International Language Environments Guide

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

Part No: 806-0169–10
February 2000

Adobe PostScript

Please Recycle

Contents

Tables

Figures

Preface

The *International Language Environments Guide* describes internationalization features that are new in the Solaris™ 8 operating environment. It contains important information on how to use this release 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 this release to write applications for international markets.
- An overall view of internationalization topics that apply to various layers within the Solaris operating 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 and administrators who want to design global products and applications for the Solaris 8 operating environment.

This guide assumes knowledge of the C programming language.

All operating system information pertains to the Solaris 8 SunOS™ 5.8 operating environment.

How this Guide is Organized

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 (English) and the localized multi-lingual Solaris releases.
- Chapter 2 describes the internationalization framework in the Solaris 8 product.
- Chapter 3 describes the contents of the Solaris 8 localized product.
- Chapter 4 covers the en_US.UTF-8 locales and the internationalization features incorporated into this release.
- Chapter 5 contains a detailed look at the procedures to write a localized version of codesets, formats, collation, and messaging. X/DPS
- Chapter 6 explains 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 7 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.
- Chapter 8 explains printing support under the Solaris 8 operating environment, with specific information for European and Asian printing.
- Appendix A contains lists of tables of available iconv Conversions between UTF-8 and UTF-EBCDIC
- Appendix B contains a table of the partial L10N package Names on the OS CD.
- Appendix C contains tables representing the language packages on the language CD. There are tables for Simplified Chinese, French, German, Italian, Japanese, Korean, Spanish, Swedish, Traditional Chinese, and Shared.

Related Books and Sites

For information about the Java Development Kit, see
`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 environment.

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.

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.

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

The docs.sun.com℠ Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

What Typographic Conventions Mean

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

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

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

CHAPTER 1

Solaris Internationalization Overview

The Solaris 8 product includes full Unicode 3.0 support, as defined in ISO-10646, for selected locales. The Solaris 8 release is a major release for Sun's international markets. It includes a number of new features. All partial locales including multibyte locales such as Japanese locales are now available on the Base Solaris 8 product.

New Internationalization and Localization Features in Solaris 8

- Simplified Chinese UTF-8 locale. This provides broader support for Unicode with the addition of new UTF-8 locales. Unicode is often used in a mixed script environment, where it is necessary to display text from multiple languages in a single environment.
- Traditional Chinese UTF-8 locale
- Asian printing enhancements
- Support for 90 locales on the base Solaris CD. This is a new packaging approach to universal language coverage.
- Enhanced Sdtudctool —support for migration of UDC (User Defined Character) from Microsoft Windows. Localized for all Asian locales.
- Three additional locales have been added for Iceland (ISO8859–1), U.S.A. (ISO8859–15), and Russia (ANSI1251). The new U.S.A. locale adds support for the euro currency glyph. The new Russian locale is in addition to the existing ISO8859–5 and KOI8–R locales. It provides native Microsoft data encoding support. The new ISO8859–1 locale for Iceland marks the introduction of Icelandic support to the Solaris environment.

- Customer-extensible codeset conversion. New codeset conversion can be added by using the `geniconvtbl` utility. Existing codeset conversions can be modified.
- European locale repackaging
- Euro font
- Adding Japanese `iconv` modules — conversions for IBM mainframe codesets and conversions between Unicode and Shift-JIS for Microsoft codeset.
- Euro currency. All foreign exchange, banking, and finance industries in the European community are converting from using their local currencies to using the Euro. Euro currency support has been enhanced in the Solaris 8 environment with the addition of U.S. and Estonian ISO8859–15 locales.
- Multibyte Partial locale — framework of multibyte locale support is included in the Base Solaris product.
- Enhanced Unicode `iconv` modules. The `iconv` module has been enhanced for various Unicode encoding formats and international and *de facto* industry standard codesets.

Internationalization and Localization Defined

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 runtime in accordance with specific cultural requirements. Localization involves establishing online 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.

Internationalization and localization are different procedures.

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

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 with an environment that can be modified dynamically without the necessity of recompiling the software.
- Divide 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 can 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. 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 can be composed of both a base language, the country (territory) of use, and possibly codeset (which 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. This allows for specific differences by country, such as currency units notation.

More than one locale can be associated with a particular language, which allows for regional differences. For example, an English-speaking user in the United States can select the `en_US` locale (English for the United States), while an English-speaking user in Great Britain can select `en_GB` (English for Great Britain).

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 country-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.

Generally the locale name is specified by the `LANG` environment variable. Locale categories are subordinate to `LANG`, but can 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 the relevant language. If no localized messages are installed, then all locales would be classified as “partial locales”. Several locales in the Solaris environment are capable of displaying localized messages, provided that the relevant language support is installed. For example, there are several locales which can use German messages:

- de_DE.ISO8859–1
- de_DE.ISO8859–15
- de_DE.UTF-8
- de_AT.ISO8859–1
- de_AT.ISO8859–15
- de_CH.ISO8859–1

When the German messages translations are installed using the Language CD, all of the above locales will become “full locales”, because they will have access to a fully translated desktop. The language CD contains message translations for the following languages:

- German
- French
- Spanish
- Swedish
- Italian
- Japanese
- Korean
- Simplified Chinese
- Traditional Chinese

All partial locales are also available in the base product, but message translations are available only in the multilingual Solaris product.

Cultural Conventions

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 layout engine that 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 might 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
Thai	11:59 PM
U.K.	11.59 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, Nekeybfoundland 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. Notice that even within a country, there can be variations.

TABLE 1–2 International Date Formats

Locale	Convention	Example
Canadian (English and French)	yyyy-mm-dd	1998-08-13
Danish	yyyy-mm-dd	1999–08–24
Finnish	dd.mm.yyyy	13.08.1998
French	dd/mm/yyyy	13/08/1999
German	yyyy-mm-dd	1999–09–18
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
GB-English	dd/mm/yy	13/08/98
US-English	mm-dd-yy	08-13-98
Thai	dd/mm/yyyy	10/12/2009

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 groups of thousands, 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)	4 294 967 295,000
Danish	4 294 967 295,000
Finnish	4 294 967 295,000
French	4 294 967 295,000
German	4 294 967.295,000
Italian	4.294.967.295,000
Norwegian	4.294.967.295,000
Spanish	4.294.967.295,000
Swedish	4 294 967 295,000
GB-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 are frequently 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 Great Britain 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)	1.234,56 mk
French	Franc (F)	1 234,56 F
German	Deutsche Mark (DM)	DM 1.234,56
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)	1.234,56 kr
GB-English	Pound	31,234.56 pounds
US-English	Dollar ($)	$1,234.56
Thai	Baht	2539 Baht
Euro	EUR	400,00 €

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.

Euro locales are based on the ISO8859–15 character set. See "European Localization" on page 61 for available locales.

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

Language 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.

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 that 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.

Kanji has tens of thousands of 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 completely avoid the use of Kanji, 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 consist of 19 consonants, 21 vowels, and an optional 27 consonants. About 3,000 Hangul characters from the entire 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, 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 are in the same cell. Otherwise, they are 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. A single sound can be represented by many different characters, which are not interchangeable in usage. A single character can 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.

Several phonetic systems represent 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. You need to provide space 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.

The Compose key can be used with English or European locales, but not with Korean, Chinese, or Japanese locales, except the UTF-8 locales.

Other Differences

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 16 for a full citation.

CHAPTER 2

Internationalization Framework in the Solaris 8 Environment

This section discusses several internationalization features contained in the Solaris 8 environment.

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

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

- Dynamically linked applications
- Solaris 8 internationalized APIs

Support for Codeset Independence

The Solaris 8 operating environment supports non-EUC encodings 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 8 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-Kanji, and Big-5.

The CSI Approach

Codeset Independence enables 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 8:

- 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 2–1 contains CSI-enabled commands in Solaris 8. These commands are marked with CSI capabilities on their man page.

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

TABLE 2–1 CSI-enabled Commands in Solaris 8

`/usr/lib/diffh`	`acctcom`	`gencat`	`script`

TABLE 2–1 CSI-enabled Commands in Solaris 8 *(continued)*

/usr/sbin/accept	apropos	getopt	sdiff
/usr/sbin/reject	batch	getoptcvt	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
/usr/xpg4/bin/mv	diff3	priocntl	write
/usr/xpg4/bin/nice	disable	ps	xargs
/usr/xpg4/bin/nohup	echo	pwd	zcat
/usr/xpg4/bin/od	expand	rcp	
/usr/xpg4/bin/pr	file	red	
/usr/xpg4/bin/rm	fine	remsh	
/usr/xpg4/bin/sed	fold	rksh	
/usr/xpg4/bin/sort	ftp	rmdir	
/usr/xpg4/bin/tail		rsh	
/usr/xpg4/bin/tr			

TABLE 2–1 CSI-enabled Commands in Solaris 8 *(continued)*

/usr/xpg4/bin/vedit
/usr/xpg4/bin/vi
/usr/xpg4/bin/view

Solaris 8 CSI-enabled Libraries

Nearly all functions in Solaris 8 `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()` `eucscol()` `getwidth()`

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

- `csetno()` `wcsetno()`

In the Solaris 8 product, `libgen` (`/usr/ccs/lib/libgen.a`) are internationalized, but not CSI enabled.

In the Solaris 8 product, `libcurses` (`/usr/ccs/lib/libcurses.a`) are internationalized, but not CSI enabled.

Here are the five deliverables:

- The utility (32-bit application):

  `/usr/bin/geniconvtbl`
- special `iconv` shared objects:

  `/usr/lib/iconv/geniconvtbl.so`

  `/usr/lib/iconv/sparcv9/geniconvtbl.so`
- Sample `geniconvtbl(1)` input source files and system-provided binary table files :

  `/usr/lib/iconv/geniconvtbl/srcs/`

  ISO8859-1_to_ISO646.txt

  ISO646_to_ISO8859-1.txt

  ISO8859-1_to_UTF-8.txt

  UTF-8_to_ISO8859-1.txt

  ShiftJIS_to_eucJP.txt

  eucJP_to_ShiftJIS.txt

/usr/lib/iconv/geniconvtbl/binarytables/

ISO8859-1%ISO646.bt

ISO646%ISO8859-1.bt

- Changed iconv_open(3) at libc.so.1s:

  /usr/lib/libc.so.1

  /usr/lib/sparcv9/libc.so.1 (sparcv9 example)

- Man pages:

  /usr/share/man/sman1/geniconvtbl.1

  /usr/share/man/sman4/geniconvtbl.4

The section for geniconvtbl(1) describes how to use the utility and where to place the generated binary table files so that they can be used by the iconv functions and utilities.

See geniconvtbl(4)

Locale Database

The locale database format and structure 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.

When using Solaris 8, use the locale databases that are included with the Solaris 8 product. Do not use locales from previous Solaris versions.

Process Code Format

The process code format in the Solaris 8 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.

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 enables 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 8 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 might 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`

These interfaces have moved to `libc` and are no longer in `libw` and `libintl`.

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 2–2 shows the stub entry points in `libw` and `libintl`.

TABLE 2–2 Stub Entry Points in `libw` and `libintl`

`libw:`	`fgetwc`	`fgetws`	`fputwc`	`fputws`	`getwc`
	`getwchar`	`getws`	`isenglish`	`isideogram`	`isnumber`
	`isphonogram`	`isspecial`	`iswalnum`	`iswalpha`	`iswcntrl`
	`iswctype`	`iswdigit`	`iswgraph`	`iswlower`	`iswprint`
	`iswpunct`	`iswspace`	`iswupper`	`iswxdigit`	`putwc`
	`putwchar`	`putws`	`strtows`	`towlower`	`towupper`
	`ungetwc`	`watoll`	`wcscat`	`wcschr`	`wcscmp`
	`wcscoll`	`wcscpy`	`wcscspn`	`wcsftime`	`wcslen`
	`wcsncat`	`wcsncmp`	`wcsncpy`	`wcspbrk`	`wcsrchr`
	`wcsspn`	`wcstod`	`wcstok`	`wcstol`	`wcstoul`
	`wcswcs`	`wcswidth`	`wcsxfrm`	`wctype`	`wcwidth`
	`wscasecmp`	`wscat`	`wschr`	`wscmp`	`wscol`
	`wscoll`	`wscpy`	`wscspn`	`wsdup`	`wslen`
	`wsncasecmp`	`wsncat`	`wsncmp`	`wsncpy`	`wspbrk`
	`wsprintf`	`wsrchr`	`wsscanf`	`wsspn`	`wstod`
	`wstok`	`wstol`	`wstoll`	`wstostr`	`wsxfrm`
`libintl:`	`bindtextdomain`	`dcgettext`	`dgettext`	`gettext`	`textdomain`

`ctype` Macros

Character classification and character transformation macros are defined in `/usr/include/ctype.h`. The Solaris 8 environment provides a new set of `ctype` 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.
- `_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 8 functions also support XPG4 semantics.

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

Internationalization APIs in libc

Solaris 8 offers two sets of APIs:

- Multibye (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 mbtwoc and mbtowcs APIs. To convert the file output data from wide character format into multibyte format, use the wcstombs and wctomb APIs.

Table 2–3 shows a list of internationalization APIs included in Solaris 8.

TABLE 2–3 Internationalization APIs in libc

API Type	Library Routine	Description
Messaging functions		
	catclose()	Close a message catalog.
	catgets()	Read a program message.
	catopen()	Open a message catalog.
	dgettext()	Get a message from a message catalog with domain specified.
	dcgettext()	Get a message from a message catalog with domain and category specified.
	textdomain()	Set and query the current domain.
	bindtextdomain()	Bind the path for a message domain.
Code conversion		

TABLE 2–3 Internationalization APIs in `libc` *(continued)*

API Type	Library Routine	Description
	`iconv()`	Convert codes.
	`iconv_close()`	Deallocate the conversion descriptor.
	`iconv_open()`	Allocate the conversion descriptor.
Regular expression		
	`regcomp()`	Compile the regular expression.
	`regexec()`	Execute regular expression matching.
	`regerror()`	Provide a mapping from error codes to error message.
	`regfree()`	Free memory allocated by `regcomp()`.
Wide character class		
	`wctype()`	Define character class.
	`wctrans`	Define character mapping.
	`towctrans`	Wide-character mapping.
	`setlocale()`	Modify and query a program's locale.
	`nl_langinfo()`	Get language and cultural information of current locale.
	`localeconv()`	Get monetary and numeric formatting information of current locale.
Character classification		
	`isalpha()`	Is character alphabetic?
	`isupper()`	Is character uppercase?
	`islower()`	Is character lowercase?
	`isdigit()`	Is character a digit?
	`isxdigit()`	Is character a hex digit?
	`isalnum()`	Is character alphabetic or digital?
	`isspace()`	Is character a space?

TABLE 2–3 Internationalization APIs in `libc` *(continued)*

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

TABLE 2–3 Internationalization APIs in `libc` *(continued)*

API Type	Library Routine	Description
	`towupper()`	Convert a lowercase wide character to uppercase.
	`towlower()`	Convert an uppercase wide character to lowercase.
Character collation		
	`strcoll()`	Collate character strings.
	`strxfrm()`	Transform character strings for comparison.
	`wcscoll()`	Collate wide character strings.
	`wcsxfrm()`	Transform wide character strings for comparison.
Monetary handling		
	`strfmon()`	Convert monetary value to string representation.
Date and time handling		
	`getdate()`	Convert user format date and time.
	`strftime()`	Convert date and time to string representation. The %u conversion function conforms to the X/Open CAE Specification, System Interfaces and Headers, Issue 4, Version 2. This function represents a weekday as a decimal number [1,7], with 1 now representing Monday.
	`strptime()`	Date and time conversion.
Multibyte handling		
	`btowc`	Single-byte to wide-character conversion.
	`mbrlen()`	Get number of bytes in character (restartable).
	`mbsinit()`	Determine conversion object status.
	`mbtowc()`	Convert a character to a wide-character code (restartable).
	`mbstowcs()`	Convert a character string to a wide-character string (restartable).
Wide characters		

TABLE 2–3 Internationalization APIs in `libc` *(continued)*

API Type	Library Routine	Description
	`wcsncat()`	Concatenate wide-character strings to length *n*.
	`wsdup()`	Duplicate wide-character string.
	`wcscmp()`	Compare wide-character strings.
	`wcsncmp()`	Compare wide-character strings to length *n*.
	`wcscpy()`	Copy wide-character strings.
	`wcsncpy()`	Copy wide-character strings to length *n*.
	`wcschr()`	Find character in wide-character string.
	`wcsrchr()`	Find character in wide-character string from right.
	`wcslen()`	Get length of wide-character string.
	`wscol()`	Return display width of wide-character string.
	`wcsspn()`	Return span of one wide-character string in another.
	`wcscspn()`	Return span of one wide-character string not in another.
	`wcspbrk()`	Return pointer to one wide-character string in another.
	`wcstok()`	Move token through wide-character string.
	`wcswcs()`	Find string in wide-character string.
	`wcstombs()`	Convert wide-character string to multibyte string.
	`wctomb()`	Convert wide-character to multibyte character.
	`wcwidth()`	Determine number of column positions of a wide character.
	`wcswidth()`	Determine number of column positions of a wide-character string.
	`wctob`	Wide-character to single-byte conversion.
	`wcrtomb`	Convert a wide-character code to a character (restartable).

TABLE 2–3 Internationalization APIs in `libc` *(continued)*

API Type	Library Routine	Description
	`wcsrtombs`	Interpret wide-character string according to format.
Wide formatting		
	`wsprintf()`	Generate wide-character string according to format.
	`wsscanf()`	Formatted input conversion.
	`fwprintf`	Print formatted wide-character output.
	`fwscanf`	Convert formatted wide-character input.
	`wprintf`	Print formatted wide-character output.
	`wscanf`	Convert formatted wide-character input.
	`swprintf`	Print formatted wide-character output.
	`swscanf`	Convert formatted wide-character input.
	`vfwprintf`	Wide-character formatted output of a `stdarg` argument list.
	`vswprintf`	Wide-character formatted output of a `stdarg` argument list.
Wide numbers		
	`wcstol()`	Convert wide-character string to long integer.
	`wcstoul()`	Convert wide-character string to unsigned long integer.
	`wcstod()`	Convert wide-character string to double precision.
Wide strings		
	`wscasecmp()`	Compare wide-character strings, ignore case differences.
	`wsncasecmp()`	Process code-string operations.
	`wcsstr`	Find a wide-character substring.
	`wmemchr`	Find a wide-character in memory.
	`wmemcmp`	Compare wide-characters in memory.
	`wmemcpy`	Copy wide-characters in memory.

TABLE 2–3 Internationalization APIs in `libc` *(continued)*

API Type	Library Routine	Description
	`wmemmove`	Copy wide-characters in memory with overlapping areas.
	`wmemset`	Set wide-characters in memory.
Wide standard I/O		
	`fgetwc()`	Get multibyte character from stream, convert to wide character.
	`getwchar()`	Get multibyte character from `stdin`, convert to wide character.
	`fgetws()`	Get multibyte string from stream, convert to wide character.
	`getws()`	Get multibyte string from `stdin`, convert to wide character.
	`fputwc()`	Convert wide character to multibyte character, puts to stream.
	`fwide`	Set stream orientation.
	`putwchar()`	Convert wide character to multibyte character, puts to `stdin`.
	`fputws()`	Convert wide character to multibyte string, puts to stream.
	`putws()`	Convert wide character to multibyte string, puts to `stdin`.
	`ungetwc()`	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:

(continued)

```
% 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"));
...
% 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.

CHAPTER 3

Contents of Solaris 8 Products

Overview of the Solaris 8 Locales

Multiple environments exist within the Solaris operating system for support of different national languages. Each of these national environments is called a locale, which considers the language, its characters, fonts, and the customs used to input and format data.

A locale defines the behavior of a program at run time according to the language and cultural conventions of a user's geographical area. Throughout the system, locales affect 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

Summary of the Solaris 8 Locale

All Solaris 8 locale packages are classified into two categories. The first category is for partial locales, which are the enablers of the locales. With partial locales installed on the system, users can run applications on the target locales, while the OS/GUI messages from Solaris are English. All partial locale packages are available on the Solaris OS CDs.

The second category is for full locale packages. These packages include translations of software messages, on-line help files, optional fonts, and language specific features. Full locale packages provide the full set of language features to 9 languages.

- German
- French
- Spanish
- Swedish
- Italian
- Japanese
- Korean
- Simplified Chinese
- Traditional Chinese

Full locale packages are available on the languages CD. Partial locale packages (locale enablers) have to be installed in order for the full locales to be functional.

Localization Content on Solaris 8 CD-ROMs

Partial locales are selected at the beginning of the install procedure on the OS CD-ROM. Full locales are automatically installed from the Language CD-ROM according to the locale selections made at the beginning of the install procedure.

The distribution of locales is shown in the table below.

TABLE 3–1 Solaris 8 Installation CD-ROMs

Disk	Contents
Solaris OS CD-ROM	Solaris 8 Operating System all partial locales
Language CD-ROM	message translations for 9 languages locale specific utilities

As mentioned, the locales include partial locales. These are based on core locales for the main language. For example, the `fr_CA.ISO8859-1` (French Canadian) is based on the `fr_FR.ISO8859-1` (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 might contain only English messages.

Localization Functions in Solaris Interfaces

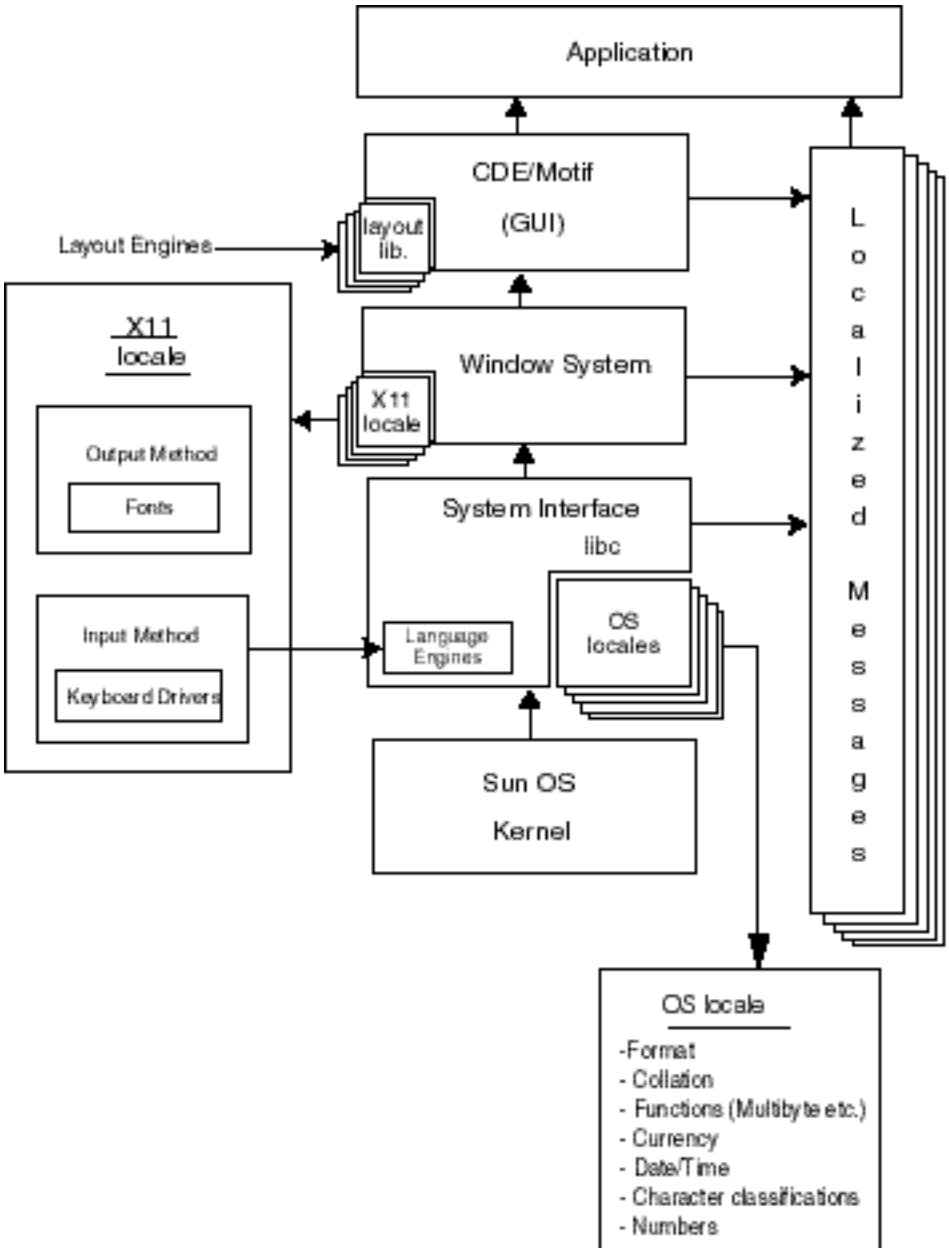
The OS locale layer provides the basic locale database and functions that are plugged into the OS system interface at the application's run time. Applications will access these OS locale modules through standard APIs as described in Chapter 2.

The X11 locale layer provides the interface to X input method and X output method such that the X11 applications can allow local text input and display. Fonts are provided to allow applications to display characters from various languages.

CDE/Motif is built on top of the X11 window system. Hence, it can utilize the X11 locale capability through X11 APIs. Solaris localizations have various locale-specific configurations for CDE applications, in order to make the desktop functional within the target locale.

Message translations and on-line help contents are provided throughout different layers as described in the following diagram.

Figure 3–1 Tabbing Behavior



Script Enabling for Solaris 8

The Solaris 8 base product provides multiple levels of script enabling, such as simple ASCII support, Latin/European support, Asian multibyte support, and Arabic/Hebrew bidirectional support.

The interfaces defined within the X/Open specification are capable of supporting a large set of languages and territories, including the following types of script:

Script	Description
Latin Language	Americas, Eastern/Western Europe, Turkey
Greek	Greece
East Asia	Japanese, Korean, and Chinese
Indic	Thai
Bidirectional	Arabic and Hebrew
Cyrillic	Russian

Localization in the Base and Multilingual Solaris Product

The base Solaris 8 product includes all partial locales, (including multibyte locales) which provide the functionality needed to input, display, and print text in their target languages while using English user interfaces.

The multilingual Solaris 8 product is a super set of the base Solaris product. It additionally includes 9 language translations (user interface and documentation) and some additional software such as BCP support, optional fonts, and optional utilities on the Language CD.

The English Unicode locale (`en_US.UTF-8`) is installed as the default, while other locales are installed when the locale is selected as `install locale` during the Solaris install process. Since the `UTF-8` locales require all the languages fonts, basic fonts supporting all languages are also installed as the default.

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` encompasses almost all of the characters for traditional single-byte and multi-byte locales for European and Asian languages for Solaris locales.

Additional locale support is packaged according to the geographic region which they support. During the Solaris install process, you are prompted to choose which geographic regions require your support. The locale support available after installation has finished depends on the choices made at this stage.

The following tables lists all the locales supported by the Solaris 8 environment. The locale names have been updated from the Solaris 7 environment in keeping with international naming standards.

All of these locales are also present in the base Solaris 8 release.

TABLE 3–2 Asia

Locale	User Interface	Territory	Codeset	Language Support
ja	Japanese	Japan	eucJP	Japanese (EUC) JISX0201-1976 JISX0208-1990 JISX0212-1990
ja_JP.PCK	Japanese	Japan	PCK	Japanese (PC kanji) JISX0201-1976 JISX0208-1990
ja_JP.UTF-8	Japanese	Japan	UTF-8	Japanese (UTF-8) Unicode 3.0
ko	Korean	Korea	5601	Korean (EUC) KSC 5601-1987
ko.UTF-8	Korean	Korea	UTF-8	Korean (UTF-8) KSC Unicode 3.0
th	English	Thailand	TIS620.2533	Thai TIS620.2533
zh	Simplified Chinese	PRC	gb2312	Simplified Chinese (EUC) GB2312-1980
zh.GBK	Simplified Chinese	PRC	GBK	Simplified Chinese (GBK) GBK
zh.UTF-8	Simplified Chinese	PRC	UTF-8	Simplified Chinese (UTF-8) Unicode 3.0
zh_TW	Traditional Chinese	Taiwan	cns11643	Traditional Chinese (EUC) CNS 11643-1992

TABLE 3–2 Asia *(continued)*

Locale	User Interface	Territory	Codeset	Language Support
zh_TW.BIG5	Traditional Chinese	Taiwan	BIG5	Traditional Chinese (BIG5) BIG5
zh_TW.UTF-8	Traditional Chinese	Taiwan	UTF-8	Traditional Chinese (UTF-8) Unicde 3.0

TABLE 3–3 Australasia

Locale	User Interface	Territory	Codeset	Language Support
en_AU.ISO8859-1	English	Australia	ISO8859-1	English (Australia)
en_NZ.ISO8859-1	English	New Zealand	ISO8859-1	English (New Zealand)

TABLE 3–4 Central America

Locale	User Interface	Territory	Codeset	Language Support
es_CR.ISO8859-1	Spanish	Costa Rica	ISO8859-1	Spanish (Costa Rica)
es_GT.ISO8859-1	Spanish	Guatemala	ISO8859-1	Spanish (Guatemala)
es_MX.ISO8859-1	Spanish	Mexico	ISO8859-1	Spanish (Mexico)
es_NI.ISO8859-1	Spanish	Nicaragua	ISO8859-1	Spanish (Nicaragua)
es_PA.ISO8859-1	Spanish	Panama	ISO8859-1	Spanish (Panama)
es_SV.ISO8859-1	Spanish	El Salvador	ISO8859-1	Spanish (El Salvador)

TABLE 3–5 Central Europe

Locale	User Interface	Territory	Codeset	Language Support
cs_CZ.ISO8859-2	English	Czech Republic	ISO8859-2	Czech (Czech Republic)
de_AT.ISO8859-1	German	Austria	ISO8859-1	German (Austria)
de_AT.ISO8859-15	German	Austria	ISO8859-15	German (Austria, ISO8859-15 - Euro)
de_CH.ISO8859-1	German	Switzerland	ISO8859-1	German (Switzerland)
de_DE.UTF-8	German	Germany	UTF-8	German (Germany, Unicode 3.0)
de_DE.ISO8859-1	German	Germany	ISO8859-1	German (Germany)
de_DE.ISO8859-15	German	Germany	ISO8859-15	German (Germany, ISO8859-15 - Euro)
fr_CH.ISO8859-1	French	Switzerland	ISO8859-1	German (Switzerland)
hu_HU.ISO8859-2	English	Hungary	ISO8859-2	Hungarian (Hungary)
pl_PL.ISO8859-2	English	Poland	ISO8859-2	Polish (Poland)
sk_SK.ISO8859-2	English	Slovakia	ISO8859-2	Slovak (Slovakia)

TABLE 3–6 Eastern Europe

Locale	User Interface	Territory	Codeset	Language Support
bg_BG.ISO8859-5	English	Bulgaria	ISO8859-5	Bulgarian (Bulgaria)
et_EE.ISO8859-15	English	Estonia	ISO8859-15	Estonian (Estonia)
hr_HR.ISO8859-2	English	Croatia	ISO8859-2	Croatian (Croatia)
lt_LT.ISO8859-13	English	Lithuania	ISO8859-13	Lithuanian (Lithuania)
lv_LV.ISO8859-13	English	Latvia	ISO8859-13	Latvian (Latvia)
mk_MK.ISO8859-5	English	Macedonia	ISO8859-5	Macedonian (Macedonia)
ro_RO.ISO8859-2	English	Romania	ISO8859-2	Romanian (Romania)

TABLE 3–6 Eastern Europe *(continued)*

Locale	User Interface	Territory	Codeset	Language Support
ru_RU.KOI8-R	English	Russia	KOI8-R	Russian (Russia, KOI8-R)
ru_RU.ANSI1251	English	Russia	ansi-1251	Russian (Russia, ANSI 1251)
ru_RU.ISO8859-5	English	Russia	ISO8859-5	Russia (Russia)
sh_BA.ISO8859-2@bosnia	English	Bosnia	ISO8859-2	Bosnian (Bosnia)
sl_SI.ISO8859-2	English	Slovenia	ISO8859-2	Slovenian (Slovenia)
sq_AL.ISO8859-2	English	Albania	ISO8859-2	Albanian (Albania)
sr_YU.ISO8859-5	English	Serbia	ISO8859-5	Serbian (Serbia)
tr_TR.ISO8859-9	English	Turkey	ISO8859-9	Turkish (Turkey)

TABLE 3–7 Middle East

Locale	User Interface	Territory	Codeset	Language Support
he_IL.ISO8859-6	English	Israel	ISO8859-6	Hebrew (Israel)

TABLE 3–8 North Africa

Locale	User Interface	Territory	Codeset	Language Support
ar_EY.ISO8859-1	English	Egypt	ISO8859-6	Arabic (Egypt)

TABLE 3–9 North America

Locale	User Interface	Territory	Codeset	Language Support
en_CA.ISO8859-1	English	Canada	ISO8859-1	English (Canada)
en_US.ISO8859-1	English	USA	ISO8859-1	English (U.S.A.)
en_US.ISO8859-15	English	USA	ISO8859-15	English (U.S.A., ISO8859-15 - Euro)
en_US.UTF-8	English	USA	UTF-8	English (U.S.A., Unicode 3.0)
fr_CA.ISO8859-1	French	Canada	ISO8859-1	French (Canada)

TABLE 3–10 North Europe

Locale	User Interface	Territory	Codeset	Language Support
da_DK.ISO8859–1	English	Denmark	ISO8859–1	Danish (Denmark)
da_DK.ISO8859–15	English	Denmark	ISO8859–15	Danish (Denmark, ISO8859–15 Euro)
fi_FI.ISO8859–1	English	Finland	ISO8859–1	Finnish (Finland)
fi_FI.ISO8859–15	English	Finland	ISO8859–15	Finnish (Finland ISO8859–15 Euro)
is_IS.ISO8859–1	English	Iceland	ISO8859–1	Icelandic (Iceland)
no_NO.ISO8859–1@bokmal	English	Norway	ISO8859–1	Norwegian (Norway — Bokmal)
no_NO.ISO8859–1@nynorsk	English	Norway	ISO8859–1	Norwegian (Norway — Nynorsk)
sv_SE.ISO8859–1	Swedish	Sweden	ISO8859–1	Swedish (Sweden)
sv_SE.ISO8859–15	Swedish	Sweden	ISO8859–15	Swedish (Sweden, ISO8859–15 Euro)
sv_SE..UTF-8	Swedish	Sweden	UTF-8	Swedish (Sweden, Unicode 3.0)

TABLE 3–11 South America

Locale	User Interface	Territory	Codeset	Language Support
es_AR.ISO8859-1	Spanish	Argentina	ISO8859-1	Spanish (Argentina)
es_BO.ISO8859-1	Spanish	Bolivia	ISO8859-1	Spanish (Bolivia)
es_CL.ISO8859-1	Spanish	Chilie	ISO8859-1	Spanish (Chile)
es_CO.ISO8859-1	Spanish	Colombia	ISO8859-1	Spanish (Colombia)
es_EC.ISO8859-1	Spanish	Ecuador	ISO8859-1	Spanish (Ecuador)
es_PE.ISO8859-1	Spanish	Peru	ISO8859-1	Spanish (Peru)
es_PY.ISO8859-1	Spanish	Paraguay	ISO8859-1	Spanish (Paraguay)
es_UY.ISO8859-1	Spanish	Uruguay	ISO8859-1	Spanish (Uruguay)
es_VE.ISO8859-1	Spanish	Venezuela	ISO8859-1	Spanish (Venezuela)
pt_BR.ISO8859-1	English	Brazil	ISO8859-1	Portuguese (Brazil)

TABLE 3–12 South Europe

Locale	User Interface	Territory	Codeset	Language Support
el_GR.ISO8859-7	English	Greece	ISO8859-7	Greek (Greece)
es_ES.ISO8859-1	Spanish	Spain	ISO8859-1	Spanish (Spain)
es_ES.ISO8859-15	Spanish	Spain	ISO8859-15	Spanish (Spain, ISO8859-15 - Euro)
es_ES.UTF-8	Spanish	Spain	UTF-8	Spanish (Spain, Unicode 3.0)
it_IT.ISO8859-1	Italian	Italy	ISO8859-1	Italian (Italy)
it_IT.ISO8859-15	Italian	Italy	ISO8859-15	Italian (Italy, ISO8859-15 - Euro)
it_IT.UTF-8	Italian	Italy	UTF-8	Italian (Italy, Unicode 3.0)
pt_PT.ISO8859-1	English	Portugal	ISO8859-1	Portuguese (Portugal)
pt_PT.ISO8859-15	English	Portugal	ISO8859-15	Portuguese Portugal, ISO8859-15 - Euro)

TABLE 3–13 Western Europe

Locale	User Interface	Territory	Codeset	Language Support
en_GB.ISO8859-1	English	Great Britain	ISO8859-1	English (Great Britain)
en_GB.ISO8859-15	English	Great Britain	ISO8859-15	English (Great Britain, ISO8859-15 - Euro)
en_IE.ISO8859-1	English	Ireland	ISO8859-1	English (Ireland)
en_IE.ISO8859-15	English	Ireland	ISO8859-15	English (Ireland, ISO8859-15 - Euro)
fr_BE.ISO8859-1	French	Belgium-Walloon	ISO8859-1	French (Belgium-Walloon)
fr_BE.ISO8859-15	French	Belgium-Walloon	ISO8859-15	French (Belgium-Walloon, ISO8859-15 - Euro)
fr_FR.ISO8859-1	French	France	ISO8859-1	French (France)
fr_FR.ISO8859-15	French	France	ISO8859-15	French (France, ISO8859-15 - Euro)
fr_FR.UTF-8	French	France	UTF-8	French (France, Unicode 3.0)
nl_BE.ISO8859-1	English	Belgium-Flemish	ISO8859-1	Dutch (Belgium-Flemish)
nl_BE.ISO8859-15	English	Belgium-Flemish	ISO8859-15	Dutch (Belgium-Flemish, ISO8859-15 - Euro)
nl_NL.ISO8859-1	English	Netherlands	ISO8859-1	Dutch (Netherlands)
nl_NL.ISO8859-15	English	Netherlands	ISO8859-15	Dutch (Netherlands, ISO8859-15 - Euro)

Locale naming conventions are as follows:

language[_territory][.codeset]

where language is from ISO639 and territory is from ISO3166.

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

A single locale can have more than one locale name. For example, `ja_JP.eucJP` is the same as `ja`. Also, `fr_FR.ISO8859–1` is the same as `fr`.

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

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 3.0

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.

European Localization

Solaris 8 software supports the euro currency. Local currency symbols are still available for backward compatibility.

TABLE 3–14 User Locales To Support the Euro Currency

Region	Locale Name	ISO Codeset
Austria	`de_AT.ISO8859-15`	8859-15
Belgium (French)	`fr_BE.ISO8859-15`	8859-15
Belgium (Dutch)	`nl_BE.ISO8859-15`	8859-15
Denmark	`da_DK.ISO8859-15`	8859-15
Finland	`fi_FI.ISO8859-15`	8859-15
France	`fr_FR.ISO8859-15`	8859-15
Germany	`de_DE.ISO8859-15`	8859-15
Ireland	`en_IE.ISO8859-15`	8859-15
Italy	`it_IT.ISO8859-15`	8859-15
Netherlands	`nl_NL.ISO8859-15`	8859-15
Portugal	`pt_PT.ISO8859-15`	8859-15

TABLE 3–14 User Locales To Support the Euro Currency *(continued)*

Region	Locale Name	ISO Codeset
Spain	`es_ES.ISO8859-15`	8859-15
Sweden	`sv_SE.ISO8859-15`	8859-15
Great Britain	`en_GB.ISO8859-15`	8859-15
Europe	`en_EU`	8859-15
U.S.A.	`en_US`	8859-15

Multiple Key Compose Sequences for Locales

The Solaris 8 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 8859-13 (Latin7) 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 = v (for example, Compose + E + v = 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 Solaris 8 Product

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

- Czech
- Hungary
- Poland
- Latvia
- Lithuania
- Russia

- Greece
- Turkey

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 is a layout table for a type 4 keyboard (1=switch up, 0=switch down).

TABLE 3–15 Layouts for Type 4 Keyboards

Dip Switch	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./GB 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 and then re-booted.

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

Changing Between Keyboards on IA

On IA, 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 IA (apart from standard UNIX tools such as `xmodmap`, `pcmapkeys`) bundled into Solaris 8 for switching keyboards.

Codesets for IA

The default codeset on the Solaris system for IA 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
```

Nonstandard U.S. date, time, currency, numbers, units, and collation are not supported. Non-English message and text presentation is not supported, nor is 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 *loadfont* and *pcmapkeys* man pages.

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).

TABLE 3–16 `iconv` Support

Code	Symbol	Target Code	Symbol	Language Support
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

TABLE 3–16 `iconv` Support *(continued)*

Code	Symbol	Target Code	Symbol	Language Support
MS 1250	win2	Mazovia	maz	Mazovia
MS 1250	win2	DHN	dhn	Dom Handlowy Naduki
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
OKI8-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
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

TABLE 3–16 `iconv` Support *(continued)*

Code	Symbol	Target Code	Symbol	Language Support
`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

Font Formats

Location of Fonts on the System

Fonts to support European locales are available in various formats, such as bitmaps, Postscript™ Type-1, and TrueType. The actual availability varies per character set.

Fonts are located at:

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

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. Remove the optional font packages first, when you are removing font packages from the system.

You must follow this procedure to add or remove fonts. The class action scripts in the font packages depend on this to function properly. 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 can occur.

Summary of Asian Locales

The following table shows the Asian supported locales.

TABLE 3–17 Summary of Asian Locales

CD Set	Locale Name	Description	Supported Character Set
Korean	`ko`	Korean (EUC)	KSC 5601-1987
	`ko.UTF-8`	Korean (UTF-8)	KSC 5601–1992
Simplified Chinese	`zh`	Simplified Chinese (EUC)	GB 2312-1980
	`zh GBK`	Simplified Chinese (GBK)	GBK
	`zh.UTF-8`	Simplified Chinese (UTF-8)	Unicode 3.0
Traditional Chinese	`zh_TW`	Traditional Chinese (EUC)	CNS 11643 –1992
	`zh_TW.BIG5`	Traditional Chinese (BIG5)	BIG5
	`zh_TW.UTF-8`	Traditional Chinese (UTF-8)	Unicode 3.0
Japanese	`ja`	Japanese (EUC)	JIS x 0201-1976
	`ja_JP.PCK`	Japanese (PCK)[1]	JIS x 0208-1990
	`ja_JP.UTF-8`	Japanese (UTF-8)	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).

Simplified Chinese Localization

Simplified Chinese in the Solaris 8 environment provides three locales: `zh`, `zh.UTF-8`, 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 both the `zh.GBK` and the `zh.UTF-8` locales:

- New QuanPin
- New ShuangPin
- Quanpy
- GBK Code
- Japanese
- Hanja
- Zhuyin
- Unicode

The following table shows the TrueType fonts for the `zh` locale.

TABLE 3–18 Solaris 8 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

The following table shows the Bitmap Fonts for the zh Locale.

TABLE 3–19 Solaris 8 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

The following table shows the TrueType fonts for the zh.GBK Locale.

TABLE 3–20 TrueType Fonts for the zh.GBK Locale

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

The following table shows the Bitmap Fonts for the zh.GBK Locale.

TABLE 3–21 Bitmap Fonts for the zh.GBK Locale

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

The following table shows the supported codeset conversions for Simplified Chinese.

TABLE 3–22 Codeset Conversions for Simplified Chinese

Code	Symbol	Target Code	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
HZ-GB-2312	HZ-GB-2312	GB2312–80	zh_CN.euc
HZ-GB-2312	HZ-GB-2312	GBK	zh_CN.gbk
HZ-GB-2312	HZ-GB-2312	UTF-8	UTF-8
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
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 Localization

Traditional Chinese in the Solaris 8 product provides three locales: zh_TW, zh_TW.UTF-8 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. The zh_TW.UTF-8 locale supports Unicode 3.0

Traditional Chinese is used mostly in Taiwan and Hong Kong, and supports the following input methods:

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

The following table shows Traditional Chinese Truetype Fonts for the `zh_TW` Locales.

TABLE 3–23 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

The following table shows the Traditional Chinese BitMap Fonts for the `zh_TW` Locales.

TABLE 3–24 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

The following table shows the Traditional Chinese TrueType Fonts for the `zh_TW.BIG5` Locales.

TABLE 3–25 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

The following table shows the Traditional Chinese BitMap Fonts for the `zh_TW.BIG5` Locales.

TABLE 3–26 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

The following table shows the supported codeset conversions for Traditional Chinese.

TABLE 3–27 Codeset Conversions for Traditional Chinese

Code	Symbol	Target Code	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
`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

TABLE 3–27 Codeset Conversions for Traditional Chinese *(continued)*

Code	Symbol	Target Code	Symbol
`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 Localization

This section describes Japanese locale-specific information.

Japanese Locales

Three Japanese locales, which support different character encoding, are available in the Solaris 8 environment. The `ja`, (or `ja_JP.eucJP`) locale is based on the Japanese EUC. The `ja_JP.PCK` locale is based on PC-Kanji code (known as Shift-JIS) and the `ja_JP.UTF-8` is based on `UTF-8`.

See *eucJP(5)* for a map between Japanese EUC and the character set. See *PCK(5)* for the map between PCK and the character set.

Japanese Character Set

Supported Japanese character sets are:

- JISX0201–1976
- JISX0208–1990
- JISX0212–1990

JISX0212–1990 is not supported in the `ja_JP.PCK` locale.

Vendor Defined Character (VDC) and User defined Character (UDC) are also supported. VDCs occupy unused (reserved) code points of JISX0208–1990 or

JISX0212–1990. UDCs occupy the same code points as VDCs except the code points are for VDCs.

Japanese Font

Three Japanese font formats are supported. They are: Bitmap, TrueType and Type1. The Japanese Type1 font includes only JIS X0212 for printing. Type1 font is also used by UDC.

Japanese Bitmap Fonts are shown below.

TABLE 3–28 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 TrueType Fonts are show below.

TABLE 3–29 Japanese TrueType Fonts

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

Japanese Input Systems

Four Japanese input systems, ATOK12, ATOK8, Wnn6, and cs00 are available in the Solaris 8 environment for all Japanese locales. It is possible to switch input systems from the workspace menu. The only Japanese input system available on the Base Solaris is cs00.

How to Input Japanese Strings by using cs00

When turning Kana-Kanji conversion mode ON, keyboard input is grabbed by Htt (X Input Method Server) and sent to the cs00 daemon through the XCI (xci(7)) interface. The cs00 deamon converts the received strings to Japanese strings by using dictionary and returns the result to the program which has a keyboard focus now. See *cs00(1M)* for more details.

CUI based dictionary maintenance utilities are available. See *udicm(1)* and *mdicm(1)* for details.

GUI based maintenance utilities, *sdtudicm(1)* or *udicmtool(1)*, are not available in the base Solaris product.

The basic Japanese input procedure is as follows:

1. Turning Japanese conversion mode on/off: Control + Space
2. Enter Kana character text: ex: Type “nihon”
3. Conversion to Kanji character text: Control + N
4. Commit the Kanji character text: Control + K

The following table shows cs00 operation list.

TABLE 3–30 cs00 Operation List

Function	Operation
Conversion mode on/off	Control + Space Control + @
Kana/Kanji conversion	next Control + N post Control + P lookup Control + W
Commit	Control + K
Move focus	forward Control + F back Control + B

TABLE 3–30 cs00 Operation List *(continued)*

Function	Operation
Focus scope	increase Control + I decrease Control + U
Delete (1 character)	Control + H Delete or backspace
Delete (all characters)	Control +] and Control + U
Full / half Katanka => Hiragana	Control +] and Control + O
Hiragana / half Katakana = > full Katakana	Control +] and Control + Y
Full Katakana / Hiragana => half Katakana	Control +] and Control + Z
Half Roma / Num = > full Roma / Num	Control +] and Control + T
Full Roma / Num = > half Roma / Num	Control +] and Control + R
Learning Mode on / off	Control +] and Control + L
Input Mode Switch: ■ Hiragana mode ■ Full Katakana mode ■ Full Roma / Num mode ■ Half Katakana mode ■ Half Roma / Num mode ■ Kuten code input mode ■ Bushu input mode	Control + O Control + Y Control + T Control + Z Control + R Control + Q Control + V

Terminal Setting for Japanese Terminals

Using Japanese locales on a character based terminal (TTY) requires that you use terminal settings to make line editing work correctly.

- If your terminal is a CDE Terminal emulator (dtterm), use `stty(1)` with argument –`defeucw`, in any Japanese locale (`ja`, `ja_JP.PCK`, or `ja_JP.UTF-8`). An example in locale `ja` is:

```
% setenv LANG ja
% stty defeucw
```

- If your terminal is not a CDE Terminal emulator, but the codeset of your terminal is the same as that of the current locale, use this setting, too.

- If your terminal's codeset doesn't match that of the current locale, use `setterm(1)` to enable code conversion. For example, if you are in locale `ja` but your terminal requires PCK (ShiftJIS code), specify:

```
% setenv LANG ja
% setterm -x PCK
```

See *setterm(1)* for details.

Japanese `iconv` Module

Several Japanese codeset conversions are supported with `iconv(1)` and `iconv(3)`. See the *iconv_ja(5)* man page for details.

The following table shows `iconv` Conversion Support.

TABLE 3–31 `iconv` Conversion Support

Source Code	Target Code
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

TABLE 3–31 `iconv` Conversion Support *(continued)*

Source Code	Target Code
ISO-2022-JP	PCK
ISO-2022-JP	SJIS
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 eucJP	ISO-2022-JP.RFC1468 ibmj-EBCDIK
ibmj-EBCDIK	eucJP
PCK	ibmj-EBCDIK
ibmj-EBCDIK	PCK

Japanese Specific Printer Support

The Japanese Solaris 8 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

User Defined Character Support

To handle UDC, sdtudctool is available. `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 in prior releases, such as `fontedit`, `type3creator`, and `fontmanager`.

Not Included on the Base Solaris Product

The following components are included in the multilingual Solaris product (on Languages CD), but not included in the base Solaris product.

- All translations such as `message`, `help`, `manpage` and `document`
- Japanese BCP support
- `ATOK12`, `ATOK8`, and `Wnn6` Japanese input systems
- GUI utilities of the cs00 Japanese input system
- Mincho and Bold typeface fonts
- Japanese-specific dumb printer support
- `Sdtudctool` for UDC
- Legacy Japanese libraries (for example, `libjapanese.a` or `libmle.a`)
- Some Japanese specific utilities (e.g. `kanji`, or `vled`)

Korean Localization

In December 1995, the Korean government announced a standard Korean codeset, KS C 5700, which is based on ISO 10646-1/Unicode 2.0.

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 can 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.

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 to 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

TABLE 3–32 Solaris 8 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
Myeongijo	R	CID/Type 1	Hanyang	Adobe-Korean
Pilki	R	CID/Type 1	Hanyang	Adobe-Korean
Roundgothic	R	CID/Type 1	Hanyang	Adobe-Korean

TABLE 3–32 Solaris 8 Korean CID/Type 1 Fonts for the ko Locale *(continued)*

TABLE 3–33 Solaris 8 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
Myeongijo	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 3–34 Solaris 8 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
Myeongijo	R	CID/Type 1	Hanyang	Adobe-Korean
Pilki	R	CID/Type 1	Hanyang	Adobe-Korean

TABLE 3–35 Solaris 8 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)

TABLE 3–35 Solaris 8 Korean Bitmap Fonts for the `ko.UTF-8` Locale *(continued)*

Full Family Name	Subfamily	Format	Encoding
Kodig	R/B	PCF (12,14,16,18,20,24)	KSC 5601-1992 (Johap)
Myeongijo	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 3–36 Solaris 8 Korean TrueType Fonts for the `ko/ko.UTF-8` Locales

Full Family Name	Subfamily	Format	Vendor	Encoding
Kodig/Gothic	R	True Type	Hanyang	Unicode
Myeongjo	R	True Type	Hanyang	Unicode
Haeso	R	True Type	Hanyang	Unicode
RoundGothic	R	True Type	Hanyang	Unicode

TABLE 3–37 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

TABLE 3–37 Korean ICONV *(continued)*

Code	Symbol	Target Code	Symbol
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
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

CHAPTER 4

Overview of en_US.UTF-8 Locale Support

Unicode Overview

The Unicode Standard is the universal character encoding standard used for representation of text for computer processing. It is fully compatible with the International Standard ISO/IEC 10646-1:1999, and contains all the same characters and encoding points as ISO/IEC 10646. The Unicode Standard provides additional information about the characters and their use. Any implementation that conforms to Unicode also conforms to ISO/IEC 10646.

Unicode provides a consistent way of encoding multilingual plain text and brings order to a chaotic state of affairs that has made it difficult to exchange text files internationally. Computer users who deal with multilingual text, business people, linguists, researchers, scientists, and others, find that the Unicode Standard greatly simplifies their work. Mathematicians and technicians, who regularly use mathematical symbols and other technical characters, also find the Unicode Standard valuable.

The design of Unicode is based on the simplicity and consistency of ASCII, but goes beyond ASCII's limited ability to encode only the Latin alphabet. The Unicode Standard provides the capacity to encode all of the characters used for the written languages of the world. It uses a 16-bit encoding that provides code points for more than 65,000 characters. To keep character coding simple and efficient, the Unicode Standard assigns each character a unique 16-bit value, and does not use complex modes or escape codes. While 65,000 characters are sufficient for encoding most of the many thousands of characters used in major languages of the world, the Unicode standard and ISO 10646 provide an extension mechanism called UTF-16 that allows for encoding as many as a million more characters, without use of escape codes. This is sufficient for all known character encoding requirements, including full coverage

of all historic scripts of the world.UTF-16 allows exactly 16 x 65536 additional code points and still uses the two byte entities to represent characters. However those 16 x 65536 characters require two two byte entities (for a total of four bytes) per each character. For more details on the UTF-16, refer to section C.3 of "The Unicode Standard, Version 2.0" from Unicode Consortium, or Annex C of ISO/IEC 10646–1:1999, Information Technology—Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.

Unicode Locale: `en_US.UTF-8` Support Overview

The `en_US.UTF-8` locale is a significant Unicode locale in the Solaris 8 product. It supports and provides multiscript processing capability by using `UTF-8` as its codeset. It can input and output text in multiple scripts. This was the first locale with this capability in the Solaris operating environment.

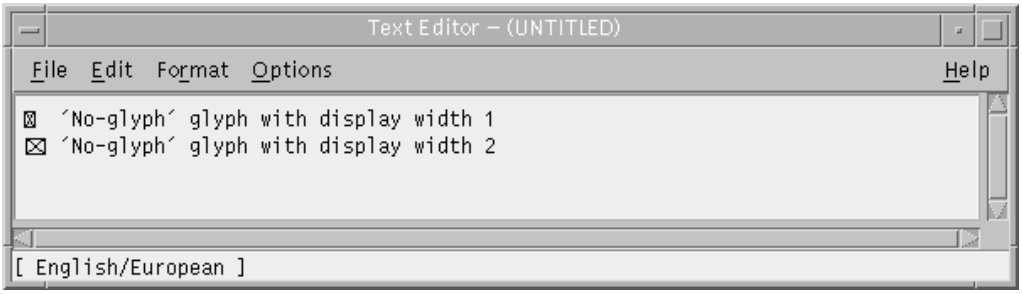
`UTF-8` is a file system safe Universal Character Set Transformation Format of Unicode / ISO/IEC 10646-1 formulated by X/Open-Uniforum Joint Internationalization Working Group (XoJIG) in 1992 and approved by ISO and IEC, as Amendment 2 to ISO/IEC 10646-1:1993 in 1996. This standard has been adopted by 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 3.0 and ISO/IEC 10646-1. In the Solaris 8 environment, language script support is not limited to pan-European locales, but also includes Asian scripts such as Korean, Traditional Chinese, Simplified Chinese, and Japanese. Due to limited font resources, Solaris 8 software includes only character glyphs from the following character sets:

- 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, including many more presentation form character glyphs)
- ISO 8859–7 (Greek)
- ISO 8859–8 (Hebrew)
- ISO 8859-9 (Turkish)
- TIS 620.2533 (Thai, including many more presentation form character glyphs)

- ISO 8859–15 (most Western European languages with euro sign)
- GB 2312–1980 (Simplified Chinese)
- Big5 (Traditional Chinese)
- JIS X0201–1976, JIS X0208–1983 (Japanese)
- KS C 5601–1992 Annex 3 (Korean)

If a user displays characters for which the en_US.UTF-8 locale does not have corresponding glyphs, the locale displays 'no-glyph' glyph instead, as in the following example:



Starting with the Solaris 8 environment, the locale is available for all clusters except the Core cluster.

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

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.

Desktop 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 language environments write their text from right-to-left, top-to-bottom, and so forth. Within the same application, you can use several fonts that apply 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
- 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.

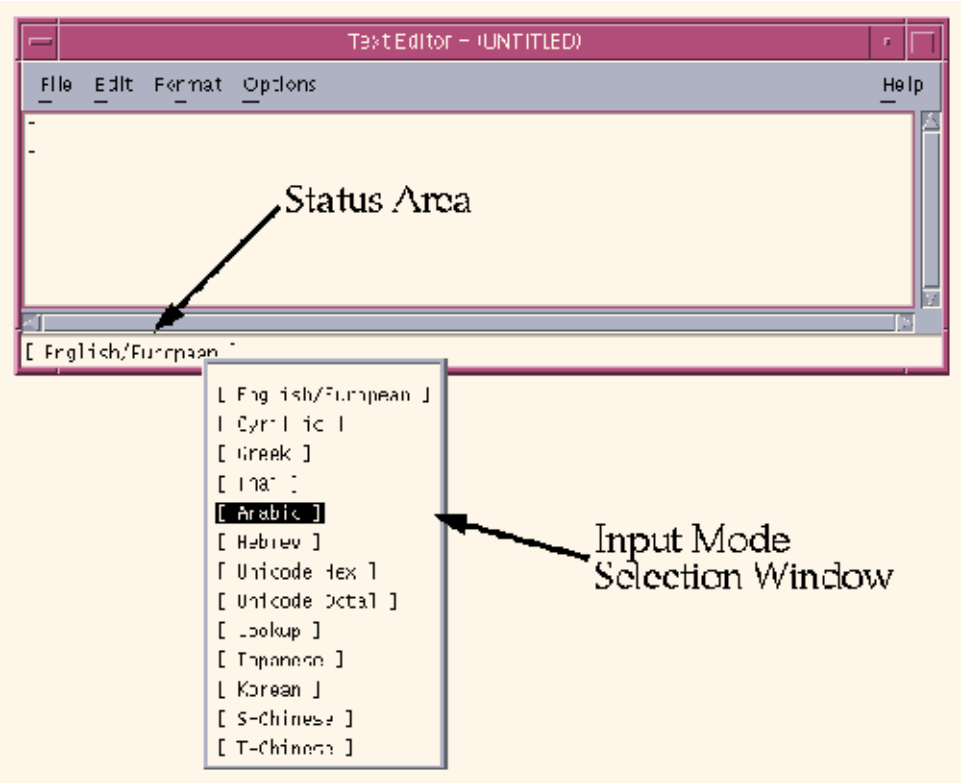
In the Solaris 8 environment, there are native Asian input methods for Simplified / Traditional Chinese, Japanese, and Korean in addition to the current multi-script input methods for Unicode locales. This section includes descriptions of selected input methods, how to use them, and how to switch between them.

Script Selection and Input Modes

The `en_US.UTF-8` locale supports multiple scripts. The en_US.UTF-8 locale has a total of twelve input modes:

- English / European
- Cyrillic
- Greek
- Arabic
- Hebrew
- Thai
- Unicode Hexadecimal and Octal code input methods
- Table lookup input method
- Japanese
- Korean
- Simplified Chinese
- Traditional Chinese

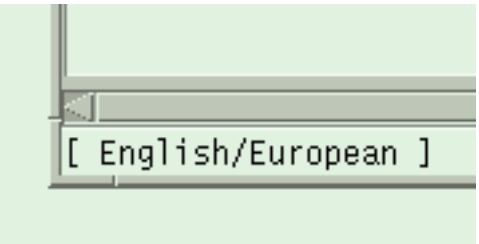
To switch into a certain input mode, you can either type in an input mode switch compose key sequence for each input mode, or press the left-most mouse button at the status area of your application to open an input mode selection window and select from the listed input modes as follows:



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, õ, 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.



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, the <Shift> key and the <t> keys 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
- <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 show the most commonly used Compose Sequences in Latin-1, Latin-2, Latin-4, Latin-5, and Latin-9 script input for the Solaris operating environment.

To start these sequences, press the <Compose> key and release it.

The following table lists the Common Latin-1 Compose Sequences.

TABLE 4–1 Common Latin-1 Compose Sequences

Press and Release	Press and Release	Result
[Spacebar]	[Spacebar]	No-break 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
'	"	acute accent ´
,	,	cedilla Ç ¸
"	"	diaeresis ¨

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

Press and Release	Press and Release	Result
-	^	macron ¯
o	o	degree °
x	x	multiplication sign x
+	-	plus-minus ±
-	-	soft hyphen –
-	:	division sign ÷
-	a	ordinal (feminine) ª
-	o	ordinal (masculine) º
-	,	not sign ¬
.	.	middle dot ·
1	2	vulgar fraction $\frac{1}{2}$
1	4	vulgar fraction $\frac{1}{4}$
3	4	vulgar fraction $\frac{3}{4}$
<	<	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 Å
A	"	A diaeresis Ä
A	^	A circumflex Â
A	~	A tilde Ã
A	E	AE diphthong Æ
C	,	C cedilla Ç
C	o	copyright sign ©

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

Press and Release	Press and Release	Result
D	-	Capital eth ð
E	‘	E grave È
E	’	E acute É
E	"	E diaeresis Ë
E	^	E circumflex Ê
I	‘	I grave Ì
I	’	I acute Í
I	"	I diaeresis Ï
I	^	I circumflex Î
L	-	pound sign £
N	~	N tilde Ñ
O	‘	O grave Ò
O	’	O acute Ó
O	/	O slash Ø
O	"	O diaeresis Ö
O	^	O circumflex Ô
O	~	O tilde Õ
R	O	registered mark ®
T	H	Thorn þ
U	‘	U grave Ù
U	’	U acute Ú
U	"	U diaeresis Ü
U	^	U circumflex Û
Y	’	Y acute ý

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

Press and Release	Press and Release	Result
Y	-	yen sign ¥
a	‘	a grave à
a	’	a acute á
a	*	a ring above å
a	"	a diaeresis ä
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 ð
e	‘	e grave è
e	’	e acute é
e	"	e diaeresis ë
e	^	e circumflex ê
i	‘	i grave ì
i	’	i acute í
i	"	i diaeresis ï
i	^	i circumflex î
n	~	n tilde ñ
o	‘	o grave ò
o	’	o acute ó
o	/	o slash ø
o	"	o diaeresis ö
o	^	o circumflex ô

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

Press and Release	Press and Release	Result
o	~	o tilde õ
s	s	German double s ß
t	h	thorn þ
u	‘	u grave ù
u	’	u acute ú
u	"	u diaeresis ü
u	^	u circumflex û
y	’	y acute y
y	"	y diaeresis ÿ
|	|	broken bar ¦

The following table lists the Common Latin-2 and Latin-4 Compose Sequences.

TABLE 4–2 Common Latin-2 Compose Sequences

Press and Release	Press and Release	Result
a	’ ’	ogonek
u	’ ’	breve
v	’ ’	caron
"	’ ’	double acute
A	a	A ogonek
A	u	A breve
C	’	C acute
C	v	C caron
D	v	D caron
-	D	D stroke
E	v	E caron

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

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

TABLE 4–2 Common Latin-2 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
r	,	r cedilla
t	|	t stroke
u	~	u tilde
u	a	u ogonek

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

Press and Release	Press and Release	Result
u	_	u macron
n	n	eng

The following table lists the Common Latin-5 Compose Sequences.

TABLE 4–3 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.

The following table lists the Common Latin-9 Compose Sequences.

TABLE 4–4 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, either press <Compose> <c> <c> at your keyboard, or press the left-most mouse button at the status area of your application and select “[Cyrillic]” from the Input Mode Selection Window.

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

After you switch to Cyrillic input mode, you cannot enter English or European text. To switch back to the English/European input mode, type <Control> + <Space> from your keyboard, or select “[English/European]” input mode from the Input Mode Selection Window by using your mouse. The Russian keyboard layout appears in the following figure.

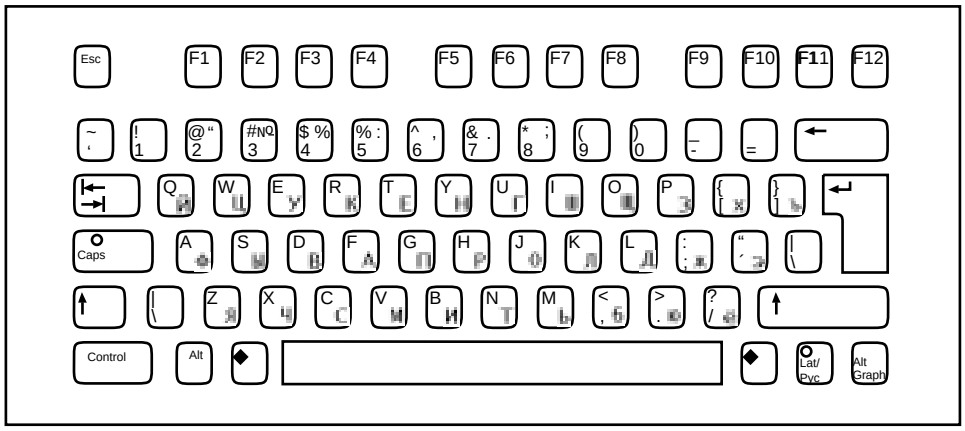


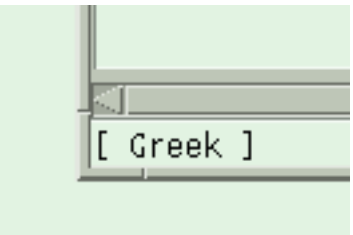
Figure 4–1 Tabbing Behavior

You can also switch into other input modes by typing the corresponding input mode switch key sequence.

Greek Input Mode

To switch to Greek input mode, either press Compose <g> <g> at your keyboard, or press the left-most mouse button at the status area of your application and select “[Greek]”, from the Input Mode Selection Window.

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



After you switch to Greek input mode, you cannot enter English or European text. To switch back to the English/European input mode, type <Control> + <Space> from your keyboard, or select “[English/European]” input mode from the Input Mode Selection Window by using your mouse. The Greek keyboard layouts appear in the following two figures.

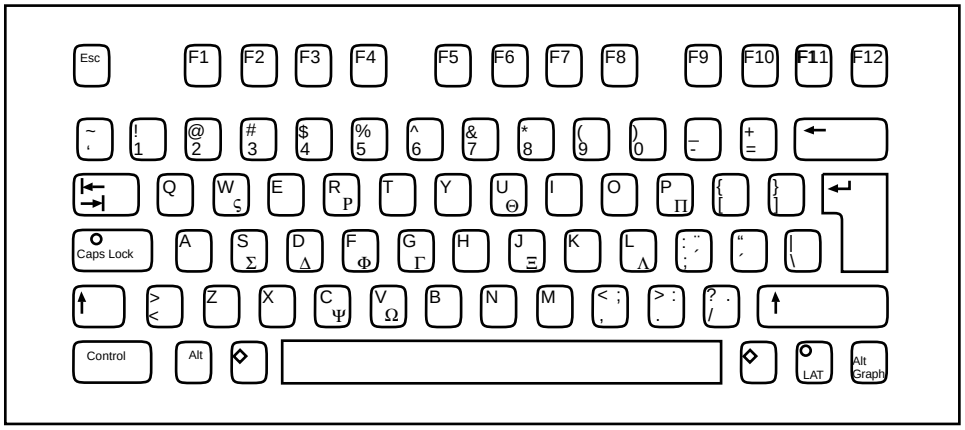


Figure 4–2 Tabbing Behavior

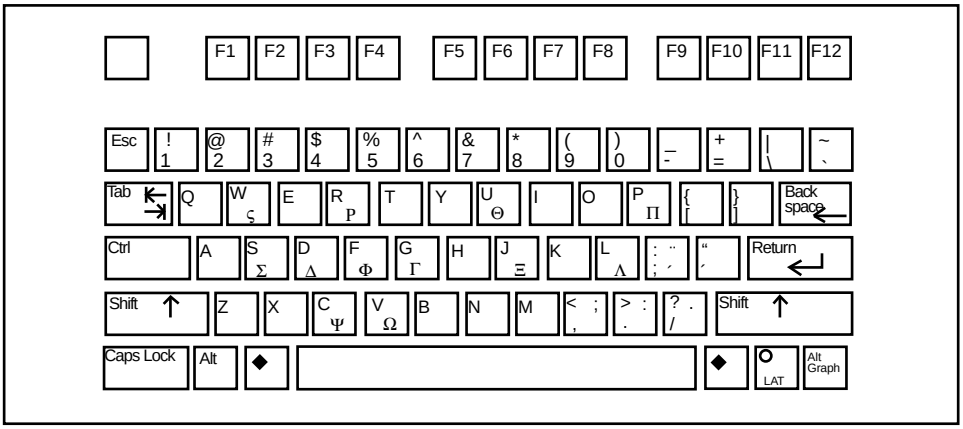


Figure 4–3 Tabbing Behavior

Arabic Input Mode

To switch to Arabic input mode, type <Compose> <a> <r> from your current input mode. The input mode is displayed at the left bottom corner of your GUI application. After 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.

You can also switch into other input modes by either typing the corresponding input mode switch key sequence from your keyboard, or selecting an input mode from the Input Mode Selection Window by using y our mouse.

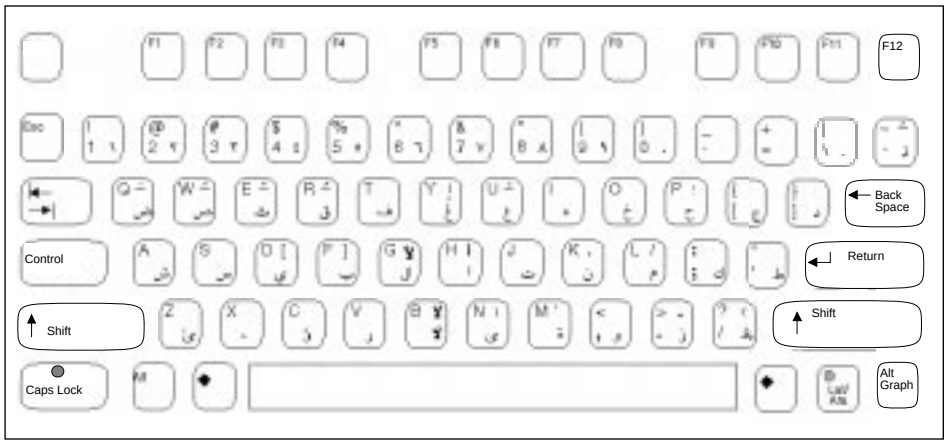


Figure 4–4 Tabbing Behavior

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 bottom left corner of your GUI application. You can also switch into the Hebrew input mode by pressing the left-most mouse button at the status area of your application and then selecting “[Hebrew]” from the Input Mode Selection Window.

After you have switched into the Hebrew input mode, you have to switch back to the English / European input mode to enter English / European characters. To switch your input mode, you can either type the corresponding input mode switch key sequence of your next input mode from your keyboard, or select an input mode from the Input Mode Selection Window by using your mouse. The Hebrew keyboard layout is shown in the following figure:

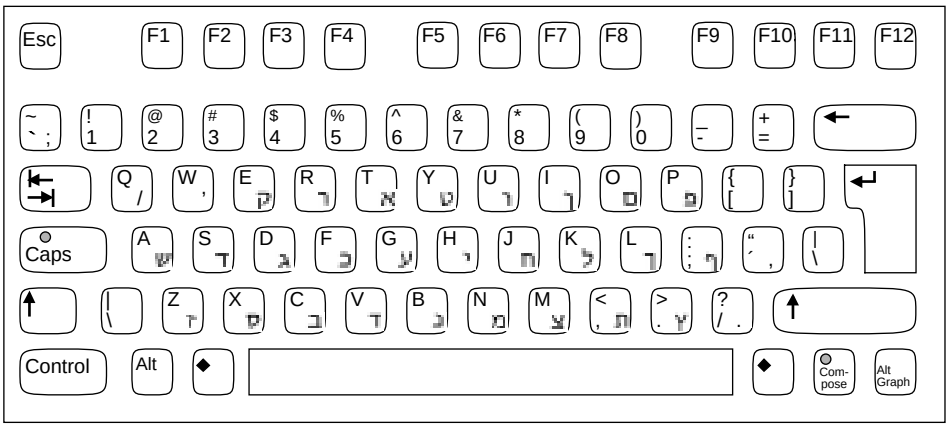


Figure 4–5 Tabbing Behavior

Thai Input Mode

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

After you have switched into the Thai input mode, you have to switch back to English/European input mode to enter English/European characters. To switch your input mode, either type the corresponding input mode switch key sequence of your next input mode from your keyboard, or select an input mode from the Input Mode Selection Window by using your mouse. The Thai keyboard layout is shown in the following figure:

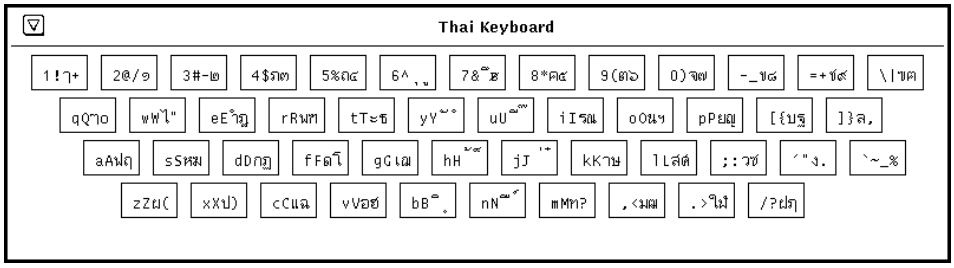


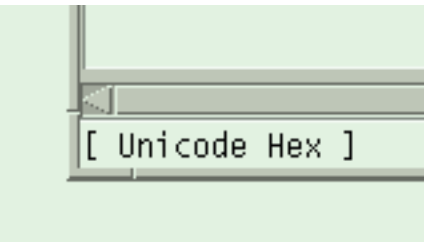
Figure 4–6 Tabbing Behavior

Unicode Hexadecimal and Octal Code Input Method Input Modes

To switch into the Unicode hexadecimal code input method input mode, type <Compose> <u> <h> from your current input mode. You can also select “[Unicode Hex]” from the Input Mode Selection Window by using your mouse. The input mode is displayed at the left bottom corner of your application.

If you prefer the octal number system, you can also switch into the Unicode octal code input method input mode by typing <Compose> <u> <o> from your current input mode or by selecting “[Unicode Octal]” from the Input Mode Selection Window

To use these input mode, you need to know about either the hexadecimal or the octal code point values of the characters. Refer to *The Unicode Standard, Version 3.0* for the mapping between code point values and characters. To input a character, type four hexadecimal digits if you are in the Unicode hexadecimal code input method input mode, for instance, 00a1 for Inverted Exclamation Mark, 03b2 for Greek Small Letter Beta, ac00 for a Korean Hangul 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 prefer the octal number system instead of hexadecimal numbers, you can input octal digits, 0 to 7. If you mistype a digit or two, you can delete the digits by using the <Delete> key or the <Backspace> key.



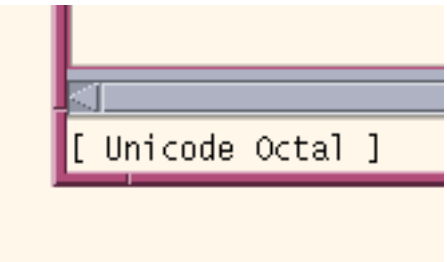
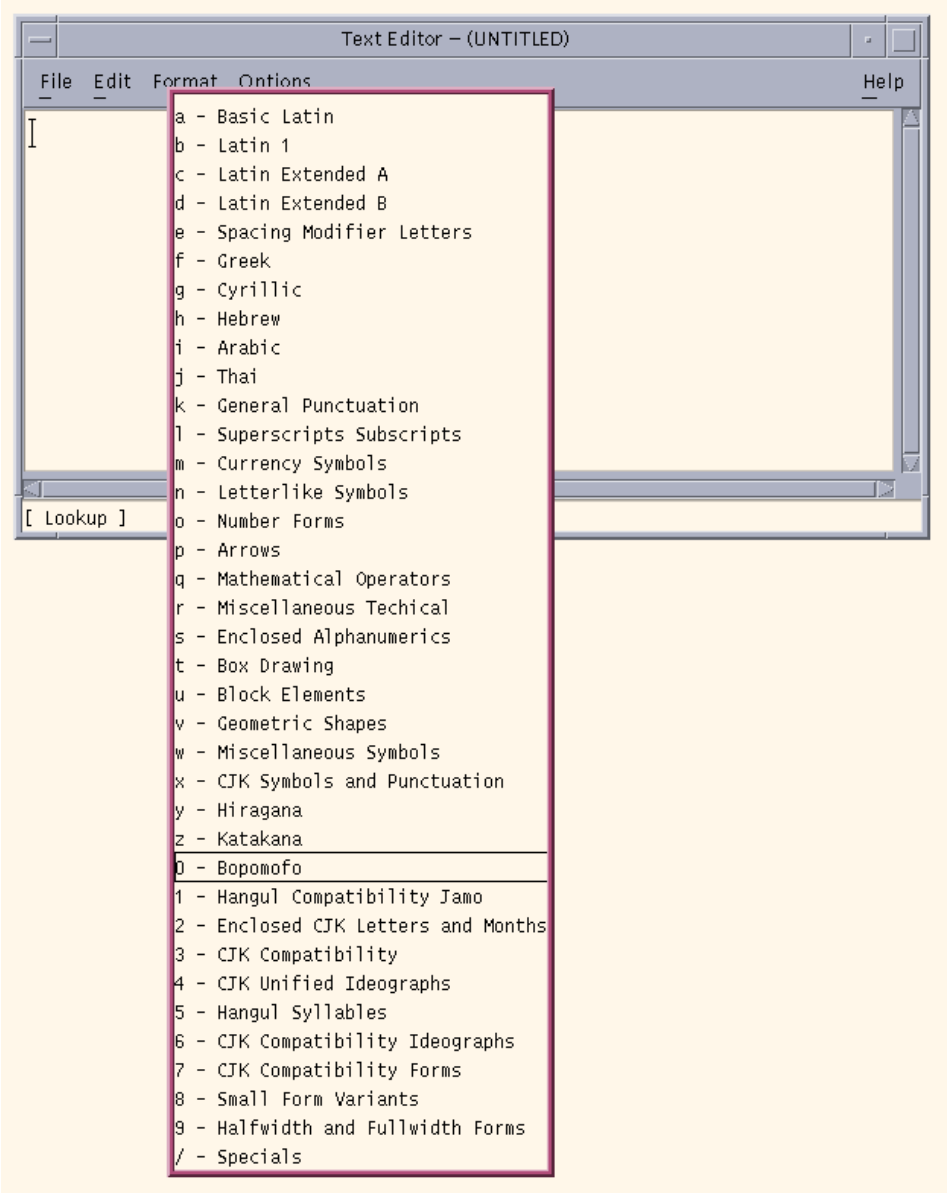


Table Lookup Input Method Input Mode

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 bottom left corner of your GUI application.



After you turn on the input mode, there is a lookup group window showing multiple groups of characters. You can choose any one of the groups to enter characters from the group. Once you select a group, there will be the second lookup window showing multiple candidates of available Unicode characters belonging to the group of your choice. You can choose any one of the candidates by moving your pointer and clicking

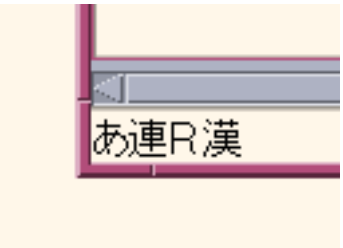
the left button on your mouse. You can also select any one of the candidates by choosing a left-hand-side letter associated with each of the candidates.

You can also see the next set of candidates by typing <Control> and <n> keys together. Similarly, to see the previous set of candidates, type the <Control> and <p> keys together. The <n> stands for 'next' and the <p> stands for 'previous'.

After you are finished using the current input mode, you can switch into another input mode by typing a corresponding input mode switch key sequence.

Japanese Input Mode

To switch into the Japanese input mode, type either <Compose> <j> <a> from your keyboard or select "[Japanese]" from the Input Mode Selection Window by using your mouse. The input mode is displayed at the left bottom corner of your application. The following figure shows a Japanese input method mode of ATOK12:

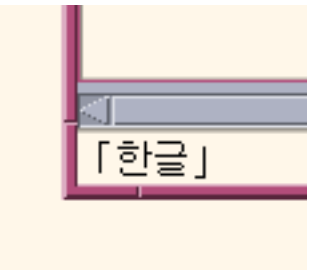


To use the native Japanese input system, you need to install one or more of Japanese locales on your system. Once you install the Japanese locales, you will be able to use any one of native Japanese input systems like ATOK12, ATOK8, Wnn6, or cs00.

For more details on how to use the Japanese Input System, refer to "*ATOK12 User's Guide*", "*ATOK8 User's Guide*", "*Wnn6 User's Guide*", and, "*cs00 User's Guide*."

Korean Input Mode

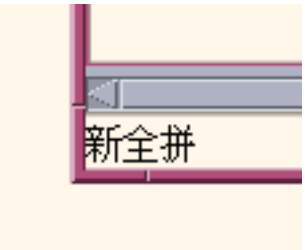
To switch into the Korean input mode, type either <Compose> <k> <o>from your keyboard, or select "[Korean]" from the Input Mode Selection Window by using your mouse. The input mode is displayed at the left bottom corner of your application. The following figure shows Phonetic Hangul input method which is one of many native Korean input methods available.



To have the native Korean input system, you need to install one or more Korean locale on your system. Once you install the Korean locale, you will be able to use the native Korean input system. For more details on how to use the Korean Input System, refer to *"Korean Solaris User's Guide"*.

Simplified Chinese Input Mode

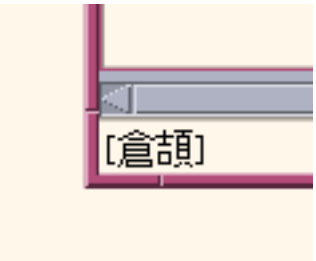
To switch input Simplified Chinese input mode, type either <Compose> <s> <c> from your keyboard, or select "[S-Chinese]" from the Input Mode Selection Window by using your mouse. The input mode is displayed at the left bottom corner of your application. The following figure shows New Pin Yin input method which is one of many native Simplified Chinese input methods available.



To use the native Simplified Chinese input system, you need to install one or more Simplified Chinese locales on your system. Once you install the Simplified Chinese locales, you will be able to use the native Simplified Chinese input system. For more details on how to use Simplified Chinese Input System, refer to *"Simplified Chinese Solaris User's Guide."*

Traditional Chinese Input Mode

To switch input Traditional Chinese input mode, type either <Compose> <t> <c> from your keyboard or select "[T-Chinese]" from the Input Mode Selection Window by using your mouse. The input mode is displayed at the left bottom corner of your application. The following figure shows the TsangChieh input method which is one of many native Traditional Chinese input methods available.



To have the native Traditional Chinese input system, you need to install one or more of Traditional Chinese locales at your system. Once you install the Traditional Chinese locales, you will be able to use the native Traditional Chinese input system. For more details on how to use the Traditional Chinese Input System, refer to *"Traditional Chinese Solaris User's Guide"*.

Input Mode Switch Key Sequence Summary

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 4–5 Input Mode Switch Key Sequences

Input Mode	Key Sequences
English/European	<Control> + <Space>
Cyrillic	<Compose> <c> <c>
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>
Unicode octal code input method	<Compose> <u> <o>
Japanese	<Compose> <j> <a>
Korean	<Compose> <k> <o>
Simplified Chinese	<Compose> <s> <c>
Traditional Chinese	<Compose> <t> <c>

System Environment

Locale Environment Variable

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

▼ How 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
```

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 have at setting the locale. See the setlocale(3C) man page for more details about the hierarchy of environment variables.

To check current locale settings in various categories, use the locale(1) utility.

```
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

Depending on the terminal and terminal emulator, such as dtterm(1)that you are using, you may need to push certain codeset-specific STREAMS modules onto your Streams.

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

The following table shows STREAMS modules supported by the en_US.UTF-8 locale in the terminal environment:

TABLE 4–6 32–bit STREAMS Modules Supported by `en_US.UTF-8`

32–bit STREAMS Module	Description
`/usr/kernel/strmod/u8lat1`	Code conversion STREAMS module between `UTF-8` and ISO 8859–1 (Western European)
`/usr/kernel/strmod/u8lat2`	Code conversion STREAMS module between `UTF-8` and ISO 8859–2 (Eastern European)
`/usr/kernel/strmod/u8koi8`	Code conversion STREAMS module between`UTF-8` and `KOI8-R` (Cyrillic)

The following table lists the 64–bit STREAMS Modules Supported by `en_US.UTF-8`.

TABLE 4–7 64–bit STREAMS Modules Supported by `en_US.UTF-8`

64-bit STREAMS module	Description
`/usr/kernel/strmod/sparcv9/u8lat1`	Code conversion STREAMS module between`UTF-8` and ISO 8859-1 (Western European)
`/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 64-bit Solaris or 32-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 u8lat1  modulename
system#
```

If the STREAMS module, such as u8lat1, is already installed, the output looks as follows:

```
system# modinfo | grep u8lat1
89 ff798000  4b13  18   1  u8lat1 (UTF-8 <--> ISO 8859-1 module)
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/u8lat1 modulename
```

This loads the 32–bit u8lat1 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/u8lat1
```

The STREAMS module is loaded 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 u8lat1 module is being unloaded.

```
system# modinfo | grep u8lat1
89 ff798000  4b13  18   1  u8lat1 (UTF-8 <--> ISO 8859-1 module)
system# modunload -i 89
```

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

Unlike in previous releases of the Solaris operating environment, the dtterm(1) and any other terminals that support input and output of the UTF-8 codeset do not need to have any other additional STREAMS module in their Stream. ldterm(7M) module is now codeset independent and supports Unicode/UTF-8 as well.

For the proper terminal environment setup for the Unicode locales, use the `stty(1)` utility as follows:

```
system% stty defeucw
```

Since `/usr/ucb/stty` is not internationalized, use `/bin/stty` instead.

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 <->  ldterm <->  u8lat1 <-> TTY
```

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`.

Make sure you already have the STREAMS module loaded into the kernel.

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

```
system% cat > tmp/mystreams
ttcompat
ldterm
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
ldterm
u8lat1
ptem
pts
system%
```

To reset the original configuration, set the STREAMS configuration as follows:

```
system% cat > /tmp/orgstreams
ttcompat
ldterm
ptem
```

(continued)

```
^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  defeucw
```

Because /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 that .cshrc refers 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(3).

In the Solaris 8 environment, the utility geniconvtbl enables user-defined code conversions. The user-defined code conversions created with the geniconvtbl utility can be used with both iconv(1) and iconv(3). For more detail on this utility, refer to geniconvtbl(1) and geniconvtbl(4) man pages.

The available fromcode and tocode names that can be applied to iconv(1) and iconv_open(3)are shown in the following table. 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).

Also see Appendix A.

UCS-2, UCS-4, UTF-16 are all fixed-width Unicode/ ISO/IEC 10646 representation forms that recognizes Byte Order Mark (BOM) characters defined in the Unicode 3.0 and ISO/IEC10646-1:1999 standards. Other forms, like UCS-2BE, UCS-4BE, and UTF-16BE, are all fixed-width Unicode/ ISO/IEC 10646 representation forms that do not recognize the BOM character and also assume Big Endian byte ordering. Representation forms like UCS-2LE, UCS-4LE, UTF-16LE, on the other hand, assume Little Endian byte ordering. They also do not recognize the BOM character.

For associated scripts/languages of ISO 8859-* and KOI8-*, see http://czyborra.com/charsets/iso8859.html.

Printing

A new and enhanced mp(1) print filter is available in the Solaris 8 environment that can print various input file formats including flat text files written inUTF-8. It uses TrueType and Type 1 scalable fonts and X11 bitmap fonts available on the Solaris system.

The output from the utility is standard PostScript, and can be sent to any PostScript printer.

Starting with the next release of the Solaris environment, `xutops(1)` will be obsolete.

To use the utility, type the following:

```
system% mp filename | lp
```

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

```
system% cat filename | mp | 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 `mp` 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 MP
system# accept lp1
system# enable lp1
```

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

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

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

```
Input types: simple
Output types: MP
Command: /usr/bin/mp
```

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

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

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

For more detail on `mp(1)`, refer to the `mp`(1) man page.

DtMail

As a result of increased coverage in scripts, Solaris 8 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)
- ISO-8859–13 (Latin-7 / Baltic)
- ISO-8859–14 (Latin-8 / Celtic)
- KOI8–U (Cyrillic / Ukranian)
- Shift_JIS (Japanese in Shift JIS)
- BIG5 (Traditional Chinese in BIG5)
- GB2312 (Simplified Chinese in EUC)
- TIS-620 (Thai)
- UTF-16 (UCS Transmission Format 16 of Unicode)
- UTF-16BE (UTF-16 Big-Endian of Unicode)
- UTF-16LE (UTF-16 Little-Endian of Unicode)

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 click again on the "Change Char Set" button by using your mouse. The next available character set name displays 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 character.

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 messages 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`. Any 8-bit characters of `UTF-8` are 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.

FontSet Used with *X* Applications

The `en_US.UTF-8` locale in the Solaris 8 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
- JIS X0201.1976
- JIS X0208.1983
- KS C 5601.1992 Annex 3
- ISO 8859–6 and Unicode based one
- ISO 8859–8
- TIS 620.2533 based one

Because the Solaris 8 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 8 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:

(continued)

```
fs = XCreateFontSet(display,
"-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-1,
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-2,
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-4,
 -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*-*-*-*-*-*-*-jisx0208.1983-0,
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-jisx0201.1976-0,
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-ksc5601.1992-3,
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-gb2312.1980-0,
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-tis620.2533-0,
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-unicode-fontspecific",
 &missing_ptr, &missing_count, &def_string);
```

Or, put more simply:

```
fs = XCreateFontSet(display,
   "-dt-interface system-medium-r-normal-*s*utf*",
    &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:\
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-1;\
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-2;\
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-4;\
 -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;\
```

(continued)

(Continuation)

```
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-big5-1;\
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-jisx0208.1983-0;\
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-jisx0201.1976-0;\
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-ksc5601.1992-3;\
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-gb2312.1980-0;\
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-tis620.2533-0;\
 -dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-unicode-fontspecific:
```

Or, put more simply:

```
*XmPushButton.fontList:\
  -dt-interface system-medium-r-normal-*s*utf*:
```

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.*

CHAPTER 5

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 explained. The book's Appendix E also explains 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.

CHAPTER 6

Desktop Environments

The Common Desktop Environment (CDE) is the standard GUI desktop interface for Solaris 8. Not only is it the user's main interface to the system, but 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 of CDE

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 needs to be localized only to match the target language and customs.

The application interface has been standardized to allow functionality in any locale, including East Asia. Solaris 8 complies with the Portable Operating Systems Interface for Computer Environments (POSIX and X/Open specifications, which are also referred to as XPG4.2).

Each layer within the desktop must 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's 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 updated version is supplied online with the CDE AnswerBooks. See "Related Books and Sites" on page 16 for more information.

Setting 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.

`Xt` programs 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 might 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 might 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 *Common Desktop Environment: Programmer's Overview* "Standard Font Names" in *Common Desktop Environment: Programmer's Overview* and also the CDE man pages *DtStdInterfaceFontNames(5)* and *DtStdAppFontNames(5)* for additional information.

Internationalization and CDE

Multiple environments can 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, China, and the Middle East.

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 that contains *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 can 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 you should develop a tool 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 explains the exchange of information between applications on different hosts. To transfer data, you must 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 might apply. The conversion depends on the specific protocol. If the protocol allows 8-bit data, the protocol must specify which side makes 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.

Because `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. `dtmail` also has the ability to use characters of different encodings within the same mail, for example, SJIS and EUC encodings for the Japanese (`ja`) locale.

OpenWindows

Solaris 8 does not have any changes in OpenWindows with regard to internationalization. Applications that were developed for previous versions of Solaris will run in Solaris 8 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*.

CHAPTER 7

Complex Text Layout

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. Because text rendering is handled through the rendition layer, other widget libraries can be easily extended to support CTL.

Overview of CTL Technology

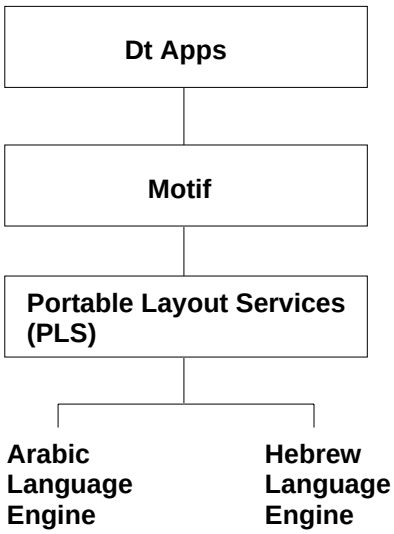
To leverage the new features, users must have the Portable Layout Services (PLS) library and the appropriate language engine. CTL uses PLS as the interface to the language engine, and uses the language engine to transform text before it is rendered. Applications that support CTL must include additional 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 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` resource[0] controls object layout. It interacts with the orientation value of the `LayoutObject` in the manner described below.

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`.

Layout Direction

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 and 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.

This document describes a set of portable functions for handling context-dependent and bidirectional text transformations as a logical extension to the existing POSIX locale model. It is intended for system and application programmers who want to provide support for complex-text languages.

XmStringDirection

XmStringDirection is the data type used to specify the direction in which the system displays characters of a string.

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 you need to do is to set the appropriate value for the XmNlayoutDirection resource. You do not need 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.

See also XmRendition and XmDirection.

XmRendition

CTL adds the following new pseudo resources to XmRendition:

TABLE 7–1 New Resources in XmRendition

Name	Class/Type	Access	Default Value
XmNfontType	XmCFontType/XmFontType	CSG	XmAS_IS
XmNlayoutAttrObject	XmClayoutAttrObject/String	CSG	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 is 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 is 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. Unpredictable behavior can result if the Orientation, Context, TypeOfText, TextShaping, or ShapeCharset are changed.

Additional Layout 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.Modifying these values dynamically, through the rendition object is not usually meaningful, and almost certain to result in unpredictable behavior.
- Layout behavior: Orientation, Context, ImplicitAlg, Swapping, Numerals

  Orientation and Context should not be modified dynamically; you can safely modify ImplicitAlg, Swapping, and Numerals.
- Editing behavior: CheckMode

XmText, XmTextField

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," while "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.

TABLE 7–2 New Resources in Xm CTL

Name	Class/Type	Access	Default Value
XmNrenditionTag	XmCRenditionTag/XmRString	CSG	XmFONTLIST_DEFAULT_TAG
XmNalignment	XmCAlignment/XmRAlignment	CSG	XmALIGNMENT_BEGINNING
XmNeditPolicy	XmCEditPolicy/XmREditPolicy	CSG	XmEDIT_LOGICAL

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.

Character Orientation 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()`

Character Orientation 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 might behave unexpectedly. Moreover, as the locale-specific modules fall outside of the scope of this specification, bi-directional editing behavior can differ from platform to platform for the same text, application, resource values, and `LayoutObject` configuration.

The visual mode actions result in a display of 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"[1] 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 that 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 can represent only a fragment of a logical character, for example.

`XmText` 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 the 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 can 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 can 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 the 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 can 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 can 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 can 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 can 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 can 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 can 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 can 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 can 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 can 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 can 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 `XmNmotionVerifyCallback` procedures with the reason value `XmCR_MOVING_INSERT_CURSOR`. If called with an extend argument, this can produce calls to the `XmNgainPrimaryCallback` procedures. See the callback description in the *Motif Programmer's Reference* for more information.

`forward-cell(extend)`

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 can 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 only 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 remain 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)
XmStringDirection direction
```

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 you need to do is set the appropriate value for the `XmNlayoutDirection` resource. You need not 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

The following table shows the UIL arguments:

TABLE 7–3 UIL

UIL Argument Name	Argument Type
`XmNlayoutAttrObject`	String
`XmNlayoutModifier`	String
`XmNrenditionTag`	String

TABLE 7–3 UIL *(continued)*

UIL Argument Name	Argument Type
`XmNalignment`	Integer
`XmNeditPolicy`	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 with the same resource.

In the case of an `XmText` widget, you must 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 you require top to bottom is alignment, 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.

The input string used in the illustration is:

The `XmNlayoutModifier` string `@ls orientation=` setting values for this illustration are shown below, in the left column.

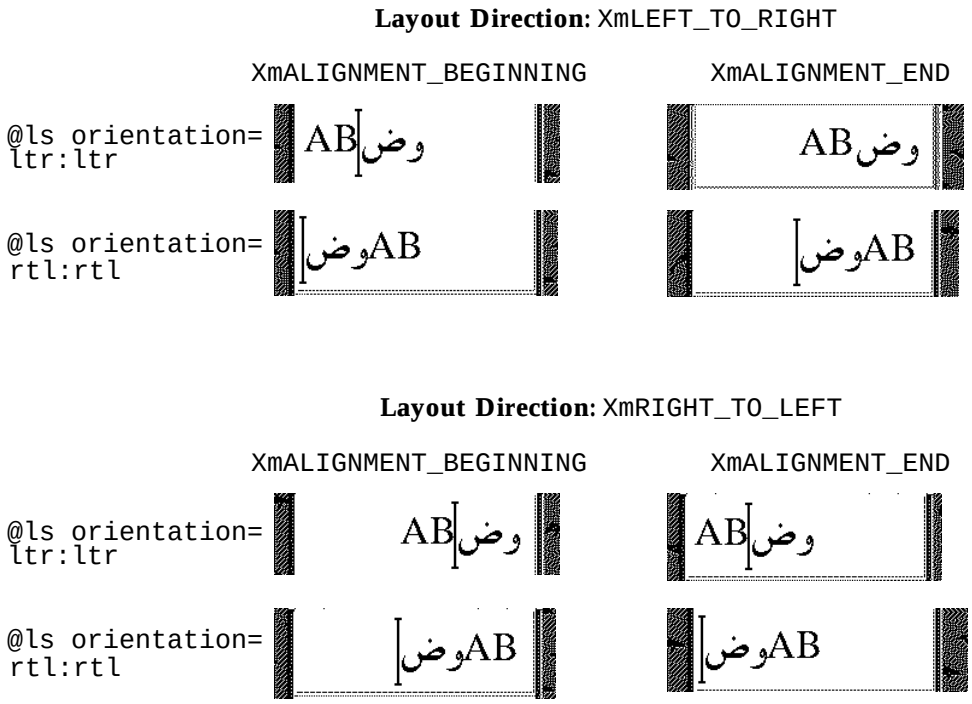


Figure 7–1 Tabbing Behavior

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-3.0."

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

3.0 codeset. Because 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\", 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, argcs
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`. Both the `XmNlayoutAttrObject` and `XmNlayoutModifier` pseudo resources have been left unspecified and therefore defaults to NULL. This means the `LayoutObject` associated with the `Rendition` belongs to the default locale, 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 through 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

`Rendition`s and render tables can be specified in resource files. For a 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 can 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

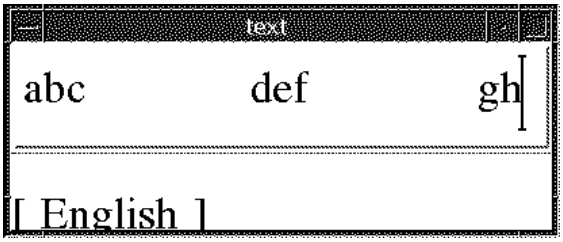
To control the placement of text, a compound string can contain tab characters. To interpret those characters on display, a widget refers to the rendition in effect for that compound string, where it finds a list of tab stops. However, the dynamic widgets (`TextField` and `XmText`) do not use the tab 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`).

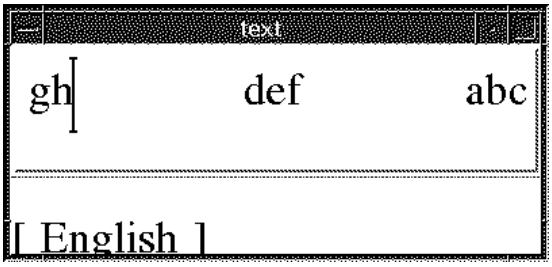
The text following a tab is always aligned at the tab stop, and the tab stop is calculated from the start of the widget, which in turn is influenced by `XmNlayoutDirection`.

The behavior of the tabs and their interaction with directionality of the text and the `XmNlayoutDirection` of the widget is illustrated in the following figure.

The input for this illustration is `abc\tdef\tgh`.



Layout Direction: `XmLEFT_TO_RIGHT`



Layout Direction: `XmRIGHT_TO_LEFT`

Figure 7–2 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 contiguous visually and will be 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 can 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 does not result in a continuous stream of characters.

Conversely, when the `XmNeditPolicy` is set to `XmEDIT_VISUAL`, the text selected can 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.

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.
Recompile the existing application. It will automatically link with the CTL-enabled Motif library `libXm.so.4`.
Add the following resources to your application resource file. Without these resources, the layout engine of the locale will not launch.
Add the following resources to your application resource file. Without these resources, the layout engine of the locale will not launch
Refer to the sample application attached to your documentation.

Use the font name that is available and appropriate to your locale in the `fontName` resource.

---

1. 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\
```

CHAPTER 8

Printing

Localization Printing Support Under the Solaris 8 Operating Environment

The Solaris environment provides support for PostScript printers. Custom print filters are available to convert localized text to PostScript. See `mp`(1) anasi `postprint`(1) man pages for further details. Fonts can also be downloaded onto a printer.

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

The Solaris 8 environment has a unified printing filter that replaces all the locale-specific filters described below. This section describes this filter and which scripts are supported in each locale Sun supports.

This filter uses font glyphs from printer-resident fonts and TrueType fonts in the Solaris operating environment; PCF bitmap fonts in the Solaris system depend on the configuration information defined for each locale. For more information on PCF (Portable Compiled Format), see man pages *bdftosnf(1)* and *bdftopcf(1)*.

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 the following table.

TABLE 8–1 `prolog.ps` Fonts

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

TABLE 8–1 `prolog.ps` Fonts *(continued)*

Table 8–1 is an example of the ISO8859–5 locale. The actual `prolog.ps` will vary depending on the locale.

Asian Multibyte Printing Support

The `xetops` and `xutops` utilities convert Asian text into a bitmapped graphics printed image. This enables you to print Asian characters on PostScript-based printers, even without having 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, zh.UTF-8 and zh_TW.UTF-8 locales)
```

Japanese Solaris 8 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. The following table shows the relation between these printers and user components. See *JFP User's Guide* for further details.

TABLE 8–2 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

TABLE 8–2 Japanese Printer Support *(continued)*

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) 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
```

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

Solaris Font Downloader

The Solaris Font Downloader is a vital part of internationalized printing.PostScript printers sold in different countries do not always have a set of locale-specific fonts installed on them. The usual solution for this problem was to have these locale-specific fonts included with each print job, which tended to lead to enormously large, slowly-processed, print jobs.

An alternative is to have all the frequently used fonts reside on the printer. They can be placed either in printer RAM, or on a hard disk if a printer has one connected to it. Most modern PostScript printers have the option of connecting a hard disk to them. The process of taking font files from the workstation and placing them on a printer is called “downloading.” Fonts downloaded to RAM are available until the printer is power-cycled. Fonts downloaded to a hard disk are available until they are removed..

The Solaris Font Downloader is a GUI application for managing fonts on PostScript printers. It supports a number of different popular printers running PostScript Level 2 or Level 3 software and connected to a network with TCP/IP protocol.

Specifically, it provides the following functionality:

- Download PostScript fonts to a printer
- Convert and download TrueType fonts to a printer
- Remove previously downloaded fonts from a printer
- Report general information, orproperties, about a printer, such as the amount of RAM and hard disk capacity, and a list of available fonts, for example.
- Print character samples
- Reformat hard disk on a printer

The Solaris Font Downloader works with a variety of different fonts available for a computer user. It can download the following PostScript fonts to a printer:

- Type 1
- Type 3
- Type 9 (CID Type 0)
- Type 10 (CID Type 1),
- Type 11 (CID Type 2),
- Type 42

It can also convert TrueType fonts to PostScript fonts such as Type 42 fonts or CID (Type 11) fonts "on the fly", while these fonts are being downloaded. A PostScript software that supports such fonts uses these converted TrueType fonts as if they were regular PostScript fonts.

There are a number of user-selectable choices for converting TrueType fonts to PostScript fonts. These are fully documented along with the rest of the Solaris Font Downloader features in the man page *fdl(1)*.

Reference Documents

- PostScript Language Reference Manual, 3rd ed. Adobe Systems Incorporated, ISBN 0-201-37922-8
- The Type 42 Font Format Specification. Adobe Systems Technical Note #5012, July 1998.
- TrueType 1.0 Font Files. Technical Specification Revision 1.66, November 1995 - Microsoft Corporation available from ftp.microsoft.com

APPENDIX **A**

iconv Code Conversions

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment

From Code (Symbol)	To Code (Symbol)
646 (ISO 646)	UTF-8
646 (ISO 646)	UCS-2
646 (ISO 646)	USC-2BE
646 (ISO 646)	UCS-2LE
646 (ISO 646)	USC-4
646 (ISO 646)	USC-4BE
646 (ISO 646)	USC-4LE
646 (ISO 646)	UTF-16
646 (ISO 646)	UTF-16BE
646 (ISO 646)	UTF-16LE
646 (ISO 646)	UTF-8
8859–11	UTF-8
8859-1 (ISO 8859-1)	UCS-2
8859-1 (ISO 8859-1)	UCS-2BE
8859-1 (ISO 8859-1)	UCS-2LE
8859-1 (ISO 8859-1)	UCS-4
8859-1 (ISO 8859-1)	UCS4-BE
8859-1 (ISO 8859-1)	UCS-4LE

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
8859-1 (ISO 8859-1)	UTF-16
8859-1 (ISO 8859-1)	UTF-16BE
8859-1 (ISO 8859-1)	UTF-16LE
8859-1 (ISO 8859-1)	UTF-8
8859-10 (ISO 8859-10)	UCS-2
8859-10 (ISO 8859-10)	UCS-2BE
8859-10 (ISO 8859-10)	UCS-2LE
8859-10 (ISO 8859-10)	UCS-4
8859-10 (ISO 8859-10)	UCS4-BE
8859-10 (ISO 8859-10)	UCS-4LE
8859-10 (ISO 8859-10)	UTF-16
8859-10 (ISO 8859-10)	UTF-16BE
8859-10 (ISO 8859-10)	UTF-16LE
8859-10 (ISO 8859-10)	UTF-8
8859-13 (ISO 8859-13)	UCS-2
8859-13 (ISO 8859-13)	UCS-2BE
8859-13 (ISO 8859-13)	UCS-2LE
8859-13 (ISO 8859-13)	UCS-4
8859-13 (ISO 8859-13)	UCS4-BE
8859-13 (ISO 8859-13)	UCS-4LE
8859-13 (ISO 8859-13)	UTF-16
8859-13 (ISO 8859-13)	UTF-16BE
8859-13 (ISO 8859-13)	UTF-16LE
8859-13 (ISO 8859-13)	UTF-8
8859-14 (ISO 8859-14)	UCS-2
8859-14 (ISO 8859-14)	UCS-2BE
8859-14 (ISO 8859-14)	UCS-2LE
8859-14 (ISO 8859-14)	UCS-4
8859-14 (ISO 8859-14)	UCS4-BE
8859-14 (ISO 8859-14)	UCS-4LE

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
8859-14 (ISO 8859-14)	UTF-16
8859-14 (ISO 8859-14)	UTF-16BE
8859-14 (ISO 8859-14)	UTF-16LE
8859-14 (ISO 8859-14)	UTF-8
8859-15 (ISO 8859-15)	UCS-2
8859-15 (ISO 8859-15)	UCS-2BE
8859-15 (ISO 8859-15)	UCS-2LE
8859-15 (ISO 8859-15)	UCS-4
8859-15 (ISO 8859-15)	UCS4-BE
8859-15 (ISO 8859-15)	UCS-4LE
8859-15 (ISO 8859-15)	UTF-16
8859-15 (ISO 8859-15)	UTF-16BE
8859-15 (ISO 8859-15)	UTF-16LE
8859-15 (ISO 8859-15)	UTF-8
8859-2 (ISO 8859-2)	UCS-2
8859-2 (ISO 8859-2)	UCS-2BE
8859-2 (ISO 8859-2)	UCS-2LE
8859-2 (ISO 8859-2)	UCS-4
8859-2 (ISO 8859-2)	UCS4-BE
8859-2 (ISO 8859-2)	UCS-4LE
8859-2 (ISO 8859-2)	UTF-16
8859-2 (ISO 8859-2)	UTF-16BE
8859-2 (ISO 8859-2)	UTF-16LE
8859-2 (ISO 8859-2)	UTF-8
8859-3 (ISO 8859-3)	UCS-2
8859-3 (ISO 8859-3)	UCS-2BE
8859-3 (ISO 8859-3)	UCS-2LE
8859-3 (ISO 8859-3)	UCS-4
8859-3 (ISO 8859-3)	UCS4-BE
8859-3 (ISO 8859-3)	UCS-4LE

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
8859-3 (ISO 8859-3)	UTF-16
8859-3 (ISO 8859-3)	UTF-16BE
8859-3 (ISO 8859-3)	UTF-16LE
8859-3 (ISO 8859-3)	UTF-8
8859-4 (ISO 8859-4)	UCS-2
8859-4 (ISO 8859-4)	UCS-2BE
8859-4 (ISO 8859-4)	UCS-2LE
8859-4 (ISO 8859-4)	UCS-4
8859-4 (ISO 8859-4)	UCS4-BE
8859-4 (ISO 8859-4)	UCS-4LE
8859-4 (ISO 8859-4)	UTF-16
8859-4 (ISO 8859-4)	UTF-16BE
8859-4 (ISO 8859-4)	UTF-16LE
8859-4 (ISO 8859-4)	UTF-8
8859-5 (ISO 8859-5)	UCS-2
8859-5 (ISO 8859-5)	UCS-2BE
8859-5 (ISO 8859-5)	UCS-2LE
8859-5 (ISO 8859-5)	UCS-4
8859-5 (ISO 8859-5)	UCS4-BE
8859-5 (ISO 8859-5)	UCS-4LE
8859-5 (ISO 8859-5)	UTF-16
8859-5 (ISO 8859-5)	UTF-16BE
8859-5 (ISO 8859-5)	UTF-16LE
8859-5 (ISO 8859-5)	UTF-8
8859-6 (ISO 8859-6)	UCS-2
8859-6 (ISO 8859-6)	UCS-2BE
8859-6 (ISO 8859-6)	UCS-2LE
8859-6 (ISO 8859-6)	UCS-4
8859-6 (ISO 8859-6)	UCS4-BE
8859-6 (ISO 8859-6)	UCS-4LE

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
8859-6 (ISO 8859-6)	UTF-16
8859-6 (ISO 8859-6)	UTF-16BE
8859-6 (ISO 8859-6)	UTF-16LE
8859-6 (ISO 8859-6)	UTF-8
8859-7 (ISO 8859-7)	UCS-2
8859-7 (ISO 8859-7)	UCS-2BE
8859-7 (ISO 8859-7)	UCS-2LE
8859-7 (ISO 8859-7)	UCS-4
8859-7 (ISO 8859-7)	UCS4-BE
8859-7 (ISO 8859-7)	UCS-4LE
8859-7 (ISO 8859-7)	UTF-16
8859-7 (ISO 8859-7)	UTF-16BE
8859-7 (ISO 8859-7)	UTF-16LE
8859-7 (ISO 8859-7)	UTF-8
8859-8 (ISO 8859-8)	UCS-2
8859-8 (ISO 8859-8)	UCS-2BE
8859-8 (ISO 8859-8)	UCS-2LE
8859-8 (ISO 8859-8)	UCS-4
8859-8 (ISO 8859-8)	UCS4-BE
8859-8 (ISO 8859-8)	UCS-4LE
8859-8 (ISO 8859-8)	UTF-16
8859-8 (ISO 8859-8)	UTF-16BE
8859-8 (ISO 8859-8)	UTF-16LE
8859-8 (ISO 8859-8)	UTF-8
8859-9 (ISO 8859-9)	UCS-2
8859-9 (ISO 8859-9)	UCS-2BE
8859-9 (ISO 8859-9)	UCS-2LE
8859-9 (ISO 8859-9)	UCS-4
8859-9 (ISO 8859-9)	UCS4-BE
8859-9 (ISO 8859-9)	UCS-4LE

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
8859-9 (ISO 8859-9)	UTF-16
8859-9 (ISO 8859-9)	UTF-16BE
8859-9 (ISO 8859-9)	UTF-16LE
8859-9 (ISO 8859-9)	UTF-8
eucJP	UTF-8
gb2312	UTF-8
iso2022	UTF-8
ko_KR-cp933	UTF-8
ko_KR-euc	UTF-8
ko_KR-iso2022–7	UTF-8
ko_KR-johap	UTF-8
ko_KR-johap92	UTF-8
zh_TW-euc	UTF-8
zh_TW-big5	UTF-8
zh_TW-cp937	UTF-8
zh_TW-iso2022–7	UTF-8
GBK	UTF-8
ISO-2022–JP	UTF-8
KOI8-R	UCS-2
KOI8-R	UCS-2BE
KOI8-R	UCS-2LE
KOI8-R	UCS-4
KOI8-R	UCS4-BE
KOI8-R	UCS-4LE
KOI8-R	UTF-8
KOI8-R	UTF-16
KOI8-R	UTF-16BE
KOI8-R	UTF-16LE
KOI8-R	UTF-8
KOI8-U	UCS-2

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
KOI8-U	UCS-2BE
KOI8-U	UCS-2LE
KOI8-U	UCS-4
KOI8-U	UCS-4BE
KOI8-U	UCS-4LE
KOI8-U	UTF-16
KOI8-U	UTF-16BE
KOI8-U	UTF-16LE
KOI8-U	UTF–8
PCK	UTF-8
UCS-2	646 (ISO 646)
UCS-2	8859-1 (ISO8859-1)
UCS-2	8859-10 (ISO 8859-10)
UCS-2	8859-13 (ISO 8859-13)
UCS-2	8859-14 (ISO 8859-14)
UCS-2	8859-15 (ISO8859-15)
UCS-2	8859-2 (ISO8859-2)
UCS-2	8859-3 (ISO8859-3)
UCS-2	8859-4 (ISO8859-4)
UCS-2	8859-5 (ISO8859-5)
UCS-2	8859-6 (ISO8859-6)
UCS-2	8859-7 (ISO8859-7)
UCS-2	8859-8 (ISO8859-8)
UCS-2	8859-9 (ISO8859-9)
UCS-2	KOI8-R
UCS-2	KOI8-U
UCS-2	UCS-4
UCS-2	UCS-4BE
UCS-2	UCS-4LE
UCS-2	UTF-7

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
UCS-2	UTF-8
UCS-2BE	646 (ISO 646)
UCS-2BE	8859-1 (ISO 8859-1)
UCS-2BE	8859-10 (ISO 8859-10)
UCS-2BE	8859-13 (ISO 8859-13)
UCS-2BE	8859-14 (ISO 8859-14)
UCS-2BE	8859-15 (ISO 8859-15)
UCS-2BE	8859-2 (ISO 8859-2)
UCS-2BE	8859-3 (ISO 8859-3)
UCS-2BE	8859-4 (ISO 8859-4)
UCS-2BE	8859-5 (ISO 8859-5)
UCS-2BE	8859-6 (ISO 8859-6)
UCS-2BE	8859-7 (ISO 8859-7)
UCS-2BE	8859-8 (ISO 8859-8)
UCS-2BE	8859-9 (ISO8859-9)
UCS-2BE	KOI8-R
UCS-2BE	KOI8-U
UCS-2BE	UCS-4
UCS-2BE	UCS-4BE
UCS-2BE	UCS-4LE
UCS-2BE	UTF-8
UCS-4	UTF-8
UCS-4LE	646 (ISO 646)
UCS-4LE	8859-1 (ISO 8859-1)
UCS-4LE	8859 -10 (ISO 8859-10)
UCS-4LE	8859-13 (ISO 8859-13)
UCS-4LE	8859-14 (ISO 8859-14)
UCS-4LE	8859-15 (ISO 8859-15)
UCS-4LE	8859-2 (ISO 8859-2)
UCS-4LE	8859-3 (ISO 8859-3)

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
UCS-4LE	8859-4 (ISO 8859-4)
UCS-4LE	8859-5 (ISO 8859-5)
UCS-4LE	8859-6 (ISO 8859-6)
UCS-4LE	8859-7 (ISO 8859-7)
UCS-4LE	8859-8 (SO 8859-8)
UCS-4LE	8859-9 (ISO8859-9)
UCS-4LE	KOI8-R
UCS-4LE	KOI8-U
UCS-4LE	UCS-2
UCS-4LE	UCS-2BE
UCS-4LE	UCS-2LE
UCS-4LE	UTF-16
UCS-4LE	UTF-16BE
UCS-4LE	UTF-16LE
UCS-4LE	UTF-8
UTF-7	UTF-8
UTF-8	646
UTF-8	8859–1
UTF-8	8859–2
UTF-8	8859–3
UTF-8	8859–4
UTF-8	8859–5
UTF-8	8859–6
UTF-8	8859–7
UTF-8	8859–8
UTF-8	8859–9
UTF-8	8859–10
UTF-8	8859–11
UTF-8	8859–15
UTF-8	eucJP

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
UTF-8	gb2312
UTF-8	iso2022
UTF-8	ko_KR-euc
UTF-8	ko_KR-johap
UTF-8	ko_KR-johap92
UTF-8	ko_KR-iso2022–7
UTF-8	zh_TW-euc
UTF-8	zh_TW-big5
UTF-8	zh_TW-iso2022–7
UTF-8	zh_TW-cp937
UTF-8	GBK
UTF-8	ISO-2022–JP
UTF-8	KOI8–R
UTF-8	PCK
UTF-8	UCS-2
UTF-8	UCS-4
UTF-8	UTF-7
UTF-8	UTF-16
UTF-16	646 (ISO 646)
UTF-16	8859-1 (ISO8859-1)
UTF-16	8859-10 (ISO8859-10)
UTF-16	8859-13 (ISO8859-13)
UTF-16	8859-14 (ISO8859-14)
UTF-16	8859-15 (ISO8859-15)
UTF-16	8859-2 (ISO8859-2)
UTF-16	8859-3 (ISO8859-3)
UTF-16	8859-4 (ISO8859-4)
UTF-16	8859-5 (ISO8859-5)
UTF-16	8859-6 (ISO8859-6)
UTF-16	8859-7 (ISO8859-7)

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
UTF-16	8859-8 (ISO8859-8)
UTF-16	8859-9 (ISO8859-9)
UTF-16	KOI8-R
UTF-16	KOI8-U
UTF-16	UCS-4
UTF-16	UCS-4BE
UTF-16	UCS-4LE
UTF-16	UTF-8
UTF-16BE	646 (ISO 646)
UTF-16BE	8859-1 (ISO 8859-1)
UTF-16BE	8859-10(ISO 8859-10)
UTF-16BE	8859-13 (ISO 8859-13)
UTF-16BE	8859-14 (ISO 8859-14)
UTF-16BE	8859-15 (ISO 8859-15)
UTF-16BE	8859-2 (ISO 8859-2)
UTF-16BE	8859-3 (ISO 8859-3)
UTF-16BE	8859-4 (ISO 8859-4)
UTF-16BE	8859-5 (ISO 8859-5)
UTF-16BE	8859-6 (ISO 8859-6)
UTF-16BE	8859-7 (ISO 8859-7)
UTF-16BE	8859-8 (ISO 8859-8)
UTF-16BE	8859-9 (ISO 8859-9)
UTF-16BE	KOI8-R
UTF-16BE	KOI8-U
UTF-16BE	UCS-4
UTF-16BE	UCS-4BE
UTF-16BE	UCS-4LE
UTF-16BE	UTF-8
UTF-16LE	646 (ISO 646)
UTF-16LE	8859-1 (ISO 8859-1)

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
UTF-16LE	8859-10 (ISO 8859-10)
UTF-16LE	8859-13 (ISO 8859-13)
UTF-16LE	8859-14 (ISO 8859-14)
UTF-16LE	8859-15 (ISO 8859-15)
UTF-16LE	8859-2 (ISO 8859-2)
UTF-16LE	8859-3 (ISO 8859-3)
UTF-16LE	8859-4 (ISO 8859-4)
UTF-16LE	8859-5 (ISO 8859-5)
UTF-16LE	8859-6 (ISO 8859-6)
UTF-16LE	8859-7 (ISO 8859-7)
UTF-16LE	8859 -8 (ISO 8859-8)
UTF-16LE	8859-9 (ISO 8859-9)
UTF-16LE	KOI8-R
UTF-16LE	KOI8-U
UTF-16LE	UCS-4
UTF-16LE	UCS-4BE
UTF-16LE	UCS-4LE
UTF-16LE	UTF-8
UTF-7	UCS-2
UTF-7	UCS-4
UTF-7	UCS-8
UTF-8	646 (ISO 646)
UTF-8	8859-1 (ISO 8859-1)
UTF-8	8859-10 (ISO 8859-10)
UTF-8	8859-13 (ISO 8859-13)
UTF-8	8859-14 (ISO 8859-14)
UTF-8	8859-15 (ISO 8859-15)
UTF-8	8859-2 (ISO 8859-2)
UTF-8	8859-3 (ISO 8859-3)
UTF-8	8859-4 (ISO 8859-4)

TABLE A–1 Available Unicode Related `iconv` Code Conversion Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
UTF-8	8859-5 (ISO 8859-5)
UTF-8	8859-6 (ISO 8859-6)
UTF-8	8859-7 (ISO 8859-7)
UTF-8	8859-8 (ISO 8859-8)
UTF-8	8859-9 (ISO 8859-9)
UTF-8	KOI8-R
UTF-8	KOI8-U
UTF-8	UCS-2
UTF-8	UCS-2BE
UTF-8	UCS-2LE
UTF-8	UCS-4
UTF-8	UCS-4BE
UTF-8	UCS-4LE
UTF-8	UTF-16
UTF-8	UTF-16BE
UTF-8	UCS-16LE
UTF-8	UTF-7

UTF-EBCDIC is a new IBM codepage name. Starting with the Solaris 8 environment, we are also supporting bidirectional UTF-8 <—> UTF-EBCDIC conversion.

TABLE A–2 Available Unicode and IBM/Microsoft EBCDIC and PC Code Page Related `iconv` Code Conversions Modules in the Solaris 8 Environment

From Code (Symbol)	To Code (Symbol)
UTF-8	IBM-037
UTF-8	IBM-273
UTF-8	IBM-277
UTF-8	IBM-278

TABLE A–2 Available Unicode and IBM/Microsoft EBCDIC and PC Code Page Related `iconv` Code Conversions Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
UTF-8	IBM-280
UTF-8	IBM-284
UTF-8	IBM-285
UTF-8	IBM-297
UTF-8	IBM-420
UTF-8	IBM-424
UTF-8	IBM-500
UTF-8	IBM-870
UTF-8	IBM-875
UTF-8	IBM-880
UTF-8	IBM-1025
UTF-8	IBM-1026
UTF-8	IBM-1112
UTF-8	IBM-1122
UTF-8	IBM-850
UTF-8	IBM-852
UTF-8	IBM-855
UTF-8	IBM-856
UTF-8	IBM-857
UTF-8	IBM-862
UTF-8	IBM-864
UTF-8	IBM-866
UTF-8	IBM-869
UTF-8	IBM-921
UTF-8	IBM-922
UTF-8	IBM-1046

TABLE A–2 Available Unicode and IBM / Microsoft EBCDIC and PC Code Page Related `iconv` Code Conversions Modules in the Solaris 8 Environment *(continued)*

From Code (Symbol)	To Code (Symbol)
UTF-8	CP850
UTF-8	CP852
UTF-8	CP855
UTF-8	CP857
UTF-8	CP862
UTF-8	CP864
UTF-8	CP866
UTF-8	CP869
UTF-8	CP874
UTF-8	CP1250
UTF-8	CP1251
UTF-8	CP1252
UTF-8	CP1253
UTF-8	CP1254
UTF-8	CP1255
UTF-8	CP1256
UTF-8	CP1257
UTF-8	CP1258

TABLE A–3 Available `iconv` Code Conversions IBM and Microsoft EBCDIC / PC Code Pages to UTF-8

UTF-EBCDIC	UTF-8
IBM-037	UTF-8
IBM-273	UTF-8
IBM-277	UTF-8

TABLE A–3 Available `iconv` Code Conversions IBM and Microsoft EBCDIC/PC Code Pages to UTF-8 *(continued)*

IBM-278	UTF-8
IBM-280	UTF-8
IBM-284	UTF-8
IBM-285	UTF-8
IBM-297	UTF-8
IBM-420	UTF-8
IBM-424	UTF-8
IBM-500	UTF-8
IBM-870	UTF-8
IBM-875	UTF-8
IBM-880	UTF-8
IBM-1025	UTF-8
IBM-1026	UTF-8
IBM-1112	UTF-8
IBM-1122	UTF-8
IBM-850	UTF-8
IBM-852	UTF-8
IBM-855	UTF-8
IBM-856	UTF-8
IBM-857	UTF-8
IBM-862	UTF-8
IBM-864	UTF-8
IBM-866	UTF-8
IBM-869	UTF-8
IBM-921	UTF-8
IBM-922	UTF-8
IBM-1046	UTF-8

TABLE A–3 Available `iconv` Code Conversions IBM and Microsoft EBCDIC/PC Code Pages to UTF-8 *(continued)*

CP850	UTF-8
CP852	UTF-8
CP855	UTF-8
CP857	UTF-8
CP862	UTF-8
CP864	UTF-8
CP866	UTF-8
CP869	UTF-8
CP874	UTF-8
CP1250	UTF-8
CP1251	UTF-8
CP1252	UTF-8
CP1253	UTF-8
CP1254	UTF-8
CP1255	UTF-8
CP1256	UTF-8
CP1257	UTF-8
CP1258	UTF-8

APPENDIX **B**

Partial L10N Package Names on OS CD

TABLE B–1 List of Partial Locales

Package Name	Description
SUNWauaox	Australasia 64-bit OS Support
SUNWauadt	Australasia CDE Support
SUNWauaos	Australasia OS Support
SUNWauaow	Australasia OW Support
SUNWcamox	Central America 64-bit OS Support
SUNWcamdt	Central America CDE Support
SUNWcamos	Central America OS Support
SUNWcamow	Central America OW Support
SUNWceuox	Central Europe 64-bit OS Support
SUNWceudt	Central Europe CDE Support
SUNWceuos	Central Europe OS Support
SUNWceuow	Central Europe OW Support
SUNWalex	Common files shared by Chinese, Japanese and Korean locales. It is a required package to run Asian Language Environment (64-bit)
SUNWeeuox	Eastern Europe 64-bit OS Support
SUNWeeudt	Eastern Europe CDE Support

TABLE B–1 List of Partial Locales *(continued)*

Package Name	Description
SUNWeeuos	Eastern Europe OS Support
SUNWeeuow	Eastern Europe OW Support
SUNWfris	French install software localization
SUNWdeis	German install software localization
SUNWitis	Italian install software localization
NSCPjacom	Japanese (common) localization of Netscape Communicator 4.7 supporting International security.
SUNWjc0r	Japanese Kana-Kanji Conversion Server cs00 Root Files
SUNWjc0u	Japanese Kana-Kanji Conversion Server cs00 User Files
SUNWjedt	Japanese (EUC) Localization for CDE DESKTOP LOGIN ENVIRONMENT
SUNWjeuc	Japanese (EUC) Feature Package specific files for `usr`; it is a required package to support EUC environment.
SUNWjeucx	Japanese (EUC) Feature Package specific 64-bit files for `usr`; it is a required package to run JFP environment.
SUNWjexpl	Japanese (EUC) Localizations for X Window System platform software.
SUNWjexpx	Japanese (EUC) Localizations for X Window System platform software (64-bit)
SUNWjfpr	Stream modules for Japanese Feature Package (JFP); it is a required package to run JFP environment.
SUNWjfpu	Japanese Feature Package (JFP) specific files for `usr`; it is a required package to run JFP environment.
SUNWjfpux	Japanese Feature Package (JFP) specific 64-bit files for `usr`; it is a required package to run JFP environment.
SUNWjman	Japanese Feature Package Man Pages to see English man pages for `SUNWjfpr` and `SUNWjfpu`
SUNWjpck	Japanese (PCK - PC Kanji Code) Feature Package specific files; it's a required package to support PCK environment.
SUNWjpckx	Japanese (PCK) Feature Package specific 64-bit files for `usr`; it is a required package to run JFP environment.

TABLE B–1 List of Partial Locales *(continued)*

Package Name	Description
SUNWjpdt	Japanese (PCK) Localization for CDE DESKTOP LOGIN ENVIRONMENT
SUNWjpxpl	Japanese (PCK) Localizations for X Window System platform software
SUNWjpxpx	Japanese (PCK) Localizations for X Window System platform software (64-bit)
SUNWju8	Japanese (UTF-8) Feature Package specific files; it's a required package to support Japanese UTF-8 environment.
SUNWju8x	Japanese (UTF-8) Feature Package specific 64-bit files for `usr`; it is a required package to run JFP environment.
SUNWjudt	Japanese (UTF-8) Localization for CDE DESKTOP LOGIN ENVIRONMENT
SUNWjuxpl	Japanese (UTF-8) Localizations for X Window System platform software
SUNWjxcft	Japanese JISX212 TrueType and bitmap fonts
SUNWkleux	Korean (EUC) Language Environment specific files. It is a required package to run Korean Language Environment (64-bit).
SUNWkulex	Korean (UTF-8) Language Environment specific files. It is a required package to run Korean Language Environment (64-bit)
SUNWkdt	Korean Localizations for CDE Desktop Login Environment.
SUNSCPcpcom	Simplified Chinese partial version of Netscape Communicator 4.7 supporting International security
SUNWsamox	Southern America 64-bit OS Support
SUNWsamdt	Southern America CDE Support
SUNWsamos	Southern America OS Support
SUNWsamow	Southern America OW Support
SUNWseuox	Southern Europe 64-bit OS Support
SUNWseudt	Southern Europe CDE Support
SUNWseuos	Southern Europe OS Support

TABLE B–1 List of Partial Locales *(continued)*

Package Name	Description
SUNWseuow	Southern Europe OW Support
SUNWfrspl	Spell Checking Engine - French Dictionary
SUNWdespl	Spell Checking Engine - German Dictionary
SUNWitspl	Spell Checking Engine - Italian Dictionary
SUNWesspl	Spell Checking Engine - Spanish Dictionary
SUNWsvspl	Spell Checking Engine - Swedish Dictionary
SUNWjfpr	Stream modules for Japanese Feature Package (JFP), it is a required package to run JFP environment
SUNWsvis	Swedish install software localization
SUNWkleu	This package contains Korean Language Environment specific files. It is a required package to run Korean Language Environment.
SUNWkuleu	This package contains Korean UTF-8 Language Environment specific files. It is a required package to run Korean Language Environment.
SUNWcleu	This package contains Korean UTF-8 Language Environment specific files. It is a required package to run Korean Language Environment.
SUNWgleu	This package contains Simplified Chinese (GBK) Language Environment specific files. It is a required package to run Simplified Chinese (GBK) Language Environment.
SUNWculeu	This package contains Simplified Chinese (UTF-8) Language Environment specific files. It is a required package to run Simplified Chinese (UTF-8) Language Environment.
SUNWtleu	This package contains Thai Language Environment specific files. It is a required package to run Thai Language Environment.
SUNWhuleu	This package contains Traditional Chinese (UTF-8) Language Environment specific files. It is a required package to run Traditional Chinese UTF-8 Language Environment.
SUNW5leu	This package contains Traditional Chinese Language Environment specific files. It is a required package to run Traditional Chinese BIG5 Language Environment

TABLE B–1 List of Partial Locales *(continued)*

Package Name	Description
SUNWhleu	This package contains Traditional Chinese Language Environment specific files. It is a required package to run Traditional Chinese Language Environment
SUNWale	This package contains common files shared by Chinese, Japanese and Korean locales. It is a required package to run Asian Language Environment
SUNWaled	This package contains man pages shared by Chinese, Japanese and Korean locales.
SUNW5leux	Traditional Chinese (BIG5) Language Environment user files (64-bit)
SUNW5xplx	Traditional Chinese (BIG5) X Windows Platform Software Package (64-bit)
SUNWhleux	Traditional Chinese (EUC) Language Environment specific files. It is a required package to run Traditional Chinese Language Environment (64-bit)
SUNWhulex	Traditional Chinese (UTF-8) Language Environment user files (64-bit)
SUNW5xplt	Traditional Chinese BIG5 X Windows Platform Software Package
SUNW5dt	raditional Chinese Localizations for CDE Desktop Login Environment
SUNWhdt	Traditional Chinese Localizations for CDE Desktop Login Environment
SUNWhicd	Traditional Chinese Solaris Install CD L10N source files
SUNW5ttf	Traditional Chinese True Type Fonts Package
SUNWhttf	Traditional Chinese True Type Fonts Package
SUNWhuplt	Traditional Chinese UTF-8 X Windows Platform Software Package
SUNWhxplt	Traditional Chinese X Windows Platform Software Package
SUNWhxfnt	Traditional Chinese X Windows Platform required Fonts Package
SUNSCPhpcom	Traditional Chinese partial version of Netscape Communicator 4.7 supporting International security

TABLE B–1 List of Partial Locales *(continued)*

Package Name	Description
SUNWweuox	Western Europe 64-bit OS Support
SUNWweudt	Western Europe CDE Support
SUNWweuos	Western Europe OS Support
SUNWweuow	Western Europe OW Support
SUNWi1cs	X11 ISO8859-1 Codeset Support
SUNWi13cs	X11 ISO8859-13 Codeset Support
SUNWi15cs	X11 ISO8859-15 Codeset Support
SUNWi2cs	X11 ISO8859-2 Codeset Support
SUNWi5cs	X11 ISO8859–5 Codeset Support
SUNWi7cs	X11 ISO8859–7 Codeset Support
SUNWi9cs	X11 ISO8859–9 Codeset Support
SUNWi2of	X11 fonts for ISO-8859-2 character set (optional fonts)
SUNWi4of	X11 fonts for ISO-8859-4 character set (optional fonts)
SUNWi5of	X11 fonts for ISO-8859-5 character set (optional fonts)
SUNWi7of	X11 fonts for ISO-8859-7 character set (optional fonts)
SUNWi9of	X11 fonts for ISO-8859-9 character set (optional fonts)
SUNWkoi8f	X11 fonts for KOI8–R character set

APPENDIX **C**

Languages CD Packages List

TABLE C–1 Simplified Chinese

NSCPccom	Simplified Chinese localization of Netscape Communicator 4.7 supporting International security.
NSCPcucom	zh.UTF-8 localization of Netscape Communicator 4.7 supporting International security.
NSCPgcom	zh.GBK localization of Netscape Communicator 4.7 supporting International security.
SUNWcadis	Simplified Chinese (EUC) Localizations for admintool and GUI install.
SUNWcadma	Simplified Chinese (EUC) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNWhadis packages for Simplified Chinese (EUC) localization.
SUNWcbcp	This package contains Simplified Chinese (EUC) Language Environment binary compatibility files.
SUNWcdab	Simplified Chinese (EUC) Localizations for CDE Desktop Application Builder
SUNWcdbas	Simplified Chinese (EUC) Localizations for CDE Base functionality
SUNWcddst	Simplified Chinese (EUC) Localizations for CDE Desktop Applications
SUNWcddte	Simplified Chinese (EUC) Localizations for CDE Desktop Login Environment
SUNWcdezt	Simplified Chinese (EUC) Localizations for Desktop Power Pack Applications

TABLE C–1 Simplified Chinese *(continued)*

SUNWcdft	Simplified Chinese (EUC) Localizations for CDE Fonts
SUNWcdhe	Simplified Chinese (EUC) Localizations for CDE Help Runtime environment
SUNWcdhev	Simplified Chinese (EUC) CDE Help Volumes
SUNWcdhez	Simplified Chinese (EUC) (Common) Desktop Power Pack Help Volumes
SUNWcdicn	Simplified Chinese (EUC) Localizations for CDE Icons
SUNWcdim	Simplified Chinese (EUC) Localizations for CDE Imagetool
SUNWcdwm	Simplified Chinese (EUC) Localizations for CDE Desktop Window Manager
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
SUNWcj2p	Simplified Chinese localization of Java Plug-in 1.2.2
SUNWcj2rt	Java virtual machine and core class libraries (Simplified Chinese supplement)
SUNWcjvdv	Simplified Chinese Localizations for JavaVM developers package
SUNWcjvrt	Simplified Chinese Localizations for JavaVM run time environment
SUNWckcsr	Simplified Chinese (EUC) KCMS Runtime Environment
SUNWcleue	This package contains Simplified Chinese (EUC) Language Environment specific files. It is a required package to run Simplified Chinese (EUC) Language Environment
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

TABLE C–1 Simplified Chinese *(continued)*

SUNWcorte	Simplified Chinese (EUC) OPENLOOK Toolkits Runtime Environment Package
SUNWcrdm	Simplified Chinese (EUC) OILBN ReadMe Directory
SUNWcreg	Simplified Chinese (EUC) Localizations for Solaris User Registration
SUNWcsadl	Simplified Chinese (EUC) Localizationsfor Solstice Admintool launcher and associated libraries.
SUNWctltk	Simplified Chinese (EUC) ToolTalk Runtime Package Package
SUNWcttfe	Simplified Chinese (EUC) True Type Fonts
SUNWcuada	Simplified Chinese (UTF-8) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNWgadis packages for Simplified Chinese (UTF-8) localization.
SUNWcuadi	Simplified Chinese (UTF-8) Localizations for admintool and GUI install
SUNWcubas	Simplified Chinese (UTF-8) Localizations for CDE Base functionality
SUNWcudab	Simplified Chinese (UTF-8) Localizations for CDE Desktop Application Builder
SUNWcudc	Simplified Chinese (EUC) Localizations for User Defined Character tool for Solaris CDE environment
SUNWcudez	Simplified Chinese (UTF-8) Localizations for Desktop Power Pack Applications
SUNWcudft	Simplified Chinese (UTF-8) Localizations for CDE Fonts
SUNWcudhe	Simplified Chinese (UTF-8) Localizations for CDE Help Runtime environment
SUNWcudhv	Simplified Chinese (UTF-8) CDE Help Volumes
SUNWcudhz	Simplified Chinese (UTF-8) Localizations for Desktop Power Pack Help Volumes
SUNWcudic	Simplified Chinese (UTF-8) Localizations for CDE Icons
SUNWcudim	Simplified Chinese (UTF-8) L10N for CDE DESKTOP Imagetool
SUNWcudst	Simplified Chinese (UTF-8) Localizations for CDE Desktop Applications
SUNWcudte	Simplified Chinese (UTF-8) Localizations for CDE Desktop Login Environment
SUNWcudwm	Simplified Chinese (UTF-8) Localizations for CDE Desktop Window Manager

TABLE C–1 Simplified Chinese *(continued)*

SUNWculee	This package contains Simplified Chinese (UTF-8) Language Environment specific files. It is a required package to run Simplified Chinese (UTF-8) Language Environment
SUNWcuman	Simplified Chinese (UTF-8) X Windows Online User Man Pages Package
SUNWcuodt	Simplified Chinese (UTF-8) Core OPENLOOK Desktop Package
SUNWcupmw	Simplified Chinese (UTF-8) Localization for Power Management OW Utilities
SUNWcurdm	Simplified Chinese (UTF-8) OILBN ReadMe Directory
SUNWcureg	Simplified Chinese (UTF-8) Localizations for Solaris User Registration
SUNWcusad	Simplified Chinese (UTF-8) Localizations for Solstice Admintool launcher and associated libraries.
SUNWcuudc	Simplified Chinese (UTF-8) Localizations for User Defined Character tool for Solaris CDE environment
SUNWcuxe	Simplified Chinese (UTF-8) X Windows Platform Software Package
SUNWcwbcp	Simplified Chinese (EUC) OpenWindows Binary Compatibility Package
SUNWcwsr	Simplified Chinese (EUC) prodreg 2.0 localizable text resources
SUNWcxe	Simplified Chinese (EUC) X Windows Platform Software Package
SUNWcxfnt	Simplified Chinese (EUC) X Windows Platform Required Fonts
SUNWcxman	Simplified Chinese (EUC) X Windows Online User Man Pages Package
SUNWcxoft	Simplified Chinese (EUC) X Windows Optional Fonts Package
SUNWgadis	Simplified Chinese (GBK) Localizations for admintool and GUI install
SUNWgadma	Simplified Chinese (GBK) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNWgadis packages for Simplified Chinese (GBK) localization.
SUNWgdab	Simplified Chinese (GBK) Localizations for CDE Desktop Application Builder
SUNWgdbas	Simplified Chinese (GBK) Localizations for CDE Base functionality
SUNWgddst	Simplified Chinese (GBK) Localizations for CDE Desktop Applications
SUNWgddte	Simplified Chinese (GBK) Localizations for CDE Desktop Login Environment
SUNWgdezt	Simplified Chinese (GBK) Localizations for Desktop Power Pack Applications

TABLE C–1 Simplified Chinese *(continued)*

SUNWgdft	Simplified Chinese (GBK) Localizations for CDE Fonts
SUNWgdhe	Simplified Chinese (GBK) Localizations for CDE Help Runtime environment
SUNWgdhev	Simplified Chinese (GBK) CDE Help Volumes
SUNWgdhez	Simplified Chinese (GBK) Localizations for Desktop Power Pack Help Volumes
SUNWgdicn	Simplified Chinese (GBK) Localizations for CDE Icons
SUNWgdim	Simplified Chinese (GBK) L10N for CDE DESKTOP Imagetool
SUNWgdwm	Simplified Chinese (GBK) Localizations for CDE Desktop Window Manager
SUNWgleue	This package contains Simplified Chinese (GBK) Language Environment specific files. It is a required package to run Simplified Chinese (GBK) Language Environment
SUNWgodte	Simplified Chinese (GBK) Core OPENLOOK Desktop Package
SUNWgpmw	Simplified Chinese (GBK) Localization for Power Management OW Utilities
SUNWgrdm	Simplified Chinese (GBK) OILBN ReadMe Directory
SUNWgreg	Simplified Chinese (GBK) Localizations for Solaris User Registration
SUNWgsadl	Simplified Chinese (GBK) Localizations for Solstice Admintool launcher and associated libraries.
SUNWgttfe	Simplified Chinese (GBK) True Type Fonts
SUNWgudc	Simplified Chinese (GBK) Localizations for User Defined Character tool for Solaris CDE environment
SUNWgxe	Simplified Chinese (GBK) X Windows Platform Software Package
SUNWgxman	Simplified Chinese (GBK) X Windows Online User Man Pages Package

TABLE C–2 French

NSCPfrcdo	French localization of Netscape Communicator 4.7 supporting U.S. security.
NSCPfrcom	French localization of Netscape Communicator 4.7 supporting International security.
SUNWf8bas	Base L10N fr CDE functionality to run a CDE application
SUNWf8dst	CDE Desktop Applications
SUNWf8dte	CDE Desktop Environment
SUNWf8he	CDE Help L10N fr Runtime Environment
SUNWf8im	CDE Desktop apps
SUNWf8wm	French UTF-8 CDE Desktop Window Manages Messages
SUNWfbcp	French OS Binary Compatibility Package
SUNWfoaud	French French OPEN LOOK (R) Audio applications
SUNWfobk	French OpenWindows online handbooks
SUNWfodcv	French OPEN LOOK (R) document and help viewer applications
SUNWfodem	French OPEN LOOK (R) demo programs
SUNWfodst	French OPEN LOOK (R) deskset tools
SUNWfodte	French OPEN LOOK (R) desktop environment
SUNWfoimt	French OPEN LOOK (R) imagetool
SUNWforte	French OPEN LOOK (R) toolkits runtime environment
SUNWfrbas	Base L10N fr CDE functionality to run a CDE application
SUNWfrdst	CDE Desktop Applications
SUNWfrdte	CDE Desktop Environment
SUNWfrhe	CDE Help L10N fr Runtime Environment
SUNWfrhed	CDE L10N fr Help Developer Environment
SUNWfrhev	CDE Help Volumes
SUNWfrim	CDE Desktop apps
SUNWfrj2p	French localization of Java Plug-in 1.2.2
SUNWfros	localizable message files for the OS-Networking consolidation

TABLE C–2 French *(continued)*

SUNWfrpmw	French (EUC) Localizations for Power Management OW Utilities
SUNWfrreg	Solaris User Registration prompts at desktop login for user registration
SUNWfrwm	French CDE Desktop Window Manages Messages
SUNWftltk	French ToolTalk binaries and shared libraries
SUNWfwacx	French OPEN LOOK (R) AccessX
SUNWfwbcp	French OpenWindows Binary Compatibility Package
SUNWfxplt	French X Windows platform software

TABLE C–3 German

NSCPdecom	German localization of Netscape Communicator 4.7 supporting International security.
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 apps
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
SUNWdedst	CDE Desktop Applications
SUNWdedte	CDE Desktop Login Environment
SUNWdehe	CDE Help L10N German Runtime Environment
SUNWdehed	CDE L10N German Help Developer Environment

TABLE C–3 German *(continued)*

SUNWdehev	CDE Help Volumes
SUNWdeim	CDE Desktop apps
SUNWdej2p	German localization of Java Plug-in 1.2.2
SUNWdeos	localizable message files for the OS-Networking consolidation
SUNWdepmw	German (EUC) Localizations for Power Management OW Utilities
SUNWdereg	Solaris User Registration prompts at desktop login for user registration
SUNWdewm	German CDE Desktop Window Manages Messages
SUNWdoaud	German OPEN LOOK (R) Audio applications
SUNWdobk	German OpenWindows online handbooks
SUNWdodcv	German OPEN LOOK (R) document and help viewer applications
SUNWdodem	German OPEN LOOK (R) demo programs
SUNWdodst	German OPEN LOOK (R) deskset tools
SUNWdodte	German OPEN LOOK (R) desktop environment
SUNWdoimt	German OPEN LOOK (R) imagetool
SUNWdorte	German OPEN LOOK (R) toolkits runtime environment
SUNWdtltk	German ToolTalk binaries and shared libraries
SUNWdwacx	German OPEN LOOK (R) AccessX
SUNWdwbcp	German OpenWindows Binary Compatibility Package
SUNWdxplt	German X Windows platform software

TABLE C–4 Italian

NSCPitcom	Italian localization of Netscape Communicator 4.7 supporting International security.
SUNWi8bas	Base L10N it CDE functionality to run a CDE application
SUNWi8dst	CDE it Desktop Applications messages
SUNWi8dte	CDE Italian UTF-8 Desktop Login Environment

TABLE C–4 Italian *(continued)*

SUNWi8he	CDE Help L10N it Runtime Environment
SUNWi8im	CDE Italian UTF-8 Desktop Image editor
SUNWi8wm	Italian UTF-8 CDE Desktop Window Manages Messages
SUNWibcp	Italian OS Binary Compatibility Package
SUNWioaud	Italian OPEN LOOK (R) Audio applications
SUNWiobk	Italian OpenWindows online handbooks
SUNWiodcv	Italian OPEN LOOK (R) document and help viewer applications
SUNWiodem	Italian OPEN LOOK (R) demo programs
SUNWiodst	Italian OPEN LOOK (R) deskset tools
SUNWiodte	Italian OPEN LOOK (R) desktop environment
SUNWioimt	Italian OPEN LOOK (R) imagetool
SUNWiorte	Italian OPEN LOOK (R) toolkits runtime environment
SUNWitbas	Base L10N it CDE functionality to run a CDE application
SUNWitdst	CDE it Desktop Applications messages
SUNWitdte	CDE Italian Desktop Login Environment
SUNWithe	CDE Help L10N it Runtime Environment
SUNWithed	CDE L10N it Help Developer Environment
SUNWithev	CDE Help Volumes
SUNWitim	CDE Italian Desktop Image editor
SUNWitj2p	Italian localization of Java Plug-in 1.2.2
SUNWitltk	Italian ToolTalk binaries and shared libraries
SUNWitos	localizable message files for the OS-Networking consolidation
SUNWitpmw	Italian (EUC) Localizations for Power Management OW Utilities
SUNWitreg	Solaris User Registration prompts at desktop login for user registration
SUNWitwm	Italian CDE Desktop Window Manages Messages
SUNWiwacx	Italian OPEN LOOK (R) AccessX

TABLE C–4 Italian *(continued)*

SUNWiwbcp	Italian OpenWindows Binary Compatibility Package
SUNWixplt	Italian X Windows platform software

TABLE C–5 Japanese

JSat8xw	Japanese Input System ATOK8 for Japanese Solaris.
JSatsvr	Japanese Input System ATOKserver root files for Japanese Solaris
JSatsvu	Japanese Input System ATOKserver usr files for Japanese Solaris
JSatsvw	Japanese Input System ATOKserver X11 support files for Japanese Solaris
NSCPjecom	Japanese (EUC) localization of Netscape Communicator 4.7 supporting International security.
NSCPjpcom	Japanese (PCK) localization of Netscape Communicator 4.7 supporting International security.
NSCPjucom	Japanese (UTF-8) localization of Netscape Communicator 4.7 supporting International security.
SUNWjadis	Japanese (EUC) Localizations for admintool and GUI install.
SUNWjadma	Japanese (EUC) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNWjadis packages for Japanese (EUC) localization.
SUNWjaj2p	Japanese localization of Java Plug-in 1.2.2
SUNWjbcp	Japanese (EUC) utilities including libc and locale data to provide a binary-compatible execution environment for SunOS 4.x applications.
SUNWjc0d	Japanese Kana-Kanji Conversion Server cs00 user dictionary maintenance tool for CDE Motif
SUNWjc0w	Japanese Kana-Kanji Conversion Server cs00 user dictionary maintenance tool for OPEN LOOK. This package is also required to use X Input Method Server on Window System.
SUNWjcs3f	Japanese JIS X0212 Type1 fonts for printing
SUNWjdab	Japanese (Common) Localization for CDE Desktop Application Builder
SUNWjdbas	Japanese (Common) Localization for CDE application basic runtime environment

TABLE C–5 Japanese *(continued)*

SUNWjddst	Japanese (EUC) Localization for CDE Desktop Applications
SUNWjddte	Japanese (EUC) Localization for Solaris Desktop Login Environment
SUNWjdhcm	Japanese Localizations for DHCP Manager
SUNWjdhe	Japanese (EUC) Localization for CDE Help Runtime environment
SUNWjdhed	Japanese (EUC) Localization for CDE Help Developer Environment
SUNWjdhev	Japanese (Common) Localization for CDE Help Volumes
SUNWjdhez	Japanese (Common) Localizations for Desktop Power Pack Help Volumes
SUNWjdim	Japanese (EUC) Localization for Solaris CDE Image Viewer
SUNWjdrme	Japanese (EUC) Localization for Common Desktop Environment (CDE) release documentation
SUNWjdwm	Japanese (EUC) Localization for CDE Desktop Window Manager
SUNWjeab	Japanese (EUC) Localization for CDE Desktop Application Builder
SUNWjebas	Japanese (EUC) Localization for CDE application basic runtime environment
SUNWject	Japanese (EUC) Localizations for UTF-8 Code Conversion Tool
SUNWjedev	Japanese (EUC) Development Environment Package specific files
SUNWjeezt	Japanese (EUC) Localizations for Desktop Power Pack Applications
SUNWjehev	Japanese (EUC) Localization for CDE Help Volumes
SUNWjehez	Japanese (EUC) Localizations for Desktop Power Pack Help Volumes
SUNWjej2m	Japanese (EUC) man pages
SUNWjejmn	Japanese (EUC) JavaVM manual pages for Java programmers and users
SUNWjeman	Japanese Feature Package Man Pages to see Japanese (EUC) manpages for SUNWjfpr and SUNWjfpu and Japanese manpages for SUNWman and SUNWaled.
SUNWjepmm	Japanese (EUC) Power Management OW Utilities Man Pages
SUNWjepmw	Japanese (EUC) Localizations for Power Management OW Utilities
SUNWjervl	Japanese (EUC) Localizations for XIL loadable pipelines for SunVideo capture and compression
SUNWjeuce	Japanese (EUC) Feature Package specific files for usr, it is a extended package to support EUC environment.

TABLE C–5 Japanese *(continued)*

SUNWjeudc	Japanese (EUC) Localizations for User Defined Character tool for Solaris CDE environment
SUNWjewnu	Japanese Input System - Wnn6 Messages, (EUC)
SUNWjexfa	Japanese (EUC) Localizations for Font Administration application for Solaris platforms
SUNWjexir	Japanese (EUC) localizations for XIL Runtime Environment
SUNWjfdl	Japanese Localization for Solaris Desktop Font Downloader for Adobe Postscript printers
SUNWjfpre	Stream modules for Japanese Feature Package (JFP), it is a extended package to run JFP environment.
SUNWjfpue	Japanese Feature Package (JFP) specific files for usr, it is a extended package to run JFP environment.
SUNWjfxmn	English manpages of Japanese features for X Window System.
SUNWjj2dv	Japanese Java virtual macTools and utilities including javac, jdb, javadoc, rmiregistry
SUNWjj2rt	Japanese Java virtual machine and core class libraries
SUNWjjvdv	Japanese Localizations for JavaVM developers package
SUNWjjvrt	Japanese Localizations for JavaVM run time environment
SUNWjkcsr	Japanese (EUC) Localizations for Kodak Color Management System Runtime
SUNWjlibj	Japanese specific library (/usr/lib/libjapanese.a), header and transition kit.
SUNWjmane	Japanese Feature Package Man Pages (Extension) to see English manpages for SUNWjfpre and SUNWjfpue.
SUNWjmfrn	Japanese (EUC) Localizations for Motif 1.2.3 RunTime Kit.
SUNWjoaud	Japanese (EUC) Localizations for Audiotool & other auxiliary audio support
SUNWjodcv	Japanese (EUC) Localizations for OPEN LOOK document and help viewer applications
SUNWjodem	Japanese (EUC) Localizations for OPEN LOOK demo programs
SUNWjodst	Japanese (EUC) Localizations for OPEN LOOK deskset tools
SUNWjodte	Japanese (EUC) Localizations for OPEN LOOK Desktop Environment (olwm, props, wsinfo, etc.)

TABLE C–5 Japanese *(continued)*

SUNWjoimt	Japanese (EUC) Localizations for OPEN LOOK imagetool
SUNWjorte	Japanese (EUC) Localizations for OPEN LOOK toolkits runtime environment
SUNWjoumn	Japanese (EUC) OPEN LOOK toolkit/desktop users man pages
SUNWjpab	Japanese (PCK) Localization for CDE Desktop Application Builder
SUNWjpacx	Japanese (PCK) Localizations for AccessX client program
SUNWjpadi	Japanese (PCK) Localizations for admintool and GUI install.
SUNWjpadm	Japanese (PCK) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNWjpadi packages for Japanese (PCK) localization.
SUNWjpbas	Japanese (PCK) Localization for CDE application basic runtime environment
SUNWjpcke	Japanese (PCK - PC Kanji Code) Feature Package specific files. It's a extened package to support PCK environment.
SUNWjpct	Japanese (PCK) Localizations for UTF-8 Code Conversion Tool
SUNWjpdas	Japanese localization for tools to synchronize desktop applications with the Palm Pilot PDA
SUNWjpdst	Japanese (PCK) Localization for CDE Desktop Applications
SUNWjpdte	Japanese (PCK) Localization for CDE Desktop Login Environment
SUNWjpezt	Japanese (PCK) Localizations for Desktop Power Pack Applications
SUNWjphe	Japanese (PCK) Localization for CDE Help Runtime environment
SUNWjphed	Japanese (PCK) Localization for CDE Help Developer Environment
SUNWjphev	Japanese (PCK) Localization for CDE Help Volumes
SUNWjphez	Japanese (PCK) Localizations for Desktop Power Pack Help Volumes
SUNWjpim	Japanese (PCK) Localization for Solaris CDE Image Viewer
SUNWjpj2m	Japanese (PCK) man pages
SUNWjpjmn	Japanese (PCK) JavaVM manual pages for Java programmers and users
SUNWjpkcs	Japanese (PCK) Localizations for Kodak Color Management System Runtime

TABLE C–5 Japanese *(continued)*

SUNWjpman	Japanese Feature Package Man Pages to see Japanese (PCK) manpages for SUNWjfpr and SUNWjfpu and Japanese manpages for SUNWman and SUNWaled.
SUNWjpmfr	Japanese (PCK) Localizations for Motif 1.2.3 RunTime Kit.
SUNWjppmm	Japanese (PCK) Power Management OW Utilities Man Pages
SUNWjppmw	Japanese (PCK) Localizations for Power Management OW Utilities
SUNWjprdm	Japanese (PCK) OILBN ReadMe Directory
SUNWjprme	Japanese (PCK) Localization for Common Desktop Environment (CDE) release documentation
SUNWjprvl	Japanese (PCK) Localizations for XIL loadable pipelines for SunVideo capture and compression
SUNWjpsal	Japanese (PCK) Localizations for Solstice Admintool launcher and associated libraries.
SUNWjptlm	Japanese (PCK) ToolTalk manual pages for ToolTalk programmers, OpenWindows users, and Common Desktop Environment (CDE) users
SUNWjptlt	Japanese (PCK) Localizations for ToolTalk binaries and shared libraries needed for Common Desktop Environment (CDE), OpenWindows, and all ToolTalk clients
SUNWjpudc	Japanese (PCK) Localizations for User Defined Character tool for Solaris CDE environment
SUNWjpwm	Japanese (PCK) Localization for CDE Desktop Window Manager
SUNWjpwnu	Japanese Input System - Wnn6 Messages, (PCK)
SUNWjpxfa	Japanese (PCK) Localizations for Font Administration application for Solaris platforms
SUNWjpxir	Japanese (PCK) Localizations for XIL Runtime Environment
SUNWjpxpm	Japanese (PCK) X Window System online programmers man pages
SUNWjpxum	Japanese (PCK) X Window System online user man pages
SUNWjrdm	Japanese (EUC) OILBN ReadMe Directory
SUNWjreg	Japanese Localizations for Solaris User Registration
SUNWjsadl	Japanese (EUC) Localizations for Solstice Admintool launcher and associated libraries.
SUNWjtlmn	Japanese (EUC) ToolTalk manual pages for ToolTalk programmers, OpenWindows users, and Common Desktop Environment (CDE) users

TABLE C–5 Japanese *(continued)*

SUNWjtltk	Japanese (EUC) Localizations for ToolTalk binaries and shared libraries needed for Common Desktop Environment (CDE), OpenWindows, and all ToolTalk clients
SUNWju8e	Japanese (UTF-8) Feature Package specific files. It's a extened package to support Japanese UTF-8 environment.
SUNWjuab	Japanese (UTF-8) Localization for CDE Desktop Application Builder
SUNWjuacx	Japanese (UTF-8) Localizations for AccessX client program
SUNWjuadi	Japanese (UTF-8) Localizations for admintool and GUI install.
SUNWjuadm	Japanese (UTF-8) Localizations for Software used to perform system administration tasks. Admintool requires both this and `SUNWjuadi` packages for Japanese (UTF-8) localization.
SUNWjubas	Japanese (UTF-8) Localization for CDE application basic runtime environment
SUNWjuct	Japanese (UTF-8) Localizations for UTF-8 Code Conversion Tool
SUNWjudst	Japanese (UTF-8) Localization for CDE Desktop Applications
SUNWjudte	Japanese (UTF-8) Localization for CDE Desktop Login Environment
SUNWjuezt	Japanese (UTF-8) Localizations for Desktop Power Pack Applications
SUNWjuhe	Japanese (UTF-8) Localization for CDE Help Runtime environment
SUNWjuhed	Japanese (UTF-8) Localization for CDE Help Developer Environment
SUNWjuhev	Japanese (UTF-8) Localization for CDE Help Volumes
SUNWjuhez	Japanese (UTF-8) Localizations for Desktop Power Pack Help Volumes
SUNWjuim	Japanese (UTF-8) Localization for Solaris CDE Image Viewer
SUNWjuj2m	Japanese (UTF-8) man pages
SUNWjujmn	Japanese (UTF-8) JavaVM Manual pages for Java programmers and users
SUNWjukcs	Japanese (UTF-8) Localizations for Kodak Color Management System Runtime
SUNWjulcf	Japanese (UTF-8) Localizations for xutops command
SUNWjuman	Japanese Feature Package Man Pages to see Japanese (UTF-8) manpages for SUNWjfpr and SUNWjfpu and Japanese manpages for SUNWman and SUNWaled.
SUNWjumfr	Japanese (UTF-8) Localizations for Motif 1.2.3 RunTime Kit.

TABLE C–5 Japanese *(continued)*

SUNWjupmm	Japanese (UTF-8) Power Management OW Utilities Man Pages
SUNWjupmw	Japanese (UTF-8) Localizations for Power Management OW Utilities
SUNWjurdm	Japanese (UTF-8) OILBN ReadMe Directory
SUNWjurme	Japanese (UTF-8) Localization for Common Desktop Environment (CDE) release documentation
SUNWjurvl	Japanese (UTF-8) Localizations for XIL loadable pipelines for SunVideo capture and compression
SUNWjusal	Japanese (UTF-8) Localizations for Solstice Admintool launcher and associated libraries.
SUNWjutlm	Japanese (UTF-8) ToolTalk manual pages for ToolTalk programmers, OpenWindows users, and Common Desktop Environment (CDE) users
SUNWjutlt	Japanese (UTF-8) Localizations for ToolTalk binaries and shared libraries needed for Common Desktop Environment (CDE), OpenWindows, and all ToolTalk clients
SUNWjuudc	Japanese (UTF-8) Localizations for User Defined Character tool for Solaris CDE environment
SUNWjuwm	Japanese (UTF-8) Localization for CDE Desktop Window Manager
SUNWjuwnu	Japanese Input System - Wnn6 Messages, (UTF-8)
SUNWjuxfa	Japanese (UTF-8) Localizations for Font Administration application for Solaris platforms
SUNWjuxir	Japanese (UTF-8) Localizations for XIL Runtime Environment
SUNWjuxpm	Japanese (UTF-8) X Window System online programmers man pages
SUNWjuxum	Japanese (UTF-8) X Window System online user man pages
SUNWjwacx	Japanese (EUC) Localizations for AccessX client program
SUNWjwbcp	Japanese (EUC) Localizations for Support files, programs, and libraries for Openwindows Binary Compatibility.
SUNWjwbk	Japanese (EUC) Localizations for OpenWindows online handbooks
SUNWjwncr	Japanese Input System - Wnn6 Client, (Root)
SUNWjwncu	Japanese Input System - Wnn6 Client, (Usr)
SUNWjwncx	Japanese Input System - Wnn6 Client X Window System
SUNWjwndt	Japanese Input System - Wnn6 Client for CDE

TABLE C–5 Japanese *(continued)*

SUNWjwnsr	Japanese Input System - Wnn6 Server, (Root)
SUNWjwnsu	Japanese Input System - Wnn6 Server, (Usr)
SUNWjwsr	Japanese Solaris Product Registry
SUNWjxfa	Japanese (Common) Localizations for Font Administration application for Solaris platforms
SUNWjxfnt	Japanese X Window System Fonts (required fonts) - gothic bold fonts and TrueType map files
SUNWjxoft	Sun Minchou bitmap fonts
SUNWjxplt	Japanese Localizations for X Window System platform software (Extensions)
SUNWjxpmn	Japanese (EUC) X Window System online programmers man pages
SUNWjxumn	Japanese (EUC) X Window System online user man pages

TABLE C–6 Korean

NSCPkocom	Korean localization of Netscape Communicator 4.7 supporting International security.
NSCPkucom	ko.UTF-8 localization of Netscape Communicator 4.7 supporting International security.
SUNWkadis	Korean (EUC) Localizations for admintool and GUI install.
SUNWkadma	Korean (EUC) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNWkadis packages for Korean (EUC) localization.
SUNWkbcp	This package contains Korean Language Environment binary compatibility files.
SUNWkcoft	Korean/Korean UTF-8 common optional font package
SUNWkdab	Korean Localizations for CDE Desktop Application Builder
SUNWkdbas	Korean Localizations for CDE Base functionality
SUNWkdcst	The localized tools package for Korean.
SUNWkddst	Korean Localizations for CDE Desktop Applications
SUNWkddte	Korean Localizations for CDE Desktop Login Environment

TABLE C–6 Korean *(continued)*

SUNWkdezt	Korean (EUC) Localizations for Desktop Power Pack Applications
SUNWkdft	Fonts for the common desktop environment, Korean L10N CDE
SUNWkdhe	Korean Localizations for CDE Help Runtime environment
SUNWkdhev	Korean CDE Help Volumes
SUNWkdhez	Korean (Common) Localizations for Desktop Power Pack Help Volumes
SUNWkdicn	Korean Localizations for CDE Icons
SUNWkdim	Korean Localizations for CDE Imagetool
SUNWkdwm	Korean Localizations for CDE Desktop Window Manager
SUNWkepmw	Korean (EUC) Localization for Power Management OW Utilities
SUNWkervl	Korean (EUC) SunVideo Runtime Support Software
SUNWkexir	Korean (EUC) XIL Runtime Environment
SUNWkj2rt	Java virtual machine and core class libraries (Korean supplement)
SUNWkjvdv	Korean Localizations for JavaVM developers package
SUNWkjvrt	Korean Localizations for JavaVM run time environment
SUNWkkcsr	Korean (EUC) KCMS Runtime Environment
SUNWkler	This package contains the stream modules for Korean Language Environment. It is a required package to run Korean Language Environment
SUNWklerx	Stream modules for Korean Language Environment. It is a required package to run Korean Language Environment (64-bit)
SUNWkleue	This package contains Korean Language Environment specific files. It is a required package to run Korean Language Environment
SUNWkoaud	Korean OPENLOOK Audio Applications Package
SUNWkodcv	Korean OPENLOOK Document and Help Viewer Applications Package
SUNWkodem	Korean OPENLOOK Demo Programs Package
SUNWkodst	Korean OPENLOOK Deskset Tools Package
SUNWkodte	Korean Core OPENLOOK Desktop Package
SUNWkoimt	Korean OPENLOOK Imagetool Package
SUNWkoj2p	Korean localization of Java Plug-in 1.2.2

TABLE C–6 Korean *(continued)*

SUNWkoman	Korean OPENLOOK Toolkit / Desktop Users Man Pages Package
SUNWkorte	Korean OPENLOOK Toolkits Runtime Environment Package
SUNWkrdm	Korean (EUC) OILBN ReadMe Directory
SUNWkreg	Korean Localizations for Solaris User Registration
SUNWksadl	Korean (EUC) Localizationsfor Solstice Admintool launcher and associated libraries.
SUNWktltk	Korean ToolTalk Runtime Package Package
SUNWkttfe	Korean True Type Font Extension
SUNWkuadi	Korean (UTF-8) Localizations for admintool and GUI install.
SUNWkuadm	Korean (UTF-8) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNWkadis packages for Korean (EUC) localization.
SUNWkudab	Korean / UTF-8 Localizations for CDE Desktop Application Builder
SUNWkudbs	Korean / UTF-8 Localizations for CDE Base functionality
SUNWkudc	Korean (EUC) Localizations for User Defined Character tool for Solaris CDE environment
SUNWkudda	Korean / UTF-8 Localizations for CDE Desktop Applications
SUNWkuddt	Korean / UTF-8 Localizations for CDE Desktop Login Environment
SUNWkudft	Fonts for the common desktop environment, Korean / UTF-8 L10N CDE
SUNWkudhr	Korean / UTF-8 Localizations for CDE Help Runtime environment
SUNWkudhv	Korean / UTF-8 CDE Help Volumes
SUNWkudhz	Korean (Common) Localizations for Desktop Power Pack Help Volumes
SUNWkudic	Korean / UTF-8 Localizations for CDE Icons
SUNWkudim	Korean / UTF-8 Localizations for CDE Imagetool
SUNWkudwm	Korean / UTF-8 Localizations for CDE Desktop Window Manager
SUNWkudzt	Korean (UTF-8) Localizations for Desktop Power Pack Applications
SUNWkulee	This package contains Korean UTF-8 Language Environment specific files. It is a required package to run Korean Language Environment
SUNWkuodf	Korean UTF-8 Core OPENLOOK Desktop Package
SUNWkupmw	Korean UTF-8 Localization for Power Management OW Utilities

TABLE C–6 Korean *(continued)*

SUNWkurdm	Korean (UTF-8) OILBN ReadMe Directory
SUNWkusal	Korean (UTF-8) Localizationsfor Solstice Admintool launcher and associated libraries.
SUNWkuudc	Korean (UTF-8) Localizations for User Defined Character tool for Solaris CDE environment
SUNWkuxe	Korean UTF-8 X Windows Platform Software Package
SUNWkuxft	Korean UTF-8 X Windows Platform Required Fonts
SUNWkwbcp	Korean OpenWindows Binary Compatibility Package
SUNWkwsr	Korean prodreg 2.0 localizable text resources
SUNWkxe	Korean X Windows Platform Software Package
SUNWkxfte	Korean X Windows Platform Required Fonts
SUNWkxman	Korean X Windows Online User Man Pages Package

TABLE C–7 Spanish

NSCPescom	Spanish localization of Netscape Communicator 4.7 supporting International security.
SUNWe8bas	Base L10N fr 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 apps
SUNWe8wm	Spanish UTF-8 CDE Desktop Window Manages Messages
SUNWeoaud	Spanish OPEN LOOK (R) Audio applications
SUNWeobk	Spanish OpenWindows online handbooks
SUNWeodcv	Spanish OPEN LOOK (R) document and help viewer applications
SUNWeodem	Spanish OPEN LOOK (R) demo programs
SUNWeodst	Spanish OPEN LOOK (R) deskset tools

TABLE C–7 Spanish *(continued)*

SUNWeodte	Spanish OPEN LOOK (R) desktop environment
SUNWeoimt	Spanish OPEN LOOK (R) imagetool
SUNWeorte	Spanish OPEN LOOK (R) toolkits runtime environment
SUNWesbas	Base L10N fr CDE functionality to run a CDE application
SUNWesdst	CDE Desktop Applications
SUNWesdte	CDE Desktop Login Environment
SUNWeshe	CDE Help L10N es Runtime Environment
SUNWeshed	CDE L10N es Help Developer Environment
SUNWeshev	CDE Help Volumes
SUNWesim	CDE Desktop apps
SUNWesj2p	Spanish localization of Java Plug-in 1.2.2
SUNWesos	localizable message files for the OS-Networking consolidation
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
SUNWewacx	Spanish OPEN LOOK (R) AccessX
SUNWexplt	Spanish X Windows platform software

TABLE C–8 Swedish

NSCPsvcom	Swedish localization of Netscape Communicator 4.7 supporting International security.
SUNWs8bas	Base Swedish UTF-8 CDE functionality messages
SUNWs8dst	Swedish UTF-8 CDE Desktop Applications messages
SUNWs8dte	Swedish UTF-8 CDE Desktop Login Environment messages
SUNWs8he	Swedish UTF-8 CDE Help Runtime Environment

TABLE C–8 Swedish *(continued)*

SUNWs8im	Swedish UTF-8 CDE Image editor messages
SUNWs8wm	Swedish UTF-8 CDE Desktop Window Manages Messages
SUNWsoaud	Swedish OPEN LOOK (R) Audio applications
SUNWsobk	Swedish OpenWindows online handbooks
SUNWsodcv	Swedish OPEN LOOK (R) document and help viewer applications
SUNWsodem	Swedish OPEN LOOK (R) demo programs
SUNWsodst	Swedish OPEN LOOK (R) deskset tools
SUNWsodte	Swedish OPEN LOOK (R) desktop environment
SUNWsoimt	Swedish OPEN LOOK (R) imagetool
SUNWsorte	Swedish OPEN LOOK (R) 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 Login Environment messages
SUNWsvhe	Swedish CDE Help Runtime Environment
SUNWsvhed	Swedish CDE Help Developer Environment messages
SUNWsvhev	CDE Help Volumes
SUNWsvim	Swedish CDE Image editor messages
SUNWsvj2p	Swedish localization of Java Plug-in 1.2.2
SUNWsvos	localizable message files for the OS-Networking consolidation
SUNWsvpmw	Swedish (EUC) Localizations for Power Management OW Utilities
SUNWsvreg	Solaris User Registration prompts at desktop login for user registration
SUNWsvwm	Swedish CDE Desktop Window Manages Messages
SUNWswacx	Swedish OPEN LOOK (R) AccessX
SUNWsxplt	Swedish X Windows platform software
SUNWvbcp	Swedish OS Binary Compatibility Package
SUNWvwbcp	Swedish OpenWindows Binary Compatibility Package

TABLE C–9 Traditional Chinese

NSCP5com	zh_TW.BIG5 localization of Netscape Communicator 4.7 supporting International security.
NSCPhcom	Traditional Chinese localization of Netscape Communicator 4.7 supporting International security.
NSCPhucom	zh_TW.UTF-8 localization of Netscape Communicator 4.7 supporting International security.
SUNW5adi	Traditional Chinese Localizations for admintool and GUI install.
SUNW5adma	Traditional Chinese Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNW5adi packages for Localization.
SUNW5dab	Traditional Chinese Localizations for CDE Desktop Application Builder
SUNW5dbas	Traditional Chinese Localizations for CDE Base functionality
SUNW5ddst	Traditional Chinese Localizations for CDE Desktop Applications
SUNW5ddte	Traditional Chinese Localizations for CDE Desktop Login Environment
SUNW5dezt	Traditional Chinese (BIG5) Localizations for Desktop Power Pack Applications
SUNW5dft	Traditional Chinese Localizations for CDE Fonts
SUNW5dhe	Traditional Chinese Localizations for CDE Help Runtime environment
SUNW5dhev	Traditional Chinese CDE Help Volumes
SUNW5dhez	Traditional Chinese (Common BIG5) Localizations for Desktop Power Pack Help Volumes
SUNW5dicn	Traditional Chinese Localizations for CDE Icons
SUNW5dim	Traditional Chinese Localizations for CDE Imagetool
SUNW5dwm	Traditional Chinese Localizations for CDE Desktop Window Manager
SUNW5leue	This package contains Traditional Chinese Language Environment specific files. It is a required package to run Traditional Chinese BIG5 Language Environment.
SUNW5odte	Traditional Chinese BIG5 Core OPENLOOK Desktop Package
SUNW5pmw	Traditional Chinese BIG5 Localization for Power Management OW Utilities
SUNW5rdm	Taiwanese (BIG5) OILBN ReadMe Directory

TABLE C–9 Traditional Chinese *(continued)*

SUNW5sadl	Traditional Chinese Localizations for Solstice Admintool launcher and associated libraries.
SUNW5ttfe	Traditional Chinese True Type Fonts Package Extension
SUNW5udc	Traditional Chinese (BIG5) Localizations for User Defined Character tool for Solaris CDE environment
SUNW5xfnt	Traditional Chinese BIG5 X Windows Platform required Fonts Package
SUNWhadis	Traditional Chinese (EUC) Localizations for admintool and GUI install.
SUNWhadma	Traditional Chinese (EUC) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNWhadis packages for Traditional Chinese (EUC) localization.
SUNWhbcp	This package contains Traditional Chinese Language Environment binary compatibility files.
SUNWhdab	Traditional Chinese Localizations for CDE Desktop Application Builder
SUNWhdbas	Traditional Chinese Localizations for CDE Base functionality
SUNWhddst	Traditional Chinese Localizations for CDE Desktop Applications
SUNWhddte	Traditional Chinese Localizations for CDE Desktop Login Environment
SUNWhdezt	Traditional Chinese (EUC) Localizations for Desktop Power Pack Applications
SUNWhdft	Traditional Chinese Localizations for CDE Fonts
SUNWhdhe	Traditional Chinese Localizations for CDE Help Runtime environment
SUNWhdhev	Traditional Chinese CDE Help Volumes
SUNWhdhez	Traditional Chinese (Common) Localizations for Desktop Power Pack Help Volumes
SUNWhdicn	Traditional Chinese Localizations for CDE Icons
SUNWhdim	Traditional Chinese Localizations for CDE Imagetool
SUNWhdwm	Traditional Chinese Localizations for CDE Desktop Window Manager
SUNWhepmw	Traditional Chinese (EUC) Localization for Power Management OW Utilities
SUNWhervl	Traditional Chinese (EUC) SunVideo Runtime Support Software
SUNWhexir	Traditional Chinese (EUC) XIL Runtime Environment
SUNWhj2p	Traditional Chinese localization of Java Plug-in 1.2.2

TABLE C–9 Traditional Chinese *(continued)*

SUNWhj2rt	Java virtual machine and core class libraries (Traditional Chinese supplement)
SUNWhjvdv	Traditional Chinese Localizations for JavaVM developers package
SUNWhjvrt	Traditional Chinese Localizations for JavaVM run time environment
SUNWhkcsr	Traditional Chinese (EUC) KCMS Runtime Environment
SUNWhler	This package contains the stream modules for Traditional Chinese Language Environment. It is a required package to run Traditional Chinese Language Environment.
SUNWhlerx	Stream modules for Traditional Chinese Language Environment. It is a required package to run Traditional Chinese Language Environment (64-bit).
SUNWhleue	This package contains Traditional Chinese Language Environment specific files. It is a required package to run Traditional Chinese Language Environment.
SUNWhoaud	Traditional Chinese OPENLOOK Audio Applications Package
SUNWhodcv	Traditional Chinese OPENLOOK Document and Help Viewer Applications Package
SUNWhodem	Traditional Chinese OPENLOOK Demo Programs Package
SUNWhodst	Traditional Chinese OPENLOOK Deskset Tools Package
SUNWhodte	Traditional Chinese Core OPENLOOK Desktop Package
SUNWhoimt	Traditional Chinese OPENLOOK Imagetool Package
SUNWhoman	Traditional Chinese OPENLOOK Toolkit/Desktop Users Man Pages Package
SUNWhorte	Traditional Chinese OPENLOOK Toolkits Runtime Environment Package
SUNWhrdm	Taiwanese (EUC) OILBN ReadMe Directory
SUNWhreg	Traditional Chinese Localizations for Solaris User Registration
SUNWhsadl	Traditional Chinese (EUC) Localizationsfor Solstice Admintool launcher and associated libraries.
SUNWhtltk	Traditional Chinese ToolTalk Runtime Package Package
SUNWhttfe	Traditional Chinese True Type optional Fonts Package Extension
SUNWhuada	Traditional Chinese (UTF-8) Localizations for Software used to perform system administration tasks. Admintool requires both this and SUNW5adi packages for Localization.

TABLE C–9 Traditional Chinese *(continued)*

SUNWhuadi	Traditional Chinese (UTF-8) Localizations for admintool and GUI install.
SUNWhubas	Traditional Chinese (UTF-8) Localizations for CDE Base functionality
SUNWhuccd	This package contains Traditional Chinese Console Display Environment specific files. It is a required package to run Traditional Chinese Console Display Environment
SUNWhudab	Traditional Chinese (UTF-8) Localizations for CDE Desktop Application Builder
SUNWhudc	Traditional Chinese (EUC) Localizations for User Defined Character tool for Solaris CDE environment
SUNWhudez	Traditional Chinese (UTF-8) Localizations for Desktop Power Pack Applications
SUNWhudft	Traditional Chinese (UTF-8) Localizations for CDE Fonts
SUNWhudhe	Traditional Chinese (UTF-8) Localizations for CDE Help Runtime environment
SUNWhudhv	Traditional Chinese (UTF-8) CDE Help Volumes
SUNWhudhz	Traditional Chinese (Common UTF-8) Localizations for Desktop Power Pack Help Volumes
SUNWhudic	Traditional Chinese (UTF-8) Localizations for CDE Icons
SUNWhudim	Traditional Chinese (UTF-8) Localizations for CDE Imagetool
SUNWhudst	Traditional Chinese (UTF-8) Localizations for CDE Desktop Applications
SUNWhudte	Traditional Chinese (UTF-8) Localizations for CDE Desktop Login Environment
SUNWhudwm	Traditional Chinese (UTF-8) Localizations for CDE Desktop Window Manager
SUNWhulee	This package contains Traditional Chinese (UTF-8) Language Environment specific files. It is a required package to run Traditional Chinese UTF-8 Language Environment.
SUNWhuodt	Traditional Chinese UTF-8 Core OPENLOOK Desktop Package
SUNWhupmw	Traditional Chinese UTF-8 Localization for Power Management OW Utilities
SUNWhurdm	Taiwanese (UTF-8) OILBN ReadMe Directory
SUNWhusad	Traditional Chinese (UTF-8) Localizations for Solstice Admintool launcher and associated libraries.

TABLE C–9 Traditional Chinese *(continued)*

SUNWhuudc	Traditional Chinese (UTF-8) Localizations for User Defined Character tool for Solaris CDE environment
SUNWhwbcp	Traditional Chinese OpenWindows Binary Compatibility Package
SUNWhwsr	Traditional Chinese prodreg 2.0 localizable text resources
SUNWhxe	Traditional Chinese X Windows Platform Software Package
SUNWhxman	Traditional Chinese X Windows Online User Man Pages Package

TABLE C–10 Shared

SUNWabcp	Asian common files for SunOS 4.x Binary Compatibility
SUNWerdm	OILBN ReadMe Directory
SUNWudct	User Defined Character tool for Solaris CDE environment

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W

X

Y

Z