding methods from Solaris OS libraries and commands. The CSI architecture allows the Solaris operating environment to support any UNIX file system safe encoding. CSI supports a number of new codesets, such as UTF-8, PC-Kanji¹, and Big-5.

The CSI Approach

Codeset Independence allows application and platform software developers to keep their code independent of encoding, such as UTF-8, and also provides the ability to adopt any new encoding without having to modify the source code. This architecture approach differs from Java internationalization in that Java requires applications to be Unicode-dependent and also requires code conversions throughout the application.

Many existing internationalized applications (for example, Motif) automatically inherit CSI support from the underlying system. These applications work in the new locales without modification. OPEN LOOK applications, however, that are XView/OLIT based, don't work in the new locales because XView is codeset-dependent.

CSI is inherently independent from any codesets. However, the following assumptions on file code encodings (codesets) still apply to Solaris 7:

- File code is a superset of ASCII.
 - Unicode (16-bits fixed width) cannot be supported as file code.
- NULL (0x00) is not part of multibyte characters for support of null-terminated multibyte character strings.
- Slash / (0x2f) is not part of multibyte characters for support of the UNIX path names.
- Only stateless file code encodings are supported.

CSI-enabled Commands

Table 6-1 contains CSI-enabled commands in Solaris 7. These commands are marked with CSI capabilities on their man page.

All commands are in the `/usr/bin` directory, unless otherwise noted.

1. Japanese Solaris 2.5.1 supports PC Kanji (also known as Shift-JIS).

TABLE 6-1 CSI-enabled Commands in Solaris 7

/usr/lib/diffh	acctcom	gencat	script
/usr/sbin/accept	apropos	getopt	sdiff
/usr/sbin/reject	batch	getoptcv	settime
/usr/ucb/lpr	bdiff	head	sh
/usr/xpg4/bin/awk	cancel	join	split
/usr/xpg4/bin/cp	cat	jsh	strconf
/usr/xpg4/bin/date	catman	kill	strings
/usr/xpg4/bin/du	chgrp	ksh	sum
/usr/xpg4/bin/ed	chmod	lp	tabs
/usr/xpg4/bin/edit	chown	man	tar
/usr/xpg4/bin/egrep	cmp	mkdir	tee
/usr/xpg4/bin/env	col	msgfmt	touch
/usr/xpg4/bin/ex	comm	news	tty
/usr/xpg4/bin/expr	compress	nroff	uncompress
/usr/xpg4/bin/fgrep	cpio	pack	unexpand
/usr/xpg4/bin/grep	csh	paste	uniq
/usr/xpg4/bin/ln	csplit	pcat	unpack
/usr/xpg4/bin/ls	cut	pg	wc
/usr/xpg4/bin/more	diff	printf	whatis

TABLE 6-1 CSI-enabled Commands in Solaris 7 *(continued)*

<code>/usr/xpg4/bin/mv</code>	<code>diff3</code>	<code>priocntl</code>	<code>write</code>
<code>/usr/xpg4/bin/nice</code>	<code>disable</code>	<code>ps</code>	<code>xargs</code>
<code>/usr/xpg4/bin/nohup</code>	<code>echo</code>	<code>pwd</code>	<code>zcat</code>
<code>/usr/xpg4/bin/od</code>	<code>expand</code>	<code>rcp</code>	
<code>/usr/xpg4/bin/pr</code>	<code>file</code>	<code>red</code>	
<code>/usr/xpg4/bin/rm</code>	<code>fine</code>	<code>remsh</code>	
<code>/usr/xpg4/bin/sed</code>	<code>fold</code>	<code>rksh</code>	
<code>/usr/xpg4/bin/sort</code>	<code>ftp</code>	<code>rmdir</code>	
<code>/usr/xpg4/bin/tail</code>		<code>rsh</code>	
<code>/usr/xpg4/bin/tr</code>			
<code>/usr/xpg4/bin/vedit</code>			
<code>/usr/xpg4/bin/vi</code>			
<code>/usr/xpg4/bin/view</code>			

Solaris 7 CSI-enabled Libraries

Nearly all functions in Solaris 7 `libc` (`/usr/lib/libc.so`) are CSI-enabled. However, the following functions in `libc` are not CSI-enabled because they are EUC dependent functions:

- `csetcol()` `csetlen()` `euccol()`
- `euclen()` `eukscol()` `getwidth()`

The following macros are not CSI-enabled because they are EUC dependent:

- `csetno()` `wcsetno()`

In the Solaris 7 product, `libgen` (`/usr/ccs/lib/libgen.a`) are internationalized, but not CSI enabled.

In the Solaris 7 product, `libcurses` (`/usr/ccs/lib/libcurses.a`) are internationalized, but not CSI enabled.

Locale Database

The locale database format and structure in Solaris 7 have changed from previous Solaris releases. The locale database is private and subject to change in a future release. Therefore, when developing an internationalized application, do not directly access the locale database. Instead, use the Solaris internationalization APIs.

Note - When using Solaris 7, use the locale databases that are included with the Solaris 7 product. Do not use locales from previous Solaris versions.

Process Code Format

The process code format in the Solaris 7 product is private and subject to change in a future release. Therefore, when developing an international application, do not assume the process code format is the same. Instead use the Solaris internationalization APIs that are described in .

Multibyte Support Environment (MSE)

A multibyte character is a character that cannot be stored in a single byte, such as Chinese, Japanese, or Korean characters. These characters require two or three bytes of storage. A more precise definition can be found in ISO/IEC 9899:1990 subclause 3.13. The programming model allows these multibyte characters to be read in as logical units and stored internally as wide characters. These wide characters can be processed by the program as logical entities in their own right. Finally, these wide characters can be written out (undergoing appropriate translation) as logical units. This is analogous to the way single byte characters are read in, manipulated, and written out again. The MSE provides a comparable set of interfaces to perform this

processing. The MSE allows programs to be written to handle multibyte characters using the same programming model that is used for single byte characters.

Dynamically Linked Applications

Solaris 7 users can choose how to link applications with the system libraries, such as `libc`, by using dynamic linking or static linking. However, any application that requires internationalization features in the system libraries must be dynamically linked. If the application has been statically linked, the operation to set the locale to other than C and POSIX using the `setlocale` function will fail. Statically linked applications can be operated only in C and POSIX locales.

By default, the linker program tries to link the application dynamically. If the command line options to the linker and the compiler include `-Bstatic` or `-dn` specifications, your application may be statically linked. You can check whether an existing application is dynamically linked using the `/usr/bin/ldd` command.

For example, if you type:

```
% /usr/bin/ldd /sbin/sh
```

the command displays the following message:

```
% ldd: /sbin/sh: file is not a dynamic executable or shared object
```

The message indicates the `/sbin/sh` command is not a dynamically linked program. Also, if you type:

```
% /usr/bin/ldd /usr/bin/ls
```

the command displays the following message:

```
% libc.so.1 => /usr/lib/libc.so.1
% libdl.so.1 => /usr/lib/libdl.so.1
```

This message indicates the `/usr/bin/ls` command has been dynamically linked with two libraries, `libc.so.1` and `libdl.so.1`.

To summarize, if the message from the `ldd` command to the application does not contain a `libc.so.1` entry, it indicates that the application has been statically linked with `libc`. In that case, you need to change the command line options to the linker so that dynamic linking is used instead, then re-link the application.

libw and libintl

In the Solaris 7 release, the implementation of `libw` and `libintl` has been moved to `libc`. The shared objects `libw.so.1` and `libintl.so.1` are provided as filters on `libc.so.1`, and the archives `libw.a` and `libintl.a` are provided as links to an empty archive.

The shared objects ensure runtime compatibility for existing applications and, together with the archives, provide compilation environment compatibility for building applications. However, it is no longer necessary to build applications against `libw` or `libintl`.

For more information on filters see the *Linker and Libraries Guide*.

Table 6-2 shows the stub entry points in `libw` and `libintl`.

TABLE 6-2 Stub Entry Points in `libw` and `libintl`

<code>libw</code>	<code>fgetwc</code>	<code>fgetws</code>	<code>fputwc</code>	<code>fputws</code>	<code>getwc</code>
	<code>getwchar</code>	<code>getws</code>	<code>isenglish</code>	<code>isideogram</code>	<code>isnumber</code>
	<code>isphonogram</code>	<code>isspecial</code>	<code>iswalnum</code>	<code>iswalpha</code>	<code>iswcntrl</code>
	<code>iswctype</code>	<code>iswdigit</code>	<code>iswgraph</code>	<code>iswlower</code>	<code>iswprint</code>
	<code>iswpunct</code>	<code>iswspace</code>	<code>iswupper</code>	<code>iswxdigit</code>	<code>putwc</code>
	<code>putwchar</code>	<code>putws</code>	<code>strtows</code>	<code>towlower</code>	<code>towupper</code>
	<code>ungetwc</code>	<code>watoll</code>	<code>wscat</code>	<code>wchr</code>	<code>wscmp</code>
	<code>wscoll</code>	<code>wscopy</code>	<code>wscspn</code>	<code>wcsftime</code>	<code>wcslen</code>
	<code>wscncat</code>	<code>wscncmp</code>	<code>wscncpy</code>	<code>wcsprbrk</code>	<code>wcsrchr</code>
	<code>wcsspn</code>	<code>wctod</code>	<code>wcstok</code>	<code>wcstol</code>	<code>wcstoul</code>
	<code>wcswcs</code>	<code>wcswidth</code>	<code>wcsxfrm</code>	<code>wctype</code>	<code>wcwidth</code>
	<code>wscasecmp</code>	<code>wscat</code>	<code>wchr</code>	<code>wscmp</code>	<code>wscol</code>
	<code>wscoll</code>	<code>wscopy</code>	<code>wscspn</code>	<code>wsdup</code>	<code>wslen</code>
	<code>wsncasecmp</code>	<code>wsncat</code>	<code>wsncmp</code>	<code>wsncpy</code>	<code>wsprbrk</code>
	<code>wsprintf</code>	<code>wsrchr</code>	<code>wsscancf</code>	<code>wsspn</code>	<code>wstod</code>
	<code>wstok</code>	<code>wstol</code>	<code>wstoll</code>	<code>wstostr</code>	<code>wsxfrm</code>
<code>libintl</code>	<code>bindtextdomain</code>	<code>dcgettext</code>	<code>dgettext</code>	<code>gettext</code>	<code>textdomain</code>

cctype Macros

Character classification and character transformation macros are defined in `/usr/include/cctype.h`. The Solaris 7 environment provides a new set of `cctype` macros. The new macros support character classification and transformation

semantics defined by XPG4. To access the new set of macros, one of the following conditions must be met:

- `_XPG4_CHAR_CLASS` is defined,
- `_XOPEN_SOURCE` and `_XOPEN_VERSION=4` are defined, or
- `_XOPEN_SOURCE` and `_XOPEN_SOURCE_EXTENDED=1` are defined

This means that all XPG4 and XPG4.2 applications automatically have the new macros. Since `_XOPEN_SOURCE`, `_XOPEN_VERSION`, and `_XOPEN_SOURCE_EXTENDED` bring in extra XPG4 related features in addition to new ctype macros, non-XPG4 or XPG4.2 applications should use `__XPG4_CHAR_CLASS__`.

There are corresponding ctype functions. The Solaris 7 functions also support XPG4 semantics.

Refer to the ctype `ctype(3C)` man page for details.

Internationalization APIs in libc

Solaris 7 offers two sets of APIs:

- Multibyte (file codes)
- Wide characters (process code)

Applications process in wide character codes.

When a program takes input from a file, convert your file's multibyte data into wide character process code with the `mbtowc` and `mbtowcs` APIs. To convert the file output data from wide character format into multibyte format, use the `wcstombs` and `wctomb` APIs.

Table 6-3 shows a list of internationalization APIs included in Solaris 7.

TABLE 6-3 Internationalization APIs in libc

API Type	Library Routine	Description
Messaging functions		
	<code>catclose()</code>	Close a message catalog.
	<code>catgets()</code>	Read a program message.

TABLE 6-3 Internationalization APIs in `libc` (continued)

API Type	Library Routine	Description
	<code>catopen()</code>	Open a message catalog.
	<code>dgettext()</code>	Get a message from a message catalog with domain specified.
	<code>dcgettext()</code>	Get a message from a message catalog with domain and category specified.
	<code>textdomain()</code>	Set and query the current domain.
	<code>bindtextdomain()</code>	Bind the path for a message domain.
Code conversion		
	<code>iconv()</code>	Convert codes.
	<code>iconv_close()</code>	Deallocate the conversion descriptor.
	<code>iconv_open()</code>	Allocate the conversion descriptor.
Regular expression		
	<code>regcomp()</code>	Compile the regular expression.
	<code>regexexec()</code>	Execute the regular expression matching.
	<code>regerror()</code>	Provide a mapping from error codes to error message.
	<code>regfree()</code>	Free memory allocated by <code>regcomp()</code> .
Wide character class		
	<code>wctype()</code>	Define character class.
	<code>wctrans</code>	Define character mapping.
	<code>towctrans</code>	Wide-character mapping.

TABLE 6-3 Internationalization APIs in `libc` (continued)

API Type	Library Routine	Description
	<code>setlocale()</code>	Modify and query a program's locale.
	<code>nl_langinfo()</code>	Get language and cultural information of current locale.
	<code>localeconv()</code>	Get monetary and numeric formatting information of current locale.
Character classification	<code>isalpha()</code>	Is character alphabetic?
	<code>isupper()</code>	Is character uppercase?
	<code>islower()</code>	Is character lowercase?
	<code>isdigit()</code>	Is character a digit?
	<code>isxdigit()</code>	Is character a hex digit?
	<code>isalnum()</code>	Is character alphabetic or digital?
	<code>isspace()</code>	Is character a space?
	<code>ispunct()</code>	Is character a punctuation mark?
	<code>isprint()</code>	Is character printable?
	<code>isctrnl()</code>	Is character a control character?
	<code>isascii()</code>	Is character an ASCII character?
	<code>isgraph()</code>	Is character a visible character?

TABLE 6-3 Internationalization APIs in `libc` (continued)

API Type	Library Routine	Description
	<code>isphonogram()</code>	Is wide character a phonogram?
	<code>isideogram()</code>	Is wide character an ideogram?
	<code>isenglish()</code>	Is wide character in English alphabet from a supplementary codeset?
	<code>isnumber()</code>	Is wide character a digit from a supplementary codeset?
	<code>isspecial()</code>	Is special wide character from a supplementary codeset?
	<code>iswalpha()</code>	Is wide character alphabetic?
	<code>iswupper()</code>	Is wide character uppercase?
	<code>iswlower()</code>	Is wide character lowercase?
	<code>iswdigit()</code>	Is wide character a digit?
	<code>iswxdigit()</code>	Is wide character a hex digit?
	<code>iswalnum()</code>	Is wide character an alphabetic character or digit?
	<code>iswspace()</code>	Is wide character white space?
	<code>iswpunct()</code>	Is wide character a punctuation mark?
	<code>iswprint()</code>	Is wide character a printable character?
	<code>iswgraph()</code>	Is wide character a visible character?
	<code>iswcntrl()</code>	Is wide character a control character?
	<code>iswascii()</code>	Is wide character an ASCII character?
	<code>toupper()</code>	Convert a lowercase character to uppercase.

TABLE 6-3 Internationalization APIs in libc *(continued)*

API Type	Library Routine	Description
	<code>tolower()</code>	Convert an uppercase character to lowercase.
	<code>towupper()</code>	Convert a lowercase wide character to uppercase.
	<code>towlower()</code>	Convert an uppercase wide character to lowercase.
Character collation		
	<code>strcoll()</code>	Collate character strings.
	<code>strxfrm()</code>	Transform character strings for comparison.
	<code>wcscoll()</code>	Collate wide character strings.
	<code>wcsxfrm()</code>	Transform wide character strings for comparison.
Monetary handling		
	<code>strfmon()</code>	Convert monetary value to string representation.
Date and time handling		
	<code>getdate()</code>	Convert user format date and time.
	<code>strftime()</code>	Convert date and time to string representation.
	<code>strptime()</code>	Date and time conversion.
Multibyte handling		
	<code>btowc</code>	Single-byte to wide-character conversion.
	<code>mbrlen()</code>	Get number of bytes in character (restartable).
	<code>mbsinit()</code>	Determine conversion object status .

TABLE 6-3 Internationalization APIs in `libc` (continued)

API Type	Library Routine	Description
	<code>mbtowc()</code>	Convert a character to a wide-character code (restartable).
	<code>mbstowcs()</code>	Convert a character string to a wide-character string (restartable).
Wide characters		
	<code>wcsncat()</code>	Concatenate wide character strings to length <i>n</i> .
	<code>wsdup()</code>	Duplicate wide character string.
	<code>wscmp()</code>	Compare wide character strings.
	<code>wcsncmp()</code>	Compare wide character strings to length <i>n</i> .
	<code>wscopy()</code>	Copy wide character strings.
	<code>wcsncpy()</code>	Copy wide character strings to length <i>n</i> .
	<code>wchr()</code>	Find character in wide character string.
	<code>wchrchr()</code>	Find character in wide character string from right.
	<code>wcslength()</code>	Get length of wide character string.
	<code>wscol()</code>	Return display width of wide character string.
	<code>wcsspn()</code>	Return span of one wide character string in another.
	<code>wscspn()</code>	Return span of one wide character string not in another.
	<code>wcspbrk()</code>	Return pointer to one wide character string in another.
	<code>wcstok()</code>	Move token through wide character string.

TABLE 6-3 Internationalization APIs in `libc` (continued)

API Type	Library Routine	Description
	<code>wcswcs()</code>	Find string in wide character string.
	<code>wcstombs()</code>	Convert wide character string to multibyte string.
	<code>wctomb()</code>	Convert wide character to multibyte character.
	<code>wcwidth()</code>	Determine number of column positions of a wide character.
	<code>wcswidth()</code>	Determine number of column positions of a wide character string.
	<code>wctob</code>	Wide-character to single-byte conversion.
	<code>wrtomb</code>	Convert a wide-character code to a character (restartable).
	<code>wcsrtombs</code>	Interpret wide character string according to format.
Wide formatting		
	<code>wsprintf()</code>	Generate wide character string according to format.
	<code>wsscanf()</code>	Formatted input conversion.
	<code>fwprintf</code>	Print formatted wide-character output.
	<code>fwscanf</code>	Convert formatted wide-character input.
	<code>wprintf</code>	Print formatted wide-character output.
	<code>wscanf</code>	Convert formatted wide-character input.
	<code>swprintf</code>	Print formatted wide-character output.
	<code>swscanf</code>	Convert formatted wide-character input.

TABLE 6-3 Internationalization APIs in `libc` (continued)

API Type	Library Routine	Description
	<code>vfwprintf</code>	Wide-character formatted output of a <code>stdarg</code> argument list.
	<code>vswprintf</code>	Wide-character formatted output of a <code>stdarg</code> argument list.
Wide numbers		
	<code>wcstol()</code>	Convert wide character string to long integer.
	<code>wcstoul()</code>	Convert wide character string to unsigned long integer.
	<code>wcstod()</code>	Convert wide character string to double precision.
Wide strings		
	<code>wscasecmp()</code>	Compare wide character strings, ignore case differences.
	<code>wsncasecmp()</code>	Process code string operations.
	<code>wcsstr</code>	Find a wide-character substring.
	<code>wmemchr</code>	Find a wide-character in memory.
	<code>wmemcmp</code>	Compare wide-characters in memory.
	<code>wmemcpy</code>	Copy wide-characters in memory.
	<code>wmemmove</code>	Copy wide-characters in memory with overlapping areas.
	<code>wmemset</code>	Set wide-characters in memory.
Wide standard I/O		
	<code>fgetwc()</code>	Get multibyte character from stream, convert to wide character.
	<code>getwchar()</code>	Get multibyte character from <code>stdin</code> , convert to wide character.

TABLE 6-3 Internationalization APIs in `libc` (continued)

API Type	Library Routine	Description
	<code>fgetws()</code>	Get multibyte string from stream, convert to wide character.
	<code>getws()</code>	Get multibyte string from <code>stdin</code> , convert to wide character.
	<code>fputwc()</code>	Convert wide character to multibyte character, puts to stream.
	<code>fwide</code>	Set stream orientation.
	<code>putwchar()</code>	Convert wide character to multibyte character, puts to <code>stdin</code> .
	<code>fputws()</code>	Convert wide character to multibyte string, puts to stream.
	<code>putws()</code>	Convert wide character to multibyte string, puts to <code>stdin</code> .
	<code>ungetwc()</code>	Push a wide character back into input stream.

genmsg Utility

The new `genmsg` utility can be used with the `catgets()` family of functions to create internationalized source message catalogs. The utility examines a source program file for calls to functions in `catgets` and builds a source message catalog from the information it finds. For example:

```
% cat example.c
...
/* NOTE: %s is a file name */
printf(catgets(catd, 5, 1, "%s cannot be opened.));
/* NOTE: "Read" is a past participle, not a present
    tense verb */
printf(catgets(catd, 5, 1, "Read"));
...
```

(continued)

(Continuation)

```
% genmsg -c NOTE example.c
The following file(s) have been created.
  new msg file = "example.c.msg"
% cat example.c.msg
$quote "
$set 5
1  "%s cannot be opened"
  /* NOTE: %s is a file name */
2  "Read"
  /* NOTE: "Read" is a past participle, not a present
   tense verb */
```

In the above example, `genmsg` is run on the source file `example.c`, which produces a source message catalog named `example.c.msg`. The `-c` option with the argument `NOTE` causes `genmsg` to include comments in the catalog. If a comment in the source program contains the string specified, the comment appears in the message catalog after the next string extracted from a call to `catgets()`.

You can use `genmsg` to number the messages in a message set automatically.

For more information, see the `genmsg(1)` man page.

Note - The material in this section is used with permission from *Creating Worldwide Software: Solaris International Developer's Guide*, 2nd edition by Bill Tuthill and David A. Smallberg, published by Sun Microsystems Press/Prentice Hall. (c)1997 Sun Microsystems, Inc.

X/DPS

The X Window System has been extended with the X Display PostScript system (often described as X/DPS). It uses application-callable libraries on the client side and corresponding extensions on the X server side.

Internationalization and localization issues using Adobe System's PostScript™ are documented in several books from Adobe Systems, Inc.:

- *PostScript Language Reference Manual, Second Edition*. Adobe Systems Inc., Addison Wesley, 1990.
- *PostScript Language Reference Manual Supplement*. Adobe Systems Inc., December 1994.
- *Programming the Display PostScript System with X*. Adobe Systems Inc., Addison Wesley, 1993.

This set of books is essential for successfully developing PostScript applications.

The *PostScript Language Reference Manual (Second Edition)* is the standard reference work for PostScript. It is the definitive documentation of every operator, Display PostScript (DPS), Level 1, and Level 2. The book covers the fundamentals of PostScript as a device-independent printing language. The special capabilities for handling fonts and characters in PostScript are covered. The book's Appendix E also covers standard character sets and encoding vectors. It discusses the organization of fonts that are built into interpreters or supplied from other sources.

Programming the Display PostScript System with X is for application developers who are working with X Windows and Display PostScript. The book documents how to write applications that use Display PostScript to produce information for the screen display and the printer output. It describes coding techniques in detail.

Localization Resource Category

The `localization` resource category specifies which natural language (for example, English or Japanese) is supported. This category is made up of dictionaries that contain the keys `Language`, `Country`, `CharSet`, and others. These keys are in the `%Console%` device parameter set.

```
"<</Language/EN /Country/U.S. /CharSet/ISO-646-ISV>>
```

```
"<</Language/JA /Country null /CharSet/JIS-...>>"
```

In the example with Japanese, the `null` value shows that no dialect was selected for Japanese.

Unique names should be used for each item in the `localization` resource category.

Information on Language Interpreters

Page Description Language (PDL) interpreters can be assigned to a PostScript product. An application or printer driver uses the `PDL` resource category to see which PDL interpreter has been assigned.

Control languages can also be assigned. An application or printer driver can use `ControlLanguage` to see which control languages are available on a PostScript product.

The `PDL` and `ControlLanguage` resource categories are available.

The `PDL` and `ControlLanguage` resource categories are made up of `key/value` pairs. See the Adobe PostScript documentation for more information.

Desktop Environments

The Common Desktop Environment (CDE) is the standard GUI desktop interface for Solaris 7. Not only is it the user's main interface to the system, it is also the interface in which many of the user's locale settings are apparent. The German user sees a German interface; the French user sees a French interface.

The *Common Desktop Environment: Internationalization Programmer's Guide* provides information for internationalizing the desktop to enable applications to support various languages and cultural conventions in a consistent user interface.

Overview

CDE is fully internationalized so that any application can run using any locale that has been installed in the system. By keeping the language- and culture-dependent information separate from the application source code, the application does not need to be rewritten or recompiled to be marketed in different countries. Instead, the external information only has to be localized to match the target language and customs.

The application interface has been standardized to allow functionality in any locale, including East Asia. Solaris 7 complies with the Portable Operating Systems Interface for Computer Environments (POSIX and X/Open specifications, which are also referred to as XPG4.2)

It is important that each layer within the desktop use the proper internationalization interface standards which are described in the following sources:

- X Window System, The Complete Reference to Xlib, Xprotocol, ICCM, XLFD-X Version, Release 5, Digital Press, 1992.

- IEEE Std. 1003.1-1990. Information Technology-Portable Operating System Interface (POSIX)-Part 1: System Application Program Interface (API). ISO/IEC 9945-1:1990.
- OSF Motif 1.2 Programmer' Reference, Revision 1.2, Open Software Foundation, Prentice Hall, 1992.
- X/Open CAE Specification Commands and Utilities, Issue 4, X/Open Company Ltd., 1992.
- *Common Desktop Environment: Programmer's Guide*, Addison Wesley, 1995. The Solaris 7 updated version is supplied online with the CDE AnswerBooks. See "Related Books and Sites" on page xviii for more information.

Locales

Most single-display clients operate in a single locale. This is set with the environment variable, usually `$LANG` or a set of `LC_` environment variables including `$LC_CTYPE`.

The `LC_CTYPE` category of the locale is used by the environment to identify the locale-specific features used at runtime. The fonts and input methods are determined by the `LC_CTYPE` category.

Xtprograms that are enabled for internationalization are expected to call the `XtSetLanguageProc()` function (which calls `setlocale()` by default) to set the locale.

Integrating Fonts

Your application may be used by someone sitting at an X terminal or by someone at a remote workstation across a network. In these situations, the fonts available to the user's X display from the X window server might be different than your application's defaults, and some fonts may not be available.

The standard interface font names defined by CDE are guaranteed to be available on all CDE-compliant systems. These names do not specify actual fonts. Instead, they are aliases that each system vendor maps to its best available fonts. If you use only these font names in your application, you can be sure of getting the closest matching font on any CDE-compliant system.

See *Solaris Common Desktop Environment: Programmer's Guide*, Chapter 2 "Integrating Fonts," and also the CDE man pages `DtStdInterfaceFontNames(5)` and `DtStdAppFontNames(5)` for additional information.

Input Methods

CDE provides the ability to enter localized input for an internationalized application that is using Xm Toolkit. The `XmText[Field]` widgets are enabled to interface with input methods from each locale. Input methods are internationalized because some languages write their text from right-to-left, top-to-bottom, and so forth. Within the same application, you can use several fonts that use different input methods.

The pre-edit area displays the string that is being pre-edited. This can be done in four modes: `OffTheSpot`, `OverTheSpot` (default), `Root`, and `None`. In `OffTheSpot` mode, the location is just below the `MainWindow` area at the right of the status area. In `OverTheSpot` mode, the pre-edit area is at the cursor point. In `Root` mode, the pre-edit and status areas are separate from the client's window.

Internationalization and CDE

Multiple environments may exist within a common open system to support various languages. Each of these is called a locale. A locale specifies the language, fonts, and customs to display data. CDE is fully internationalized so that any application can run in any locale. Any application should be code-set-independent and include support for any multibyte codeset.

All components are shipped as a single, worldwide executable. These support the U.S.A., Europe (Western and Eastern), Japan, Korea, Taiwan, Thailand and China.

Matching Fonts to Character Sets

Various sets of fonts are used to render a locale's characters. The specific font character set depends on the locale. This information should be in a locale specific `app-defaults` file. It will contain *font sets*, *fonts*, and *font lists*.

`XmFontSet` specifies the locale-dependent fonts. The resource name is `*fontSet`. Fonts should not be specifically named. The resource name for `XFontStruct` is `*font`. Font lists contain lists of fonts and font sets. `XFontList` specifies the fonts.

Storage of Localized Text

Text strings in each language should be stored outside of the application and in directories that are identified by locale names. These strings are stored in three types of files: resource files, message catalogs, and private files.

Resource files and message catalogs are both files that deliver text strings. Resource files are compiled when they are loaded and message catalogs are precompiled and ready to be accessed. Any application should be codeset-independent and include support for any multibyte codeset. Private files may be databases of information that include some text strings. Ideally, text strings should be in resource files or message catalogs. If text strings are supplied in a private file, then a tool should also be developed to extract and replace text strings.

Xlib Dependencies

X locale supports one or more of the locales defined by the host environment. Direct Xlib™ conforms to the American National Standards Institute (ANSI) C library and the locale announcement method is the `setlocale()` function. This function configures the locale operation of both the host C library and Xlib. The operation of Xlib is governed by the `LC_CTYPE` category; this is called the current locale. The `XSupportsLocale()` function is used to determine whether the current locale is supported by X.

Message Guidelines

Message guidelines should be developed and used to create a consistent format and style for text. Use clear and simple English so that all users, including those whose command of English is minimal, can understand every message. The book *Common Desktop Environment: Internationalization Programmer's Guide* ends with a number of guidelines for producing clear, concise, translatable messages. Messages should explain the problem and suggest how to perform the action successfully. Comments to the translators should also be included that explain concepts, variables, and so forth. The book includes several lists of suggestions for the format style of the message catalogs and the style of the messages themselves.

Before sending out the message catalogs to be translated, it is useful to have the message catalogs translated from English into international English, that is, into a simplified English that can be easily translated into other languages. This speeds up the translation process, reduces the translator queries, and saves costs.

Internationalization and Distributed Networks

This section of the book covers the exchange of information between applications on different hosts. The transfer of data has to consider several parameters:

- The sender's and receiver's codeset
- Whether the protocol is 7-bit or 8-bit
- The type of interchange encoding allowed by the protocol

If the remote host uses the same codeset as the local host, and if the protocol allows 8-bit data, no conversion is needed. If the protocol allows only 7-bit data, the 8-bit code points must be mapped onto 7-bit ASCII values. There are various strategies for conversion.

If the remote host's codeset is different from that of the local host, the following two cases may apply. The conversion depends on the specific protocol. If the protocol allows 8-bit data, the protocol will need to specify which side does the conversion. If the protocol allows only 7-bit data, a 7-bit interchange encoding is needed along with an identifying character repertoire.

Mail Interchange

With the increased use of the Internet and the ease of communicating with people around the world, an email message can be viewed on many platforms and dozens of locales. Standards for email interchange, however, are restricted by desktop machines for which the default email standard is Simple Mail Transfer Protocol (SMTP), which supports only 7-bit transmission channels.

The sending agent converts the body of the message into a standard format and labels it as body. The receiving agent looks at the body and, if it supports the character encoding, converts the body into the local character set.

Due to the fact that `dtmail` now uses the Language Conversion Library (LCL), `dtmail` has the capacity to support multibyte characters in both the subject line, the mail body, and in attachments. There is also the ability for `dtmail` to have characters of different encodings within the same mail, for example, SJIS and EUC encodings for the Japanese (`ja`) locale.

OpenWindows

Solaris 7 does not have any changes in OpenWindows with regard to internationalization. Applications that were developed for previous versions of Solaris will run in Solaris 7 without any changes.

The XView toolkit is not codeset independent. Applications that use the XView toolkit are not supported in non-EUC locales, such as `ja_JP.PCK`, `en_US.UTF-8`, or `ko.UTF-8`.

For information on international XView, see the internationalization portions of the *XView Developer's Notes*.

For information on international OLIT, see the internationalization chapter of the *OLIT Reference Manual*.

Printing

Localization Printing Support Under the Solaris 7 Operating Environment

Solaris provides support for PostScript printers. Custom print filters are available to convert localized text to PostScript. See `mp(1)` and `postprint(1)postprintman` pages for further details. The ability to download fonts onto a printer is also present.

For more details see the `download(1)download` man pages. This support is configured for PostScript printers.

No internationalization-specific changes were made to printing in the Solaris 7 product. Look for printing information in the AnswerBook; the *System Administration Guide* has several chapters that discuss printing.

European Printing Support

For European locales based on character sets that are not ISO-8859, such as Greek and Russian, `prolog.ps` files are supplied. The files are located in `/usr/openwin/lib/locale/print`.

When you print in one of these locales, the files are automatically downloaded to the printer. These fonts are PostScript Type 1 and include Times, Helvetica, and Courier.

These are in normal, bold, *italic*, and bold-italic styles.

This allows printing on PostScript printers from both CDE and OpenWindows desktops. From a command line, use `/usr/openwin/bin/mp <filename> | lp` in each locale that is not based on ISO 8859-1 character sets.

For the Eastern European locales such as Russian, non `iso-8859-1` encoded, `prolog.ps` files are supplied. The files are located in:

```
/usr/openwin/lib/locale/locale/directories/print/prolog.ps
```

for each relevant locale. At `directories`, insert one of the following

```
/iso8859-2/  
/iso8859-4/  
/iso8859-5/  
/iso8859-7/  
/iso8859-9/  
/iso8859-10/
```

The files are downloaded automatically when you print in one of the Eastern European locales. A minimum set of fonts allow printing.

The fonts in the `prolog.ps` files are defined in Table 9-1 below.

TABLE 9-1 `prolog.ps` Fonts

<code>/LC_Courier</code>	CourierCyr AliasFont
<code>/LC_Courier-Italic</code>	CourierCyr Inclined AliasFont
<code>/LC_Courier-Bold</code>	CourierCyr Bold AliasFont
<code>/LC_Courier-BoldOblique</code>	CourierCyr BoldInclined AliasFont
<code>/LC_Times-Roman</code>	TimesNewRomanCyr
<code>/LC_Times-Italic</code>	TimesNewRomanCyr-Inclined Aliasfont
<code>/LC_Times-Bold</code>	TimesNewRomanCyr-Bold AliasFont
<code>/LC_Times-BoldOblique</code>	TimesNewRomanCyr-BoldIncl AliasFont
<code>/LC_Helvetica</code>	LucidaSansCyr AliasFont
<code>/LC_Helvetica-Italic</code>	LucidaSansCyr ItalicFont
<code>/LC_Helvetica-Bold</code>	LucidaSansCyr-Bold AliasFont
<code>/LC_Helvetica-BoldOblique</code>	LucidaSansCyr-BoldItalic AliasFont

Asian Multibyte Printing Support

The `xetops` and `xutops` utilities convert Asian text into a bitmapped graphics printed image. This allows you to print Asian characters on PostScript-based printers even without Asian fonts resident on the printers.

A typical command line for printing such a file would be as follows:

```
system% pr <filename> | xetops |lp
```

or

```
system% pr <filename> | xutops |lp (for the ko.UTF-8 locale)
```

Japanese Solaris 7 supports the following Japanese-specific printers:

- Japanese PostScript printer
- Epson VP-5085 (based on ESC/P)
- NEC PC-PR201 (based on 201PL)
- Canon LASERSHOT (based on LIPS)

Japanese texts can be printed with these printers through the LP print service. Table 9-2 shows the relation between these printers and user components. See *JFP User's Guide* for further details.

TABLE 9-2 Japanese Printer Support

Printer	terminfo(-T)	interface(-i)	content(-I)	filter
Japanese PS	PS	jstandard	postscript	jpostprint
Epson VP-5085	epson-vp5085	jstandard	None	jprconv
NEC PC-PR201	nec-pr201	jstandard	None	jprconv
Canon LASERSHOT	canon-ls-a408	jstandard	None	jprconv

Use the following to set up a Japanese PostScript printer.

In the following example, the PostScript printer name is `lw`. The `/dev/lp1` is the device that is associated with the printer. For more information, see the `lpadmin(1M)lpadmin` man page.

```
# lpadmin -p lw -v /dev/lp1 -T PS -I postscript
# lpadmin -p lw -i /usr/lib/lp/model/jstandard
# cd /etc/lp/fd
# lpfilter -x -f postprint
# lpfilter -f jpostprint -F jpostprint.fd
# accept lw
# enable lw
# /etc/init.d/lp stop
# /etc/init.d/lp start
```

To print, use the following operation:

```
% lp -d lw Japanese Text File
```

Note - These features are supported only on Japanese Solaris. Input codesets to a printer depend on the system locale.

CDE Font Downloader

PostScript printers connected to a Solaris host may each have different sets of resident fonts. Users can purchase additional fonts and install them on the host or they can remove fonts. There are a number of different ways to accomplish this task. Note, however, that there is no user-level command to manage fonts on a PostScript printer.

Font Downloader is a CDE application for managing fonts on PostScript printers. Specifically it will provide the following functionality:

- Download PostScript fonts to a Postscript printer
- Download TrueType fonts to a PostScript printer
- Remove previously downloaded font from a printer
- Check printer memory
- List all fonts available on the printer
- Print character samples
- Reformat hard disk on the printer

Technical Description

Font downloader/manager program code is reused for the DT Font Downloader. In addition, the following functionality is provided:

- GUI front end
- Support of generic PostScript printers
- Support TrueType font downloading as type 42 fonts
- Support TrueType - PostScript type 3 font conversion
- Change the encoding vectors of the fonts

As a result, DT Font Downloader supports type 1, type 3, type 9 (CID 0), type 10 (CID 1), type 11 (CID 2), and type 42 fonts.

If the printer doesn't support TrueType glyph procedures (font types 11 and 42) fonts are converted to one of the supported formats before downloading.

TrueType binary font files located on the host cannot be used in the printer without converting them to a different format. Some printers may support TrueType glyph procedures. Downloading fonts to such printers requires translating single byte TrueType fonts to Type 42 fonts and double byte TrueType font to Type 11 fonts.

Printers with no support for TrueType glyph procedures do not interpret Type 42 or Type 11 fonts. In this case more complex translation is required.

Single-byte TrueType fonts can be translated to Type 3 fonts with loss of quality because of interpolation and hint conversion involved in such a translation.

Double-byte TrueType fonts can be translated to Type 10 fonts.

Reference Documents

- *PostScript Language Reference Manual*, Adobe Systems. - 2nd ed. ISBN 0-201-18127-4 - defines PostScript language and fonts
- *PostScript Language Extension for CID-Keyed Fonts*. PostScript™ Software Version 2015 Adobe Systems 29 June 1995
- *The Type 42 Font Format Specification*. Adobe Systems Technical Note #5012 1 March 1993
- *Fontadmin Functional Specification* - 12/04/95 - documents font downloader
- *TrueType Font File Version 1.00* Microsoft Corporation —available from ftp.microsoft.com - documents TrueType fonts

Complex Text Layout

Overview of CTL Technology

Complex Text Layout (CTL) extensions enable Motif APIs to support writing systems that require complex transformations between logical and physical text representations, such as Arabic, Hebrew, and Thai. CTL Motif provides character shaping, such as ligatures, diacritics, and segment ordering, and supports the transformation of static and dynamic text widgets. It also supports right-to-left and left-to-right text orientation and tabbing for dynamic text widgets. Since text rendering is handled through the rendition layer, other widget libraries can be easily extended to support CTL.

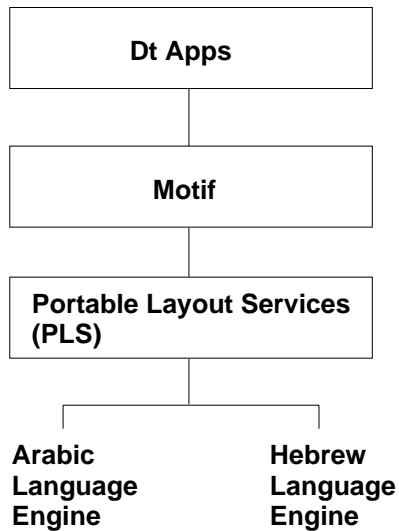
To leverage the new features, users must have the Portable Layout Services (PLS) library and the appropriate language engine. CTL uses PLS as the interface to the language engine, and uses the language engine to transform text before it is rendered. Applications that support CTL must include additional resources as described in the CTL documentation.

Specifically, `XmCTL` supports the following complex language shaping and reordering features provided by underlying locale-dependent PLS module transformations:

- Positional variation
- Ligation (many-to-one) and character composition (one-to-many)
- Diacritics
- Bi-directionality
- Symmetrical swapping
- Numeral shaping
- String validation

Overview of CTL Architecture

The CTL architecture is organized as shown in the diagram below. Dt Apps at the top of the stack employs Motif CTL functionality for rendering text. Motif in turn interfaces with locale-specific language engines using PLS and performs transformations to support positional variation, numeral shaping, and so on.



The CTL architecture is built to support new languages by simply adding a new locale-specific engine. In other words, support for Thai and Vietnamese can be added without altering Motif or Dt Apps.

Changes in Motif to Support CTL Technology

XmDirection

The `XmNlayoutDirection` resource¹ controls object layout. It interacts with the orientation value of the `LayoutObject` in the following manner.

Description

First, when `XmNlayoutDirection` is specified as `XmDEFAULT_DIRECTION`, then the widget's layout direction is set at creation time from the governing pseudo-XOC. In the case of dynamic text (`XmText` and `XmTextField`), the governing pseudo-XOC is the one that is associated with the `XmRendition` used for the widget. In the case of static text (`XmList`, `XmLabel`, `XmLabelG`), the layout direction is set from the first compound string component that specifies a direction. This specification happens in one of two ways:

- Directly, if the component is of type `XmSTRING_COMPONENT_LAYOUT_PUSH` or `XmSTRING_COMPONENT_DIRECTION`
- Indirectly, if the component is of type `XmSTRING_COMPONENT_LOCALE_TEXT`, `XmSTRING_COMPONENT_WIDECHAR_TEXT`, or `XmSTRING_COMPONENT_TEXT`, from the component's associated `XmRendition`'s associated `LayoutObject`.

Second, if `XmNlayoutDirection` is not specified as `XmDEFAULT_DIRECTION`, and the `XmNlayoutModifier @ls` orientation value is not specified explicitly in the layout modifier string, then the `XmNlayoutDirection` value is passed through to the XOC and its `LayoutObject`.

If both `XmNlayoutDirection` and the `XmNlayoutModifier @ls` orientation value are explicitly specified, then the behavior is mixed; the `XmNlayoutDirection` controls widget object layout, and the `XmNlayoutModifier @ls` orientation value controls layout transformations.

For More Information

For more information, see *CAE Specification: Portable Layout Services: Context-dependent and Directional Text*, The Open Group: Feb 1997; ISBN 1-85912-142-X; document number C616.

1. See section 11.3 of the *Motif Programmer's Guide* (Release 2.1) for an overview of `XmNlayoutDirection`, and especially for a description of the interaction between `XmStringDirection` and `XmNlayoutDirection`.

XmStringDirection

`XmStringDirection` is the data type used to specify the direction in which the system displays characters of a string.

Description

The `XmNlayoutDirection` resource sets a default rendering direction for any compound string (`XmString`) that does not have a component specifying the direction of that string. Therefore, to set the layout direction, all that is required is to set the appropriate value for the `XmNlayoutDirection` resource. It is not required that you create compound strings with specific direction components. When the application renders an `XmString`, it should look to see if the string was created with an explicit direction (`XmStringDirection`). If there is no direction component, the application should check the value of the `XmNlayoutDirection` resource for the current widget and use that value as the default rendering direction for the `XmString`.

Related Information

See also `XmRendition` and `XmDirection`.

XmRendition

CTL adds the following new pseudo resources to `XmRendition`:

New Resources

Name	Class/Type	Access	Default Value
XmNfontType	XmCFontType/XmFontType	CSG	XmAS_IS
XmNlayoutAttrObject	XmCLayoutAttrObject/String	CG	NULL
XmNlayoutModifier	XmCLayoutModifier/String	CSG	NULL

XmNfontType

Specifies the type of the Rendition font object. For CTL, the value of this resource must be the `XmFONT_IS_XOC` value. If it is not, then the `XmNlayoutAttrObject` and `XmNlayoutModifier` resources are ignored.

When the value of this resource is `XmFont_IS_XOC`, and if the `XmNfont` resource is not specified, then at create time the value of the `XmNfontName` resource gets converted into an XOC object in either the locale specified by the `XmNlayoutAttrObject` resource or the current locale. Furthermore, the value of the `XmNlayoutModifier` resource gets passed through to any `LayoutObject` associated with the XOC.

XmNlayoutAttrObject

Specifies the layout `AttrObject` argument to be used to create the `Layout Object` associated with the XOC associated with this `XmRendition`. Refer to the `Layout Services m_create_layout()` specification for the syntax and semantics of this string. (See the description of `XmNfontType` above for an explanation of the interaction between the `Layout Modifier Orientation` output value and the `XmNlayoutDirection` widget resource.)

XmNlayoutModifier

Specifies the layout values to be passed through to the `Layout Object` associated with the XOC associated with this `XmRendition`. For the syntax and semantics of this string, see *CAE Specification*.

Setting this resource using `XmRendition{Retrieve,Update}` causes the string to be passed through to the `LayoutObject`

associated with the XOC associated with this Rendition. This is the mechanism for configuring layout services dynamically. Note that unpredictable behavior may result if the Orientation, Context, TypeOfText, TextShaping, or ShapeCharset are changed.

Additional Behavior

The `XmNlayoutModifier` affects the layout behavior of the text associated with the `XmRendition`. For example, if the layout default treatment of numerals is `NUMERALS_NOMINAL`, the user can change to `NUMERALS_NATIONAL` by setting `XmNlayoutModifier` to:

- `@ls numerals=nominal:national, or`
- `@ls numerals=:national`

The layout values can be classified into the following groups:

- **Encoding description:** `TypeOfText`, `TextShaping`, `ShapeCharset` (and locale codeset)
`TypeOfText` is essentially segment ordering, and can be illustrated with opaque blocks. It is usually not meaningful to modify these values dynamically through the `Rendition` object, and almost certain to result in unpredictable behavior.
- **Layout behavior:** `Orientation`, `Context`, `ImplicitAlg`, `Swapping`, `Numerals`
`Orientation` and `Context` should not be modified dynamically; it is safe to modify `ImplicitAlg`, `Swapping`, and `Numerals`.
- **Editing behavior:** `CheckMode`

XmText, XmTextField

Description

`Xm CTL` extends `XmText` and `XmTextField` by adding a parallel set of movement and deletion actions that operate visually, patterned after the Motif 2.0 `CSText` widget. The standard Motif 2.1 `Text` and `TextField` do not distinguish between logical and physical order: “next” and “forward” mean “to the right,” and “previous” and “backward” mean “to the left.” `CSText`, however, makes the proper distinction and defines a new set of actions with strictly physical names (for example, `left-character()`, `delete-right-word()`, and so on). All of these

action routines are defined to be sensitive to the `XmNlayoutDirection` of the widget and to call the appropriate “next-“ or “previous-“ action. The `Xm CTL` extensions are slightly more complex than `CSText`'s in that they are sensitive not to the global orientation of the widget, but to the specific directionality of the physical characters surrounding the cursor, as determined by the pseudo-XOC (including neutral stabilization).

There is also a new resource to control selection policy, to provide a rendition tag, and to control alignment.

The set of new `Xm CTL` actions is roughly the cross product of `{Move,Delete,Kill}` by `{Left,Right}` by `{Character,Word}`, and is listed below.

New Resources

The following new resources are added to `XmText` and `XmTextField`:

Name	Class/Type	Access	Default Value
<code>XmNrenditionTag</code>	<code>XmCRenditionTag/XmRString</code>	CSG	<code>XmFONTLIST_DEFAULT_TAG</code>
<code>XmNalignment</code>	<code>XmCAlignment/XmRAlignment</code>	CSG	<code>XmALIGNMENT_BEGINNING</code>
<code>XmNeditPolicy</code>	<code>XmCEditPolicy/XmREditPolicy</code>	CSG	<code>XmEDIT_LOGICAL</code>

`XmNrenditionTag` Specifies the rendition tag of the `XmRendition` (in the `XmNrenderTable` resource) to be used for this widget.

`XmNalignment` Specifies the text alignment to be used in the widget. Only `XmALIGNMENT_END` and `XmALIGNMENT_CENTER` are supported.

`XmNeditPolicy` Specifies the editing policy to be used for the widget, either `XmEDIT_LOGICAL` or `XmEDIT_VISUAL`. In the case of `XmEDIT_VISUAL`, selection, cursor movement, and deletion are in a visual style. Setting this resource also changes the translations for the standard keyboard movement and deletion events either to the new “visual“ actions list below or to the existing logical actions.

Action Routines

All of the actions in the following list query the orientation of the character in the direction specified. If the direction is left-to-right, they call the corresponding next-/forward- or previous-/backward- variants.

- `delete-left-character()`
- `delete-left-word()`
- `delete-right-character()`
- `delete-right-word()`
- `kill-left-character()`
- `kill-left-word()`
- `kill-right-character()`
- `kill-right-word()`
- `left-character()`
- `left-word()`
- `prev-cell()`
- `right-character()`
- `right-word()`
- `forward-cell()`

Additional Behavior

The actions determine the orientation of characters by using the Layout Services transformation `OutToInp` and `Property` buffers (for the nesting level). The widget's behavior is therefore dependent on the locale-specific transformation. If the information in the `OutToInp` or, especially, `Property` buffers is inaccurate, the widget may behave unexpectedly. Moreover, as the locale-specific modules fall outside of the scope of this specification, bi-directional editing behavior may differ from platform to platform for the same text, application, resource values, and `LayoutObject` configuration.

The visual mode actions result in display cell-based behavior. The logical mode actions result in logical character-based behavior. For example, the `delete-right-character()` operation deletes the input buffer characters that correspond to the display cell (that is, one input buffer character whole `LayoutObject` transformation “property” byte “new cell indicator” is 1, and all of the succeeding characters whose “new cell indicator”² is 0).

². For more information on the `Property` buffer, see the specification for `m_transform_layout()` in *CAE Specification*.

Similarly, for `backward-character()`, the insertion point is moved backward one character in the input buffer, and the cursor is redrawn at the visual location corresponding to the associated output buffer character. This means that several keystrokes are required to move across a composite display cell; the cursor does not actually change display location as the insertion point moves across input buffer characters whose “new cell indicator“ is 0 (that is, diacritics or ligature fragments).

This means deletion operates either from the logical/input buffer side, or from the display cell level of the physical/output side. There is no mode for a strict, physical character-by-character deletion, since there is no one-to-one correspondence between the input and output buffers. A given physical character may represent only a fragment of a logical character, for example.

Action Routines

The `XmText` action routines are as follows:

`left-character(extend)`

If the `XmNeditPolicy` is `XmEDIT_LOGICAL` and is called without arguments, it moves the insertion cursor back logically by a character. If the insertion cursor is at the beginning of the line, it moves the insertion cursor to the logical last character of the previous line if one exists, otherwise the insertion cursor position doesn't change.

If the `XmNeditPolicy` is `XmEDIT_VISUAL`, then the cursor moves to the left of cursor position. If the insertion cursor is at the beginning of the line, then it moves to the end character of the previous line if one exists.

If it is called with an `extend` argument, it moves the insertion cursor as in the case of no argument and extends the current selection.

The `left-character()` action produces calls to the `XmNmotionVerifyCallback` procedures with the reason value `XmCR_MOVING_INSERT_CURSOR`. If called with an `extend` argument, this may produce calls to the

XmNgainPrimaryCallback procedures. See the callback description in the Motif *Programmer's Reference* for more information.

`left-word(extend)`

If the `XmNeditPolicy` is `XmEDIT_LOGICAL` and is called without any arguments, and the insertion cursor is at the logical starting of the word, it moves the insertion cursor to the logical starting of the logical preceding word, if one exists, otherwise the insertion cursor position doesn't change. If the insertion cursor is in the word but not at the logical start of the word, it moves the insertion cursor to the logical start of the word. If the insertion cursor is at the logical start of the line, it moves the insertion cursor to the logical start of the logical last word in the previous line if one exists, otherwise the insertion cursor position doesn't change.

If the `XmNeditPolicy` is `XmEDIT_VISUAL` and is called without arguments, it moves the insertion cursor to the first non-white space character after the first white space character to the left or after the beginning of the line. If the insertion cursor is already at the beginning of the word, it moves the insertion cursor to the beginning of the previous word. If the insertion cursor is already at the beginning of the line, it moves to the starting of the last word in the previous line.

If called with an argument of `extend`, it moves the insertion cursor as in the case of no argument and extends the current selection.

The `left-word()` action produces calls to the `XmNmotionVerifyCallback` procedures with the reason value `XmCR_MOVING_INSERT_CURSOR`. If it is called with an `extend` argument, this

may produce calls to the `XmNgainPrimaryCallback` procedures. See the callback description in the *Motif Programmer's Reference* for more information.

`right-character(extend)`

If the `XmNeditPolicy` is `XmEDIT_LOGICAL` and is called without any arguments, it moves the insertion cursor logically forward by a character. If the insertion cursor is at the logical end of the line, it moves the insertion cursor to the logical starting of the next line, if one exists.

If the `XmNeditPolicy` is `XmEDIT_VISUAL`, then the cursor moves to the right of cursor position. If the insertion cursor is at the end of the line, it moves the insertion cursor to the starting of the next line, if one exists.

If called with an argument of `extend`, it moves the insertion cursor as in the case of no argument and extends the current selection.

The `right-character()` action produces calls to the `XmNmotionVerifyCallback` procedures with the reason value `XmCR_MOVING_INSERT_CURSOR`. If called with `extend` argument, this may produce calls to the `XmNgainPrimaryCallback` procedures. See the callback description in the *Motif Programmer's Reference* for more information.

`right-word(extend)`

If the `XmNeditPolicy` is `XmEDIT_LOGICAL` and is called without any arguments, it moves the insertion cursor to the logical starting of the logical succeeding word if one exists, otherwise it moves to the logical end of the current word. If the insertion cursor is at the logical end of the line or in the

logical last word of the line, it moves the cursor to the logical first word in the next line if one exists, otherwise it moves to the logical end of the current word.

If the `XmNeditPolicy` is `XmEDIT_VISUAL` and is called without arguments, it moves the insertion cursor to the first nonwhite space character after the first white space character to the right or after the end of the line.

If called with an argument of `extend`, it moves the insertion cursor as in the case of no argument and extends the current selection.

The `left-word()` action produces calls to the `XmNmotionVerifyCallback` procedures with the reason value `XmCR_MOVING_INSERT_CURSOR`. If called with `extend` argument, this may produce calls to the `XmNgainPrimaryCallback` procedures. See the callback description in the *Motif Programmer's Reference* for more information.

`delete-left-character()`

If the `XmNeditPolicy` is `XmEDIT_LOGICAL`, it is equivalent to `delete-previous-char`. If the `XmNeditPolicy` is `XmEDIT_VISUAL`, then in normal mode, if there is a non-null selection, it deletes the selection; otherwise it deletes the character left of the insertion cursor. In add mode, if there is a non-null selection, the cursor is not disjointed from the selection and `XmNpendingDelete` is set to `True`, it deletes the selection; otherwise it deletes the character left of the insertion cursor. This may impact the selection.

The `delete-left-character()` action produces calls to the `XmNmodifyVerifyCallback` procedures with reason value

XmCR_MODIFYING_TEXT_VALUE and the XmNvalueChangedCallback procedures with reason value XmCR_VALUE_CHANGED.

`delete-right-character()`

If the XmNeditPolicy is XmEDIT_VISUAL, it is equivalent to `delete-next-character`. If the XmNeditPolicy is XmEDIT_VISUAL, then in normal mode, if there is a non-null selection, it deletes the selection; otherwise, it deletes the character right of the insertion cursor. In add mode, if there is a non-null selection and the cursor is not disjointed from the selection, the XmNpendingDelete is set to True and the selection is deleted; otherwise, the character right of the insertion cursor is deleted. This may impact the selection.

The `delete-right-character()` action produces calls to the XmNmodifyVerify- Callback procedures with reason value XmCR_MODIFYING_TEXT_VALUE, and the XmNvalue- ChangedCallback procedures with reason value XmCR_VALUE_CHANGED.

`delete-left-word()`

If the XmNeditPolicy is XmEDIT_VISUAL, it is equivalent to `delete-prev-word()`. If the XmNeditPolicy is XmEDIT_LOGICAL, then in normal mode, if there is a non-null selection, it deletes the selection; otherwise, it deletes the characters left of the insertion cursor to the next space, punctuation character, tab, or beginning-of-line character. In add mode, if there is a non-null selection, the cursor is not disjointed from the selection; otherwise it deletes the characters left of the insertion cursor the right space, tab, or beginning-of-line character. In add mode, if there is a non-null selection, the cursor is not

disjointed from the selection, the `XmNpendingDelete` is set to `True`, and the selection is deleted; otherwise, it deletes the character left of the insertion cursor, the right space, tab, or beginning of new-line character. This may impact the selection.

`delete-right-word()`

If the `XmNeditPolicy` is `XmEDIT_VISUAL`, it is equivalent to `delete-right-word()`. If the `XmNeditPolicy` is `XmEDIT_LOGICAL`, then in normal mode, if there is a non-null selection, it deletes the selection; otherwise, it deletes the characters right of the insertion cursor to the next space, punctuation character, tab, or end-of-line character. In add mode, if there is a non-null selection, the cursor is not disjointed from the selection, `XmNpendingDelete` is set to `True`, and deletes the selection; otherwise, it deletes the characters right of the insertion cursor to the next space, tab, or end-of-line character. This may impact the selection.

`kill-left-character()`

If the `XmNeditPolicy` is `XmEDIT_LOGICAL`, it is equivalent to `kill-prev-char`. If the `XmNeditPolicy` is `XmEDIT_VISUAL`, then in normal mode, if there is a non-null selection, it deletes the selection; otherwise, it kills the character left of the insertion cursor and stores the character in the cut buffer. In add mode, if there is a non-null selection, the cursor is not disjointed from the selection, `XmNpendingDelete` is set to `True`, and deletes the selection; otherwise, it deletes the character left of the insertion cursor. This may impact the selection.

The `kill-left-character()` action produces calls to the `XmNmodifyVerifyCallback` procedures with the reason value

XmCR_MODIFYING_TEXT_VALUE, and produces the XmNvalueChangedCallback procedures with the reason value XmCR_VALUE_CHANGED.

kill-right-character()

If the XmNeditPolicy is XmEDIT_VISUAL, it is equivalent to delete-next-character. If the XmNeditPolicy is XmEDIT_VISUAL, then in normal mode, if there is a non-null selection, it deletes the selection; otherwise, it deletes the character right of the insertion cursor and stores it in the cut buffer. In add mode, if there is a non-null selection, the cursor is not disjointed from the selection, the XmNpendingDelete is set to True and deletes the selection; otherwise, it deletes the character right of the insertion cursor. This may impact the selection.

The kill-right-character() action produces calls to the XmNmodifyVerify-Callback procedures with reason value XmCR_MODIFYING_TEXT_VALUE, and produces calls to the XmNvalue-ChangedCallback procedures with reason value XmCR_VALUE_CHANGED.

kill-left-word()

If the XmNeditPolicy is XmEDIT_VISUAL, it is equivalent to delete-prev-word(). If the XmNeditPolicy is XmEDIT_LOGICAL, then in normal mode, if there is a non-null selection, it deletes the selection; otherwise, it deletes the characters left of the insertion cursor to the next space, punctuation character, tab, or beginning-of-line character. In add mode, if there is a non-null selection, the cursor is not disjointed from the selection; otherwise it deletes the characters left of the insertion cursor

the right space, tab, or beginning-of-line character and stores it in the cut buffer. In add mode, if there is a non-null selection, the cursor is not disjointed from the selection, `XmNpendingDelete` is set to `True` and deletes the selection; otherwise it deletes the characters left of the insertion cursor the right space, tab, or beginning of new-line character. This may impact the selection.

`kill-right-word()`

If the `XmNeditPolicy` is `XmEDIT_VISUAL`, it is equivalent to `delete-right-word()`. If the `XmNeditPolicy` is `XmEDIT_LOGICAL`, then in normal mode, if there is a non-null selection, it deletes the selection; otherwise, it deletes the characters right of the insertion cursor to the next space, tab, or end-of-line character. In add mode, if there is a non-null selection, the cursor is not disjointed from the selection, `XmNpendingDelete` is set to `True`, and deletes the selection; otherwise, it deletes the characters right of the insertion cursor to the next space, punctuation character, tab, or end-of-line character and stores in the cut buffer. This may impact the selection.

A few cell-based routines are implemented to support character composition, ligatures, and diacritics. In other words, two or more characters might be represented by a single glyph occupying one presentation cell.

The `XmText` cell action routines are as follows:

`prev-cell(extend)`

Moves the insertion cursor back one cell. If the `XmNeditPolicy` is `XmEDIT_LOGICAL`, then the insertion cursor is moved to the start of the cell that precedes the current cell logically, if one exists; otherwise it moves to the start of the current cell.

If the `XmNeditPolicy` is `XmEDIT_VISUAL`, then the cursor moves to the start of cell to the left of the cursor, if one exists. The `prev-cell()` action produces calls to the

`forward-cell(extend)`

XmNmotionVerifyCallback procedures with the reason value XmCR_MOVING_INSERT_CURSOR. If called with an extend argument, this may produce calls to the XmNgainPrimaryCallback procedures. See the callback description in the Motif *Programmer's Reference* for more information.

Moves the insertion cursor to the start of the logical next cell, if one exists; otherwise it moves it to the end of the cell. If the XmNeditPolicy is XmEDIT_LOGICAL, then the cursor moves forward one cell.

If the XmNeditPolicy is XmEDIT_VISUAL, then the cursor moves to the start of the cell to the right of the cursor position if one exists; otherwise it moves to the end of the current cell. The `forward-cell()` action produces calls to the XmNmotionVerifyCallback procedures with the reason value XmCR_MOVING_INSERT_CURSOR. If called with an extend argument, this may produce calls to the XmNgainPrimaryCallback procedures. See the callback description in the Motif *Programmer's Reference* for more information.

XmTextFieldGetLayoutModifier

Purpose

A `TextField` function that returns the layout modifier string that reflects the state of the layout object tied to its rendition.

Synopsis

```
#include <Xm/TextF.h>  
String XmTextFieldGetLayoutModifier(Widget widget)
```

Description

`XmTextFieldGetLayoutModifier` accesses the value of the current layout object settings of the rendition associated with the widget. When the layout object modifier values are changed using a convenience function, the `XmTextFieldGetLayoutModifier` function returns the complete state of the layout object, not just the changed values.

Return Value

Returns the layout object modifier values in the form of a String value.

Related Information

`XmTextField`

XmTextGetLayoutModifier

Purpose

A `Text` function that returns the layout modifier string that reflects the state of the layout object tied to its rendition.

Synopsis

```
#include <Xm/Text.h>String XmTextGetLayoutModifier(Widget widget)
```

Description

`XmTextGetLayoutModifier` accesses the value of the current layout object settings of the rendition associated with the widget. When the layout object modifier values are changed using a convenience function, the `XmTextGetLayoutModifier` function returns the complete state of the layout object, not just the changed values.

Return Value

Returns the layout object modifier values in the form of a String value.

Related Information

XmText

XmTextFieldSetLayoutModifier

Purpose

A `TextField` function that sets the layout modifier values, which changes the behavior of the layout object tied to its rendition.

Synopsis

```
#include <Xm/TextF.h>void XmTextFieldSetLayoutModifier(Widget  
widget, string layout_modifier)
```

Description

`XmTextFieldSetLayoutModifier` modifies the layout object settings of a rendition associated with the widget. When the layout object modifier values are set using this convenience function, only the attributes specified in the input parameter are changed; the rest of the attributes are left untouched.

Related Information

XmTextField

XmTextSetLayoutModifier

Purpose

A `Text` function that sets the layout modifier values, which changes the behavior of the layout object tied to its rendition.

Synopsis

```
#include <Xm/Text.h>void XmTextSetLayoutModifier(Widget  
widget, string layout_modifier)
```

Description

`XmTextSetLayoutModifier` modifies the layout object settings of a rendition associated with the widget. When the layout object modifier values are set using this convenience function, only the attributes specified in the input parameter are changed; the rest of the attributes are left untouched.

Related Information

`XmText`

XmStringDirectionCreate

Synopsis

```
#include <Xm/Xm.h>XmString XmStringDirectionCreate(direction)  
XmStringDirectiondirection
```

Description

`XmStringDirectionCreate` creates a compound string with a single component, a direction with the given value. On the other hand, the `XmNlayoutDirection`

resource sets a default rendering direction for any compound string (`XmString`) that does not have a component specifying the direction for that string. Therefore, to set the layout direction, all that is required is to set the appropriate value for the `XmNlayoutDirection` resource. It is not required to create compound strings with specific direction components. When the application renders an `XmString`, it should look to see if the string was created with an explicit direction (`XmStringDirection`). If there is no direction component, the application should check the value of the `XmNlayoutDirection` resource for the current widget and use that value as the default rendering direction for the `XmString`.

Related Information

See also `XmRendition`, `XmDirection`.

UIL

UIL Argument Name	Argument Type
<code>XmNlayoutAttrObject</code>	String
<code>XmNlayoutModifier</code>	String
<code>XmNrenditionTag</code>	String
<code>XmNalignment</code>	Integer
<code>XmNeditPolicy</code>	Integer

How to Develop CTL Applications

Layout Direction

The direction of a compound string is stored so that the data structure will be equally useful for describing text in left-to-right languages such as English, Spanish, French, and German, as well as for text in right-to-left languages, such as Hebrew

and Arabic. In Motif applications, you can set the layout direction using the `XmNlayoutDirection` resource from the `VendorShell` or `MenuShell`. `Manager` and `Primitive` widgets (as well as `Gadgets`) also have an `XmNlayoutDirection` resource. The default value is inherited from the closest ancestor that has the same resource.

In the case of an `XmText` widget, you need to specify the vertical direction as well. Setting the `layoutDirection` to `XmRIGHT_TO_LEFT` will result in the string direction from right-to-left, but the cursor will move vertically down. If the vertical direction is important and top to bottom is desired, be sure to specify `XmRIGHT_TO_LEFT_TOP_TO_BOTTOM`, which specifies that the components are laid out from right-to-left first and then top-to-bottom, and will result in the desired behavior.

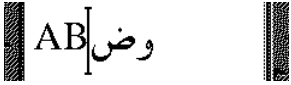
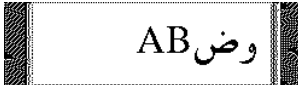
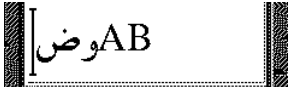
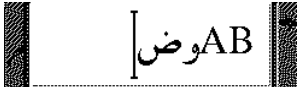
Furthermore, the behavior of `XmText` and `TextField` widgets is influenced by the `XmNalignment` and `XmNlayoutModifier` resources of the `XmRendition`. These resources, in addition to `XmNlayoutDirection`, control the layout behavior of the `Text` widget. This can be illustrated using the example below.

A B و ض


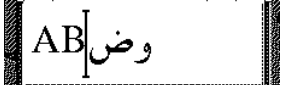
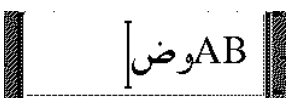
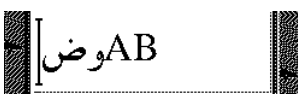
The input string used in the illustration is

The `XmNlayoutModifier` string `@ls orientation=` setting values for this illustration are shown in the left column.

Layout Direction: `XmLEFT_TO_RIGHT`

	<code>XmALIGNMENT_BEGINNING</code>	<code>XmALIGNMENT_END</code>
<code>@ls orientation= ltr:ltr</code>		
<code>@ls orientation= rtl:rtl</code>		

Layout Direction: `XmRIGHT_TO_LEFT`

	<code>XmALIGNMENT_BEGINNING</code>	<code>XmALIGNMENT_END</code>
<code>@ls orientation= ltr:ltr</code>		
<code>@ls orientation= rtl:rtl</code>		

As the illustration shows, `XmNAlignment` dictates whether the text is flush-right or -left in conjunction with the layout direction. On the other hand, `XmNlayoutModifier` breaks the text into segments and arranges them left-to-right or right-to-left depending on the orientation value. In other words, if the `XmNlayoutDirection` is `XmRIGHT_TO_LEFT`, and the `XmNAlignment` value is `XmALIGNMENT_BEGINNING`, the string is flush-right.

Creating a Rendition

The following code creates an `XmLabel` whose `XmNlabelString` is of the type `XmCHARSET_TEXT`, using the Rendition whose tag is “ArabicShaped.” The Rendition is created with an `XmNlayoutAttrObject` of “ar” (corresponding to the locale name for the Arabic locale) and a layout modifier string that specifies for the output buffer a Numerals value of `NUMERALS_CONTEXTUAL` and a ShapeCharset value of “unicode-1.”

The locale-specific layout module transforms its input text (in this example encoded in ISO 8859-6) in an output buffer of physical characters encoded using the 16-bit Unicode 2.0 codeset. Since an explicit layout locale has been specified, this text is rendered properly independent of the runtime locale setting.

```
int n;
Arg args[10];
Widget w;
XmString labelString;
XmRendition rendition;
XmStringTag renditionTag;
XmRenderTable renderTable;
    /* alef lam baa noon taa - iso8859-6 */
labelString = XmStringGenerate("\307\344\310\346\312\312", NULL,
    XmCHARSET_TEXT, "ArabicShaped");
w = XtVaCreateManagedWidget("a label", xmLabelWidgetClass, parent,
    XmNlabelString, labelString,
    XmNlabelType, XmSTRING,
    NULL);

n = 0;
XtSetArg(args[n], XmNfontName, "-*-medium-r-normal--24-*-*-*-*-*");
n++;
XtSetArg(args[n], XmNfontType, XmFONT_IS_XOC); n++;
XtSetArg(args[n], XmNlayoutAttrObject, "ar"); n++;
XtSetArg(args[n], XmNlayoutModifier,
    "@ls numerals=:contextual, shapecharset=iso8859-6"); n++;
renditionTag = (XmStringTag) "ArabicShaped";
rendition = XmRenditionCreate(w, renditionTag, args,
s, n);
renderTable =
    XmRenderTableAddRenditions(NULL, &rendition, 1, XmREPLACE_MERGE);
XtVaSetValues(w, XmNrenderTable, renderTable, NULL);
```

Editing a Rendition

The following code creates a `TextField` widget and a `RenderTable` with a single Rendition. Note that both the `XmNlayoutAttrObject` and `XmNlayoutModifier` pseudo resources have been left unspecified and therefore defaults to `NULL`. This means the `LayoutObject` associated with the Rendition is the default locale's, if one exists.

For this example to work properly, the locale must be set to one whose codeset is `ISO 8859-6` and whose locale-specific layout module can support the `IMPLICIT_BASIC` algorithm. It then modifies the Rendition's `LayoutObject`'s `ImplicitAlg` value via the Rendition's `XmNlayoutModifier` pseudo resource.

```
int n;
Arg args[10];
Widget w;
    XmRendition rendition;
XmStringTag renditionTag;
XmRenderTable renderTable;
w = XmCreateTextField(parent, "text field", args, 0);
n = 0;
    XtSetArg(args[n], XmNfontName, "-*-medium-r-normal-*-24-*-*-*-*-*");
        n++;
    XtSetArg(args[n], XmNfontType, XmFONT_IS_XOC); n++;
    renditionTag = (XmStringTag) "ArabicShaped";
    rendition = XmRenditionCreate(w, renditionTag, args, n);
    renderTable =
        XmRenderTableAddRenditions(NULL, &rendition, 1, XmREPLACE_MERGE);
XtVaSetValues(w, XmNrenderTable, renderTable, NULL);
    ....
n = 0;
XtSetArg(args[n], XmNlayoutModifier, "@ls implicitalg=basic");
    n++;
XmRenditionUpdate(rendition, args, n);
```

Related Information

See also `XmDirection`, `XmText`.

Creating a Render Table in a Resource File

Renditions and render tables may be specified in resource files. For properly internationalized application, in fact, this is the preferred method. When the render tables are specified in a file, the program binaries are made independent of the particular needs of a given locale, and may be easily customized to local needs.

Render tables are specified in resource files with the following syntax:

resource_spec: [*tag* [, *tag*] *]

where tag is some string suitable for the `XmNtag` resource of a rendition.

This line creates an initial render table containing one or more renditions as specified. The renditions are attached to the specified tags

```
resource_spec[*|. ] rendition[*|. ]resource_name: value
```

The following examples illustrate the CTL resources related to `XmRendition` that can be set using resource files. The `fontType` must be set to `FONT_IS_XOC` for the layout object to take effect. The `layoutModifier` specified using `@ls` is passed on to the layout object by the rendition object.

For a complete list of resources that can be set on the layout object using `layoutModifier`, see *CAE Specification: Portable Layout Services: Context-dependent and Directional Text*, The Open Group: Feb 1997; ISBN 1-85912-142-X; document number C616.

Creating a Render Table in an Application

Before creating a render table, an application program must first have created at least one of the renditions that is part of the table. The `XmRenderTableAddRenditions` function, as its name implies, is also used to augment a render table with new renditions. To create a new render table, call the `XmRenderTableAddRenditions()` function with a `NULL` argument in place of an existing render table.

The following code creates a render table using a rendition created with `XmNfontType` set to `XmFONT_IS_XOC`.

```
int n;
Arg args[10];
Widget w;
XmString labelString;
XmRendition rendition;
XmStringTag renditionTag;
XmRenderTable renderTable;
/* alef lam baa noon taa - iso8859-6 */
labelString = XmStringGenerate("\307\344\310\346\312\", NULL
                               XmCHARSET_TEXT, "ArabicShaped");
w = XtVaCreateManagedWidget("a label", xmLabelWidgetClass, parent,
                             XmNlabelString, labelString,
                             XmNlabelType, XmSTRING,
                             NULL);
n = 0;
XtSetArg(args[n], XmNfontName, "-*-medium-r-normal--24--*--*--*--*");
n++;
XtSetArg(args[n], XmNfontType, XmFONT_IS_XOC); n++;
XtSetArg(args[n], XmNlayoutAttrObject, "ar"); n++;
XtSetArg(args[n], XmNlayoutModifier,
```

```
        "@ls numerals=nominal:contextual, shapecharset=iso8859-6"); n++;
renditionTag = (XmStringTag) "ArabicShaped";
rendition = XmRenditionCreate(w, renditionTag, args, n);
renderTable =
    XmRenderTableAddRenditions(NULL, &rendition, 1, XmREPLACE);
XtVaSetValues(w, XmNrenderTable, renderTable, NULL);
```

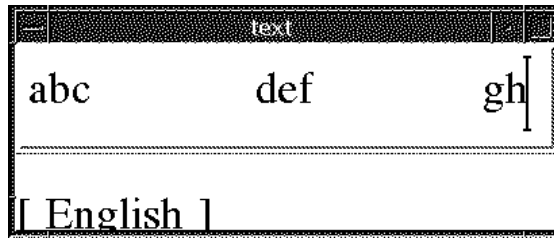
Horizontal Tabs

To control the placement of text, a compound string can contain tab characters. To interpret those characters on display, a widget refers to the rendition in effect for that compound string, where it finds a list of tab stops. However, the dynamic widgets (`TextField` and `XmText`) do not use the tab resource of the rendition. Instead, they compute the tab width using the formula of `8*(width of character 0)`.

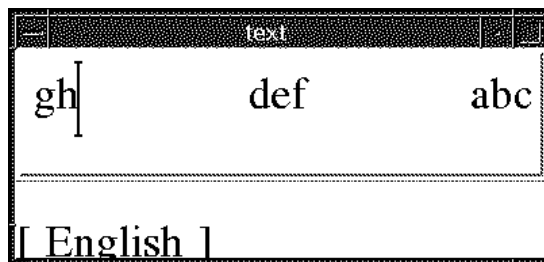
The tab measurement is the distance from the left margin of the compound string display, or from the right margin if the layout direction is right-to-left. It is important to note that regardless of the direction of the text (Arabic right-to-left or English left-to-right) the tab inserts space to the right or left as specified by the layout direction (`XmNlayoutDirection`).

The text following a tab is always aligned at the tab stop, and the tab stop is calculated from the start of the widget, which in turn is influenced by `XmNlayoutDirection`. The behavior of the tabs and their interaction with directionality of the text and the `XmNlayoutDirection` of the widget is illustrated in Table 10-1.

The input for this illustration is `abc\tdef\tgh`.



Layout Direction: XmLEFT_TO_RIGHT



Layout Direction: XmRIGHT_TO_LEFT

Figure 10-1 Tabbing Behavior

Mouse Selection

The user makes a primary selection with SELECT (the left mouse button). Pressing SELECT deselects any existing selection and moves the insertion cursor and the anchor to the position in the text where the button is pressed. Dragging SELECT selects all text between the anchor and the pointer position, deselecting any text outside the range.

The text selected is influenced by the resource `XmNeditPolicy`, which can be set to `XmEDIT_LOGICAL` or `XmEDIT_VISUAL`. If the `XmNeditPolicy` is set to `XmEDIT_LOGICAL`, and if the text selected is bi-directional, the selected text is not be contiguous visually and is a collection of segments. This is because the text in the logical buffer does not have a one-to-one correspondence with the display.

As a result, the contiguous buffer of logical characters of bi-directional text when rendered does not result in a continuous stream of characters. Conversely, when the `XmNeditPolicy` is set to `XmEDIT_VISUAL`, the text selected may be contiguous visually but is segmented in the logical buffer. So the sequence of selection, deletion, and insertion of bi-directional text at the same cursor point does not result in the same string.

Keyboard Selection

The selection operation available with the mouse is also available with the keyboard. The combination of Shift-arrow keys allows the selection of text.

The text selected is influenced by the resource `XmNeditPolicy`, which can be set to `XmEDIT_LOGICAL` or `XmEDIT_VISUAL`. If the `XmNeditPolicy` is set to `XmEDIT_LOGICAL`, and if the text selected is bi-directional, the selected text will not be contiguous visually and will be a collection of segments. This is because the text in the logical buffer does not have one-to-one correspondence with the display. As a result, the contiguous buffer of logical characters of bi-directional text when rendered will not result in a continuous stream of characters.

Conversely, when the `XmNeditPolicy` is set to `XmEDIT_VISUAL`, the text selected may be contiguous visually but is segmented in the logical buffer. So the sequence of selection, deletion, and insertion of bi-directional text at the same cursor point does not result in the same string.

Text Resources and Geometry

Text has several resources that relate to geometry, including the following:

- The render table `XmNrenderTable` that the widget uses to select a font or font set and other attributes in which to display the text.

The `Text` and `Textfield` widgets can use only the font-related rendition resources, such as `XmNfontType`, and can also specify the attributes of the layout object, such as `XmNlayoutAttrObject`, usually a locale identifier, and `XmNlayoutModifier`, which specifies the layout values to be passed through to the Layout Object associated with the XOC associated with this `XmRendition`.

- A resource (`XmNwordWrap`) that specifies whether lines are broken at word boundaries when the text would be wider than the widget.

Breaking a line at a word boundary does not insert a new line into the text. In the case of cursive languages like Arabic, if the word length is greater than the widget length, the word is wrapped to the next line, but the first character in the next line is shaped independently of the previous character in the logical buffer.

Porting Instructions

The new CTL enabled Motif library can be found in `/usr/dt/lib/libXm.so.4`. If your application links to `libXm.so.3` (`ldd app_name` shows which library the application is linking to), then it will not support Complex Text Layout (CTL). In order to port the existing applications to enable CTL, you need to perform the following steps.

1. Add `-DSUN_CTL` to your Makefile. This flag is important and includes the necessary data structures to support CTL. This should be set during compilation.
2. Recompile the existing application. It will automatically link with the CTL enabled Motif library `libXm.so.4`.
3. Add the following resources to your application resource file. Without these resources the layout engine of the locale will not launch.
4. Refer to the sample application attached with your documentation.

Note - Use the font name that is available and appropriate to your locale in the `fontName` resource.

5. If you want the cell-based character movement (Thai) in `XmTextField` or `XmText` widgets, set the translations of the corresponding widgets as follows. Refer to the documentation for further detailed explanation.

```
XmText.translations: #override \n\  
<Key>osfRight:forward-cell() \n\  
<Key>osfLeft:backward-cell() \n\  
<Key>osfDelete:delete-next-cell() \n\  
<Key>osfBackSpace:delete-previous-cell() \n\  

```


Index

Numbers

- 16-bit Unicode 2.0 codeset, 217
- 32-bit STREAMS, 66
- 64-bit STREAMS, 66

A

- adding packages, 95
- addresses, formats, 13
- Adobe Type Manager (ATM) fonts, 44
- alphabets, 10, 11
- APIs, 171, 179
 - using to develop applications, 167
- applications
 - FontSet/XmFontList definitions, 92
 - internationalizing, 92
 - linking to system libraries, 168
 - XPG4, 171
- architectures (SPARC and x86), xvii
- Asian
 - packages, 114
 - printing support, 191
- ATM fonts, 44
- ATOK8, 53
- AttrObject, 199

B

- base language, 3
- base Solaris 2.6, 17, 23

- locales supported, 24
- Bi-directionality, 195
- Big-5
 - codeset, 163
- bin/stty, 70
- /bin/stty directory, 70
- bitmap
 - fonts, 44
- books@sun.com, xviii
- bopomofo in Chinese, 12
- breve, 26

C

- caron, 26
- catgets(), 179
- CD
 - installing software from, 96
- CDE, 183
 - en_US.UTF-8 locale support of, 23
 - input methods, 185
 - localization packages, 152
 - using fonts for locales, 25
- Central European languages, character support, 24
- character classification macros, 171
- character shaping, 195
- character support, 24
- character transformation macros, 171

- characters
 - number, 10
- Chinese
 - package files, 126
- Chinese text
 - bopomofo, 12
 - linguistic introduction, 12
 - pinyin, 12
 - zhuyin, 12
- code conversion STREAMS modules, 66
- code conversions, 71, 74
- codeset, 4
 - Big-5, 163
 - character support, 24
 - Extended UNIX Code (EUC), 163
 - Shift-JIS, 163
- Codeset conversion utilities, 1
- Codeset Independence, 164
- command names,
- command-line placeholder,
- commands
 - CSI-capable, 164
- Common Desktop Environment (book), xix
- Common Desktop Environment
 - Internationalization
 - Programmer's Guide, 183
- complex language shaping, 195
- Complex Text Layout (CTL), 2
 - CTL, 195
- Compose c c sequence, 85
- Compose g g sequence, 86
- Compose Key, 13
- compose sequences
 - Latin-1, 76, 77, 80
 - Latin-2, 81, 82
 - Latin-4, 82 to 84
 - Latin-5, 84, 90
- compose sequences, for new locales, 25
- Context, 200
- conversion
 - multibyte and wide character process
 - code, 171
- conversions, 71, 74
- converting characters, 61
- core locales, 18, 19
- country of use, 3
- creating
 - message catalogs, 179

- Creating Worldwide Software, xviii, 14
- cs00, 53
- .cshrc, 71
- CSI, , *see* Codeset Independence,
- CSI-capable commands, 164
- CSI-enabled libraries, 166
- CSText, 200
- CTL architecture, 196
- ctype
 - macros, 171
- currency, 3
 - presentation order of, 9
 - sizes of, 10
 - units of, 9
- currency symbol, 26
- Cyrillic input mode, 85
- Czech
 - character support, 24
 - keyboards, 25

D

- Date, 3
- date formats, 7
- Daylight Savings Time (DST), 7
- decimal places, 8
- degree symbol, 26
- delimiters
 - numeric, 8
 - thousands, 8
 - word, 10
- descriptions of European package files, 104
- desktop environments, 183
- Desktop Font Downloader, 2
- desktop layers, 183
- deutsche mark, 9
- developer's cluster, in Solaris 2.6, 23, 64
- diacritical marks, 25
 - in English input mode, 75
- diacritics, 195
- diaeresis, 25
- directories,
- disk space
 - Asian packages, 158
- documentation, ordering, xix
- dollar, 9
- doubleacute, 26

- DST (Daylight Savings Time), 7
- Dt Apps, 196
- dtlogin command, 25
- dtmail, 187
- dtterm, 68
- dynamic linking, 168
- dynamic text widgets, 195

E

- Editing behavior, 200
- English
 - character support, 24
 - input mode, 75
 - language locales, 28
- English Solaris 2.6, , see base Solaris 2.6
- en_US.UTF-8
 - code conversions, 71
 - fontset definitions, 92, 94
 - overview, 17, 63
 - printing utility, 90
- Euro currency, 2
- European Codesets, 113
- European font packages, 113
- European printing support, 189
- European Solaris, 2
- extended locales, 22, 23
- Extended UNIX Code (EUC), 163

F

- file code, 164
- file names,
- fonts
 - across different platforms, 184
 - adding or removing, 44
 - formats, 44
 - location, 44
 - packages for Europe, 113
 - SUNiXxf format for new locales, 25
 - X11 bitmaps, 90
- FontSet definitions, 92, 94
- FontSet/XmFontList definitions, 92
- formats
 - addresses, 13
 - currency, 9
 - dates, 7
 - numeric, 8

- time, 6
- franc, 9
- French package files, 99
- Full Solaris locale, 4

G

- gender in language, 13
- genmsg utility, 179, 180
- German
 - character support, 24
 - package files, 100
- GMT offset, 7
- Greek
 - character support, 24
 - input mode, 86
- Greenwich Mean Time offset, 7

H

- Hangul in Korean, 11
- Hanja in Korean, 11
- Hanzi in Chinese, 12
- head side module, 66
- Hiragana in Japanese, 11
- Horizontal Tabs, 220
- Hungarian
 - character support, 24
 - keyboards, 25

I

- IBM DOS 437, 28
- iconv, 37
 - command, 71
 - how to use, 61
 - Japanese character code conversion, 54
- imperial system, 13
- input modes
 - Cyrillic, 85
 - English, 75
 - Greek, 86
- installation, 95, 98
- Internationalization, 2
- internationalization
 - ISO Latin-1, 3
 - Java, 164
- internationalization APIs, 171, 179

- internationalizing applications, 92
- ISO 8859, 63
- ISO 8859-n character support, 24
- ISO Latin-1, 3
- ISO-10646, 1
- Italian package files, 101

J

- ja, 53
- Japanese
 - package files, 132
 - Solaris, 3
- Japanese text
 - Hiragana, 11
 - Kanji, 11
 - Katakana, 11
 - linguistic introduction, 11
- Japanese-specific printer support, 57
- Java internationalization, 164
- ja_JP.PCK, 53
- JLE Binary, 57
- JumpStart, 29

K

- Kanji in Japanese, 11
- Katakana in Japanese, 11
- key compose sequences, 25
- Keyboard Selection, 222
- keyboards, 12
 - Changing keyboards on x86, 27
 - Changing on SPARC, 26
 - Czech, 25
 - Hungarian, 25
 - Latvian, 25
 - Lithuanian, 25
 - Polish, 25
 - Support in Solaris 2.6, 26
 - Turkish, 25
- Korean package files, 125
- Korean Solaris, 3
- Korean text
 - Hangul, 11
 - Hanja, 11
 - linguistic introduction, 11
- krona, 9
- krone, 9

- kroner, 9
- KSC-5700, 46

L

- LANG, 65
- LANG environment variable, 65, 184
- language, 3
- Language Conversion Library, 187
- language engine, 195
- language-dependent rendering, 6
- Latin-1 compose sequences, 76, 77, 80
- Latin-2 compose sequences, 81, 82
- Latin-4 compose sequences, 82 to 84
- Latin-5 compose sequences, 84, 90
- Latin-n terminals, 69
- Latvian keyboards, 25
- Layout behavior, 200
- Layout Direction, 215
- Layout Modifier Orientation, 199
- Layout Services, 199
- layoutDirection, 216
- LayoutObject, 197, 199
- LCL, 187
- LC_ALL, 4
- LC_COLLATE, 5, 6
- LC_CTYPE, 5
- LC_MESSAGES, 6
- LC_MONETARY, 5
- LC_NUMERIC, 5
- LC_TIME, 5
- left-character(), 203
- libc, 168, 171
- libintl, 169
- libraries, linking applications to, 168
- libw, 169
- Ligation, 195
- ligatures, 195
- linking applications, 168
- lira, 9
- list separators, 9
- Lithuanian keyboards, 25
- loading
 - STREAMS modules, 67, 68
- locale, 3
- locale utility, 65
- locale(1), 65

- locales, 2 to 4, 24
 - categories of, 5
 - compose sequences, 25
 - core, 17 to 19
 - database, 163, 167
 - environment variables, 65, 184
 - extended, 17, 22, 23
 - font format, 25
 - full, 4
 - operating system, 17
 - partial, 4, 17
 - what is..., 3
 - window system, 17
- localization, 2
- localization resource category, 182
- LO_LTYPE, 6
- lpadmin command, 90
- lpfilter command, 90
- lpr command, 90
- @ls numerals=:national, 200
- @ls numerals=:nominal:national, 200

M

- macros
 - ctype, 171
- mail interchange, 187
- markka, 9
- mbtowcs, 171
- mbtowc, 171
- message catalogs, creating, 179
- metric system, 13
- modinfo command, 67
- modload command, 68
- Motif 2.1, 2
- Mouse Selection, 221
- mp(1), 189
- multi-byte Unicode representation, 17
- multibyte file code, 171
- multiple input, 1
- mystreams file, 71
- m_create_layout(), 199

N

- NULL (0x00), 164
- number of characters, 10
- Numbers, 8

- Numeral shaping, 195
- numeral shaping, 196
- Numerals, 217
- NUMERALS_CONTEXTUAL, 217
- NUMERALS_NATIONAL, 200
- NUMERALS_NOMINAL, 200
- numeric conventions, 8

O

- ogonek, 26
- OLIT Reference Manual, xix
- on-screen computer output,
- OpenWindows
 - changes, 188
 - using fonts for locales, 25
- operating system locale, 17
- order for sorting, 10
- ordering documentation, xix
- Orientation, 200
- OSF/Motif Programmer's Guide, xix
- OSF/Motif Programmer's Reference, xix
- outline fonts, 44
- OutToInp, 202

P

- packages
 - adding, 95
- Page Description Language (PDL)
 - interpreters, 182
- page sizes, 14
- paper sizes, 14
- partial locales, 17
- Partial Solaris locale, 4
- PDL interpreters, 182
- People's Republic of China, 2, 12
- peseta, 9
- pinyin in Chinese, 12
- pkgadd command, 95
- pkgchk command, 96
- PLS, 195
- Polish
 - character support, 24
 - keyboards, 25
- Portable Layout Services (PLS)
 - PLS, 195

- Porting Instructions, 223
- positional variation, 196
- POSIX, 183
- postprint(1), 189
- PostScript printer, 2
- PostScript, 44, 182
 - output, 90
 - support under Solaris, 189
 - Type 1 fonts, 44
- PostScript Language Reference Manual, xix, 182
- PostScript Language Reference Manual Supplement, xix, 182
- pound, 9
- printing, 90
- printing support
 - Asian, 191
 - European, 189
 - Japanese, 57
- Programming the Display PostScript System with X, xix, 182
- Property, 202
- pseudo-XOC, 197
- punctuation, 13

R

- radix, 8
- radix characters, 8
- region, 3
- remote package server
 - installing software from, 97, 98
- Render Table, 218
- Rendition, 217
- Russian
 - character support, 24

S

- saving
 - STREAMS modules settings, 71
- sbin/sh, 168
- /sbin/sh command, 168
- Scandinavian and Baltic language character support, 24
- script selection, 75
- segment ordering, 195
- separators

- list, 9
- thousands, 8
- word, 10
- setenv command, 65
- setlocale man page, 65
- setting
 - terminal options, 70
- setup
 - TTY environment, 66
- ShapeCharset, 200, 217
- Shift-JIS codeset, 163
- shortcuts, , *see* compose sequences,
- Simple Mail Transfer Protocol, 187
- single-display clients, 184
- Slash (0x2f), 164
- Smallberg, David, xviii, 14
- SMTP, 187
- Solaris
 - Asian, 44
 - Austrian, 36
 - base product, 17, 23
 - Chinese, 49
 - contents, 31
 - Czech, 36
 - Eastern European, 3
 - English, 31
 - Estonian, 36
 - European, 31
 - French, 2, 31
 - German, 2, 31
 - Greek, 36
 - Hungarian, 36
 - Italian, 2, 31
 - Japanese, 3, 53
 - Japanese printing support, 191
 - Korean, 3, 45
 - Latvian, 36
 - Lithuanian, 36
 - localized products in, 2
 - Polish, 36
 - PostScript support, 189
 - Russian, 36
 - Simplified Chinese, 3
 - Spanish, 2, 31
 - Swedish, 2, 31
 - Traditional Chinese, 3
 - Turkish, 36

- sort order, 10
- Spanish
 - character support, 24
 - package files, 102
- SPARC architecture, xvii
- SPARC keyboards, 26
- standalone system
 - adding packages to, 95, 96
- standards
 - interface, 183
 - internationalization, 183
- stateless file code encodings, 164
- static and dynamic text, 195
- static linking, 168
- strchg command, 69
- strconf command, 70
- STREAMS modules
 - loading, 67, 68
 - saving settings, 71
- String validation, 196
- String XmTextFieldGetLayoutModifier, 211
- stty command, 71
- stub entry points, in libw and libintl, 169
- su command, 67
- SunDocs program, xix
- SUNWpldte, 25
- SUNWploc, 17
- SUNWploc1, 17, 25
- SUNWplow, 17
- SUNWplow1, 17, 25
- Swedish package files, 103
- symbols, 13
- Symmetrical swapping, 195
- system libraries
 - linking applications to, 168

T

- tabbing, 195
- Tabbing Behavior, 221
- tail side module, 66
- terminal options, setting, 70
- terminal support for Latin-1, Latin-2, or KOI8-R, 69
- terminals
 - Latin-n, 69
 - Latin-n terminals, 69
- text orientation, 195

- text rendering, 195
- Text Resources and Geometry, 222
- TextField, 216, 213
- TextShaping, 200
- Thai text, 11
- thousands separators, 8
- time, 3
- Time Formats, 6
- time zones, 7
- titles,
 - titles in language, 13
- TTY environment setup, 66
- TTY STREAMS, 57
- Turkish
 - character support, 24
 - keyboards, 25
- Tuthill, Bill, xviii, 14
- Type 1 fonts, 44
- TypeOfText, 200

U

- u8lat1 STREAMS module, 69
- u8lat2 STREAMS module, 69
- UIL, 215
- Unicode 2.0, 1, 17
 - support, 1
- Universal Character Set Transformation
 - Format for 8 bits encoding, ,
 - see* UTF-8 encoding,
- user type,
- usr/bin/ldd, 168
- usr/ucb/stty, 70
- /usr/ucb/stty directory, 70
- UTF-8, 1
- UTF-8 encoding, 23
- utilities
 - genmsg, 179, 180
 - locale, 65
 - printing, 90

W

- wcstombs, 171
- wctomb, 171
- Western European alphabets, 11

Western European languages, character support, 24

wide character

expression, 163

process code, 171

window system locale, 17

Wnn6, 53

words

delimiters, 10

order of, 10, 28

X

X Display PostScript, 182

X Window System, 182

X/DPS, 181

X/Open-Uniform Joint Internationalization Working Group, 23

X11 bitmap fonts, 90

x86

architecture, xvii

keyboards, 27

xetops, 191

XFontStruc, 185

Xlib dependencies, 186

XmALIGNMENT_CENTER, 201

XmALIGNMENT_END, 201

XmCR_MOVING_INSERT_CURSOR, 203, 204

XmDEFAULT_DIRECTION, 197

XmDirection, 198, 215

XmEDIT_LOGICAL, 201, 204, 205, 221

XmEDIT_VISUAL, 201, 204, 221

XmFontSet, 185

XmFont_IS_XO, 199

XmFONT_IS_XOC, 199, 219

XmLabel, 197, 217

XmLabelG, 197

XmList, 197

XmNalignment, 201, 216

XmNAlignment, 217

XmNeditPolicy, 201, 204, 221

XmNfont, 199

XmNfontName, 199

XmNfontType, 199

XmNgainPrimaryCallback, 204, 205

XmNlabelString, 217

XmNlayoutAttrObject, 199

XmNlayoutDirection, 197 to 199, 214, 216

XmNlayoutModifier, 197, 199, 200, 216, 217

XmNmotionVerifyCallback, 203, 204

XmNrenderTable, 201, 222

XmNrenditionTag, 201

XmRenderTableAddRenditions, 219

XmRendition, 197 to 201, 216, 215

XmRendition{Retrieve,Update}, 199

XmString, 198, 215

XmStringDirection, 198, 214

XmStringDirectionCreate, 214

XmSTRING_COMPONENT_DIRECTION, 197

XmSTRING_COMPONENT_LAYOUT_PUSH, 197

XmSTRING_COMPONENT_LOCALE_TEXT, 197

XmSTRING_COMPONENT_TEXT, 197

XmSTRING_COMPONENT_WIDECHAR_TEXT, 197

XmText, 197, 200, 201, 216

XmTextField, 197, 200, 201, 212

XmTextFieldGetLayoutModifier, 212

XmTextFieldSetLayoutModifier, 213

XmTextGetLayoutModifier, 212

XmTextSetLayoutModifier, 214

XoJIG, 23

XPG4 applications, 171

xutops, 191

xutops utility, 90

XView Developer's Notes, xix

XView toolkit, 188

Y

yen, 9

Z

zh.GBK, 2

zhuyin in Chinese, 12