



Asian-Language Support in the Solaris Operating Environment

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

Part Number 806-5582
May 2000

Copyright 2000 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. All rights reserved.

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

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

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

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

Federal Acquisitions: Commercial Software—Government Users Subject to Standard License Terms and Conditions.

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

Copyright 2000 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, Californie 94303-4900 Etats-Unis. Tous droits réservés.

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

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

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

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

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



Contents

	Preface	5
1.	Solaris Operating Environment in International Markets	9
1.1	The Need for Global Software Development	9
1.2	Software Internationalization	9
1.3	Benefits of Internationalized Software	11
2.	Internationalized Software for the Solaris Operating Environment	13
2.1	Solaris Language-Support Framework	13
2.2	Locale	14
2.3	Interface Localization	14
2.4	Codeset Independence	14
3.	Asian Language Overview	17
3.1	Chinese	17
3.2	Japanese	18
3.3	Korean	19
4.	Technical Considerations	21
4.1	Asian-Specific Architecture	21
4.1.1	Input Methods	22
4.1.2	Character Conversion	23
4.1.3	Input-Method Server	24

4.1.4	Font Editor	25
4.1.5	User-Defined Character Tool	25
5.	Common Development Issues	27
5.1	Casing	27
5.2	Sort Order	27
5.3	Text Manipulation	28
5.4	Fonts	29
A.	Product Overview	31
A.1	Common Desktop Environment (CDE) Deskset	31
A.2	Simplified Chinese Solaris 8 Operating Environment Features	33
A.3	Traditional Chinese Solaris 8 Operating Environment Features	35
A.4	Japanese Solaris 8 Operating Environment Features	37
A.5	Korean Solaris 8 Operating Environment Features	39
A.5.1	Korean Dictionary Tools	41

Preface

The *Asian-Language Support in the Solaris™ Operating Environment* white paper presents information and software features for internationalizing software in Asian-language markets.

Who Should Use This Book

This white paper is intended for software developers who are interested in developing internationalized software for the Asian-language Solaris™ operating environment. This white paper is part of a 4-part series on internationalization for Solaris software developers. The four internationalization white papers are:

- *Asian-Language Support in the Solaris™ Operating Environment*
- *Complex Text Layout Language Support in the Solaris™ Operating Environment*
- *Unicode Support in the Solaris™ Operating Environment*
- *Euro Currency Support in the Solaris™ Operating Environment*

How This Book Is Organized

Chapter 1 provides an overview of internationalization.

Chapter 2 provides an overview of Solaris internationalization.

Chapter 3 describes internationalization details of Asian languages, specifically Chinese, Japanese, and Korean.

Chapter 4 addresses the technical concerns of the Asian-specific architecture.

Chapter 5 describes common development issues in multibyte applications.

Appendix A shows features for the Common Desktop Environment, Simplified Chinese, Traditional Chinese, Japanese, and Korean Solaris operating environment.

Related Books

The following books are related to software internationalization:

- *Creating Worldwide Software: Solaris International Developer's Guide* Bill Tuthill and David Smallberg.
- *Internationalization Guide, Version 2: Open Group Guide* The Open Group
- *International Language Environments Guide* Solaris Developer Collection.
- *Programming for the World: A Guide to Internationalization* Sandra Martin O'Donnell.
- *The Unicode Standard, Version 3.0* The Unicode Consortium.
- *X Windows on the World, Developing Internationalized Software with X, Motif, and CDE* Thomas C. McFarland.
- The following book is related to Asian-language internationalization:
- *CJKV Information Processing: Chinese, Japanese, Korean, and Vietnamese Computing* Ken Lunde

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.comSM 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`.

Solaris Operating Environment in International Markets

1.1 The Need for Global Software Development

In an integrated global economy, software applications must be compatible in numerous languages and cultures. Users want to run applications in their own language, using their own local conventions. Furthermore, international companies have international needs. For example, a large corporation with headquarters in Tokyo and branches in New York and Paris may require a mixture of English, Japanese, and French software environments supporting multiple languages on one site.

To sell software to multinational companies, developers must always be aware of local customs, conventions, and requirements during development, such as character sets, numeric, time, date, and monetary formats, and messages. Adapting software to localized writing systems is particularly challenging in Asian markets. 8-bit encoding is good enough for European phonetic alphabets, but Chinese, Japanese, and Korean ideographs require multibyte encoding.

1.2 Software Internationalization

Internationalized software applications include internationalized code and localized locale-specific data.

Internationalization *generalizes* software by using a single internationalized binary which retrieves locale-specific data and shared objects at run time. The application runs on any localized version of the Solaris operating environment, without requiring source code changes or recompilation.

Localization *customizes* software data, providing locale-specific modules that meet local requirements. Localization can be either of the following:

- Full localization—input, output, print, cultural conventions, and translated message text.
- Partial localization—input, output, print, cultural conventions, without translated message text.

Developers generally create applications for the U.S. market. Internationalization is especially important in newer, smaller markets which don't yet justify full localization. Here, a phased approach is recommended, beginning with the current internationalized Solaris operating environment version, followed by localization as the market grows as illustrated in Figure 1-1.

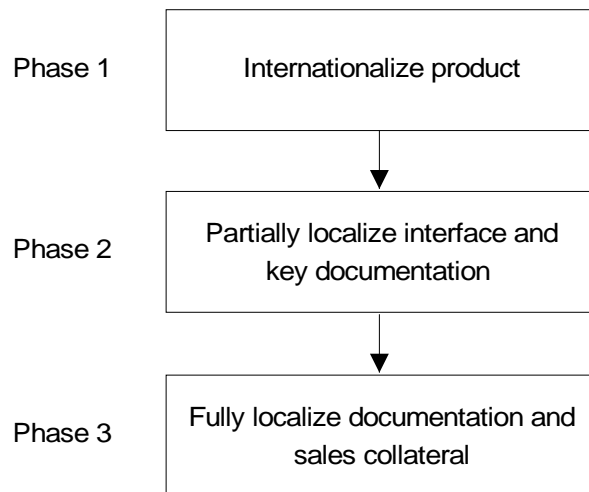


Figure 1-1 Market-entry strategy for localized products

Sun Microsystems requires that all applications be internationalized.

1.3 Benefits of Internationalized Software

Internationalization addresses many of the key software issues:

- Improving software quality
- Reducing development time and cost
- Enabling code reuse
- Reducing localization costs
- Reducing maintenance costs
- Increasing customer satisfaction

Using separate locale-specific files to localize applications is simpler, faster, and more cost effective on the whole. Software is more easily released world wide at less expense. As well, users are much happier working in their own language with their own conventions.

Using a single binary in an internationalized application ensures that the same feature set is available for a particular software version and lessens support, maintenance, and system-administration costs. Interoperability and productivity improves and common training materials can be used. Most importantly, internationalized and localized applications help developers compete and succeed in new foreign-language markets. Everyone benefits with internationalized software.

Internationalized Software for the Solaris Operating Environment

2.1 Solaris Language-Support Framework

In an internationalized application, language-specific features and cultural data are separated from application code. The Solaris internationalization framework divides code and language and cultural data into the following three areas:

- Locale
- Interface localization
- Codeset independence

A *locale* is a set of language and cultural variables, particular to a global region. The locale is selected by the user and loaded in memory at run time. The selected locale applies to the operating system and subsequent application launches.

Interface localization is the process of translating the interface language into another language by storing text strings and messages in a separate message file. Messages are more easily composed, translated, and referenced in a separate file than in hard-coded statements throughout the application. Furthermore, recompilation of the source binary is unnecessary.

Codeset independence does not assume a particular codeset to display and manipulate data.

2.2 Locale

The Solaris operating environment provides a number of locales. Each locale includes:

- Associated codeset and codeset conversion modules
- Numeric, time, and date formats
- Collation (sort order)
- Monetary format
- Interface information (messages and icons)
- Input method(s)
- Fonts

Developers access locale settings directly through Solaris operating environment APIs. For example, instead of encoding a particular currency symbol, an application calls the appropriate system API, which returns the currency symbol of the set locale.

2.3 Interface Localization

The Solaris operating environment supports several messaging schemes for localizing the interface, including the Sun proprietary API `gettext()` and the XPG `catgets()`. These APIs directly reference the message file.

Note that the size and position of interface elements (icons, graphics, and functions or private data affecting text elements) may be different in different languages. For example, Japanese messages are usually longer than English messages and Japanese ideographs are taller and wider than English characters. Text widget positioning should be relative, not absolute.

Icons and graphics should also be culturally neutral or be easily changeable to local tastes. Essentially, what a user sees and what affects text should be changed only in the message catalog, resources, or some other means.

2.4 Codeset Independence

The Solaris operating environment architecture supports codeset independence (CSI), expanding the number of supported codesets from Extended UNIX[®] Codeset (EUC)

to both EUC and non-EUC encodings, including PC-Kanji (also known as ShiftJIS) and GBK.

Note that text-handling routines should not define the size of the character codeset. Nor should other locale-specific components, such as the window system, input method, and online help, depend on a particular codeset. Figure 2-1 shows the locale-specific components which should be codeset independent.

Locale components	Text and Codesets	Collation and formats	Window system and GUI	Input method	Program manager	Online docs and Help
Generic components	Internationalized Solaris libraries and window system					

Figure 2-1 Design model for international software

Support for Unicode, a universal codeset encompassing most written characters, is often confused with codeset independence. Unicode is often referred to as ISO 10646 and is an International Standards Organization (ISO) standard. Note that codeset independence must also apply to Unicode. Although Unicode supports many languages and writing systems, to an application Unicode is just another codeset. The Solaris operating environment supports the Unicode UTF-8 (File System Safe UCS Transformation Format) format, which is compatible with ISO 10646. For more information, see *Unicode Support in the Solaris Operating Environment*.

Note - Codeset independence is often assumed because the idea of a character (in ISO C terms) and char (or byte) is thought of as a one-to-one relationship in programming languages. In written languages, however, the idea of a character can encompass one char/byte or multiple bytes. An alphabetic character from most European languages can be represented in one byte. An Asian-language character often requires more than one byte because there are more characters in the charset than one byte can represent.

Furthermore, applications often assume the representation of a given character. For example, a codeset independent application does not assume that 'a' = \x61 or char = byte. Instead, during text-manipulation routines, such as truncating a stream of characters, the APIs determine the size of the number of bytes by the character and its definition or type. By not assuming the size of a character or the codeset, the application will be codeset independent.

Solaris maintains a codeset independence framework. Applications can use Solaris APIs to determine the size of the number of bytes used by the character and its definition or type. By not making assumptions about the underlying codeset, an application is codeset independent in Solaris.

Asian Language Overview

A phonetic writing system, such as English, consists of a collection of phonetic letters to represent a word or idea. Asian languages, such as Chinese, Japanese, and Korean, however, use symbols or *ideographs* to represent words and ideas.

Chinese, Japanese, and Korean ideographs are all derived from the Chinese ideographic system, numbered in the tens of thousands. Collectively, the ideographs are called *han characters* and are referred to as *hanzi* in Chinese, *kanji* in Japanese, and *hanja* in Korean.

Note that an ideograph may be pronounced in several ways, depending on the context. As well, two different ideographs may be identically pronounced. The Solaris operating environment has been designed to include support for contextual ideographs in Asian-language writing systems.

3.1 Chinese

Two Chinese writing systems are used today—Traditional Chinese and Simplified Chinese. Their ideographs originated in China thousands of years ago.

Used in the Republic of China (Taiwan), Traditional Chinese has approximately 50,000 characters. Many of the older and more complex characters are still used today. Figure 3–1 shows Traditional Chinese characters representing the word "China."



Figure 3–1 Traditional Chinese character representing the word "China"

Used in the People's Republic of China (PRC), Simplified Chinese is a subset of the characters in Traditional Chinese. In 1955, the PRC government started eliminating and simplifying some ideographs by reducing the number of strokes needed to render a character. The Simplified Chinese character set is now simpler and smaller. Figure shows Simplified Chinese characters representing the word "China."

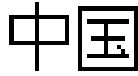


Figure 3-2 Simplified Chinese character representing the word "China"

3.2 Japanese

The Japanese language uses a combination of four different writing systems—*kanji* characters, *hiragana*, *katakana*, and the Roman alphabet phonetic system *romaji*.

Kanji characters are derived from Traditional Chinese characters and are often found in combination with hiragana, katakana, and romaji.

Hiragana is a set of 83 symbols, called a *syllabary*, that encompasses all the basic syllables used for Japanese pronunciation. In written Japanese, the hiragana syllabary expresses grammatical parts of speech, verb tenses, and some words for which there are no kanji characters or have become obsolete.

Katakana is another phonetic syllabary consisting of a different set of symbols for the same sounds expressed in hiragana. The syllables represented by hiragana and katakana are generically called *kana*. Figure 3-3 shows the differences between hiragana, katakana, and kanji characters.

Romaji is used to write Japanese sounds with Roman letters. Romaji characters are usually displayed in double-width format.

Kanji : 日本語
Hiragana : にほんご
Katakana : ニホンゴ

Figure 3-3 The differences between hiragana, katakana, and kanji characters

3.3 Korean

The Korean language uses a combination of two different writing systems—*hanja* characters and *hangul* characters.

Hanja characters are derived from Traditional Chinese characters and are often used for formal written communication and proper names. An example of hanja is shown in Figure 3-4.

The image shows the Korean word for 'Korea' written in Hanja (Traditional Chinese characters). The characters are 韓 (Han) and 國 (Guk), which together mean 'Korea'.

Figure 3-4 Hanja characters

Hangul characters are formed by combining one or more consonant and vowel signs from a syllabary consisting of 24 basic elements called *jamos*. There are approximately 11,000 hangul characters. An example of hangul is shown in Figure 3-5.

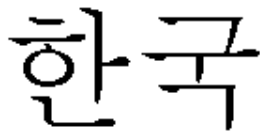
The image shows the Korean word for 'Korea' written in Hangul (Korean alphabet). The characters are 한 (Han) and 국 (Guk), which together mean 'Korea'.

Figure 3-5 Hangul characters

Technical Considerations

The large number of ideographs needed to support the Traditional Chinese, Simplified Chinese, Japanese, and Korean writing systems cannot be represented in one byte, and are often called double-byte or *multibyte* languages, depending on the platform architecture. The Solaris operating environment supports multibyte encoding, representing characters in one, two, or more bytes.

Separate software versions for multibyte locales need not be developed in the Solaris operating environment. However, there are issues unique to multibyte locale development—most importantly, that one character is not one byte in multibyte locales.

4.1 Asian-Specific Architecture

All localized versions of Solaris software are supersets of the U.S. English version and contain the same utilities and features. The difference between the U.S. English and a localized version is the addition of locale-specific data and tools facilitating input, display, and printing of local-language characters. All Asian versions of Solaris software include a locale database, user interface, and other locale-specific features. For example, Figure 4-1 shows how the locale database fits into the Japanese Solaris architecture.

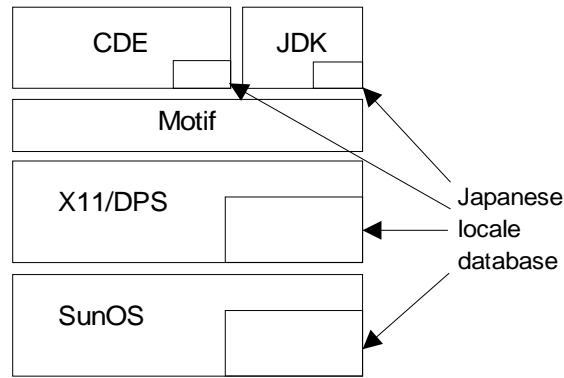


Figure 4-1 A Japanese Solaris architecture

Specific features were also added to the Traditional Chinese, Simplified Chinese, Japanese, and Korean localized versions of the Solaris operating environment to address the following issues:

- The thousands of characters used in everyday communication
- Ideographs with multiple meanings depending on context or pronunciation

4.1.1 Input Methods

How to enter thousands of characters is always an important issue in a multibyte language. Designing a keyboard with enough keys is simply not feasible. Instead, localized Solaris operating environments use *input methods*. Input methods (IMs) are system applications that convert keyboard input into a system-supported character. Figure 4-2 shows how an input method works.

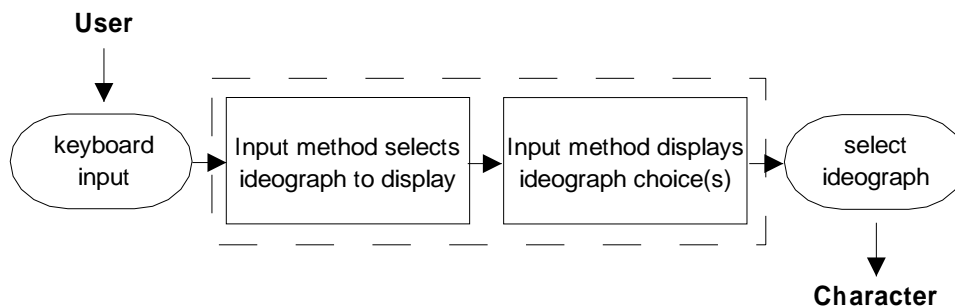


Figure 4-2 Input method

Generally, the Motif text widget manages the input method. However, to customize the input method or have direct control, call the X11 XIM (X Input Method) APIs.

Note - An application cannot assume a one-to-one mapping between a key-input stroke and a character. A single character may require more than a one key-input stroke and a one key-input event may trigger the input of more than one character.

4.1.2 Character Conversion

In Chinese, Japanese, and Korean, more than one ideograph can correspond to an input string. To avoid confusion, the Solaris operating environment uses a *Conversion Manager* to display the possible dictionary choices in a window as shown in Figure 4-3. The pre-edit, status, and lookup choice areas are highlighted for the sample Simplified Chinese input-method.

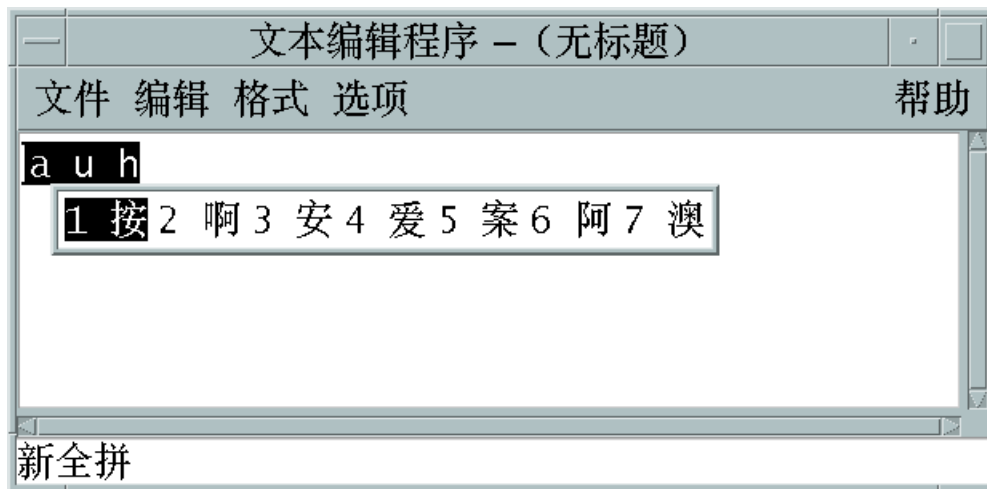


Figure 4-3 The pre-edit, status, and lookup choice areas

- Pre-edit area—displays characters as entered
- Status area—displays whether conversion is activated and the states or mode of the input method
- Lookup choice area—displays ideographic choices for the corresponding phonetic representation

Note - Input methods and associated dictionaries are often referred to as *language engines*.

Note - For more information on how to input Asian characters, see *Section 3.1* in the *Unicode Support in the Solaris Operating Environment*.

4.1.3 Input-Method Server

An input-method server (IM server) acts as the interface between input methods and applications as shown in Figure 4-4.

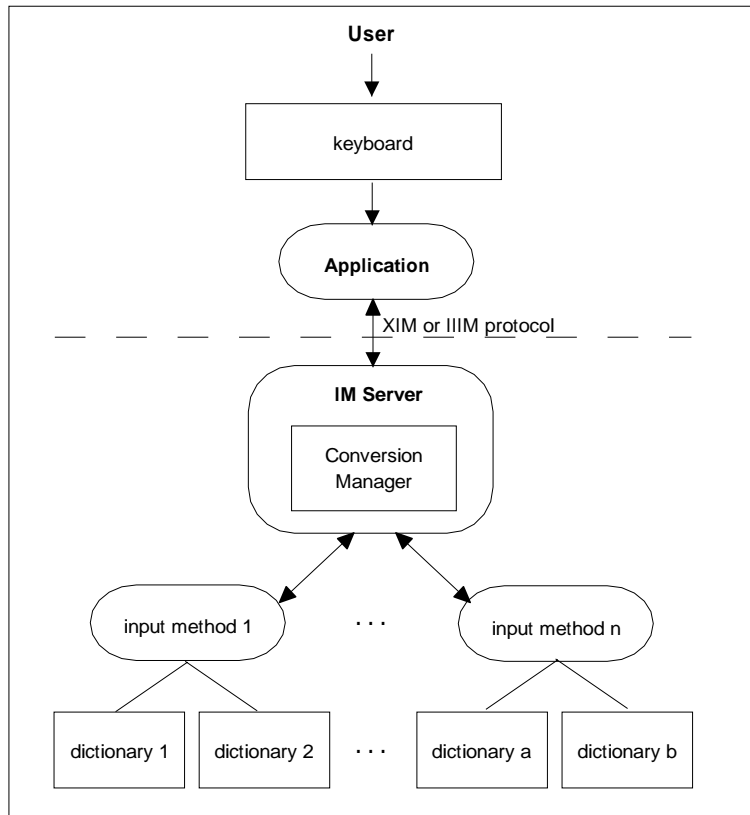


Figure 4-4 Input-method server

The IM server can support multiple language engines and provides user control over language-engine preferences, such as:

- Method of displaying status string when the portion of the string under consideration for conversion loses input focus.
- Number of rows and columns in the input conversion candidate pop-up window.

- Whether input conversion candidate selection window is displayed.

Many X toolkit-based applications automatically use the IM server for Asian text input. If you use any of Sun's toolkits (Motif, XViewTM, or OLIT), the input/output conversion process is transparent to the application.

4.1.4 Font Editor

The Font Editor is used to edit bitmap fonts. For example, a user may want to create a character not supported by the operating system because the repertoire of Han characters is too large. Using the Font Editor, new characters can be created and existing characters modified.

To start the Font Editor, type `fontedit` at the system prompt.

4.1.5 User-Defined Character Tool

The User-Defined Character Tool is used to create new characters as well as to specify font size for new characters. This utility can support both bitmap and Type 1 fonts.

To start the User-Defined Character Tool, type `sdtudctool` at the system prompt.

Common Development Issues

Writing and cultural conventions can vary greatly in different locales, such as character sets and numeric, time, date, and monetary formats. Some issues apply particularly to multibyte development.

5.1 Casing

Uppercase and lowercase words do not always apply in multibyte languages. For example, ideographs don't have case. Thus, characters that do not change after a casing function should not be treated as an error when calling an API which returns casing rules.

The following APIs process multibyte characters:

- `toupper()`: Convert wide characters to uppercase
- `tolower()`: Convert wide characters to lowercase
- `wctype()`: Define character class

5.2 Sort Order

Sorting conventions vary widely across languages and locales. Some languages even have different rules for collating the same character. Sorting ideographs is different than sorting phonetic scripts and is based on either the form or pronunciation of characters.

A form-based system sorts first on the character's primary radical and then on the number of strokes to write the character (stroke count). A pronunciation-based system sorts first on ideograph pronunciation and then on stroke count.

5.3 Text Manipulation

When supporting multibyte languages, it is important to understand the difference between multibyte, wide and Unicode characters, and the impact of these on software development.

In the Solaris operating environment, a multibyte character (or file code) is a sequence of one or more bytes terminated by a null string. Thus, a string may contain characters of different length. On the other hand, a wide character (or process code) is defined as a fixed-size number of bytes. In the Solaris operating environment, a wide character is defined to be four bytes long. The Solaris operating environment supports the Unicode UTF-8 format, a variable-length encoding similar to multibyte encoding.

In many cases, there is no need to distinguish double-byte (or three-byte) characters from single-byte characters. It is simpler to convert multibyte strings (file code) to wide-character formats (process code) before manipulating or processing text data.

The following APIs convert multibyte characters:

- `mbstowcs()`: Convert multibyte string to wide-character string
- `mbstowc()`: Convert multibyte to wide-character code

The following `wstring(3c)` APIs process multibyte characters:

- `wscmp()`: Compare wide-character strings
- `wscpy()`: Copy wide-character strings
- `wcslon()`: Get length of wide-character string
- `wcschr()`: Find character in wide-character string

Note - File code is in multibyte format. Process code is in wide-character format. Do not assume particular character encodings of the process code.

5.4 Fonts

Mixed codeset strings cannot usually be rendered with a single font. A font set is a collection of fonts suitable for rendering all codesets in a locale's encoding, and includes data about the locale in which it was created. For example, in the Korean locale, both the ASCII and Korean fonts are loaded. This is known as `FontSet` in the X11 Window System. The number of fonts and their character-set registry in a `FontSet` vary from one locale to another. Because the Solaris operating environment manages the `FontSet` at run time, applications do not need to know that multiple fonts are being used. You just need to use `FontSet` interfaces.

Common font family names, such as Times and Courier, are not usually available in multibyte locales. Locale-sensitive font family names should not be hard coded in applications. In the Common Desktop Environment, all locales have a common set of font alias names, such as `dt-application`.

Product Overview

The following tables show features for the Common Desktop Environment (CDE), Simplified Chinese, Traditional Chinese, Japanese and Korean Solaris 8 operating environment.

A.1 Common Desktop Environment (CDE) Deskset

Table A-1 lists the CDE Deskset tools and Table A-2 lists the Asian printing tools.

TABLE A-1 CDE Deskset Tools

DeskSet Tool	Features
Style Manager	Provides interactive customization of visual elements and system behavior for the desktop
Calendar	Enables group scheduling over the network, displays appointments and to-do items, sends automatic reminders using electronic mail
Mailer	Write, send, receive, and organize mail files including audio, image, and document files using simple drag-and-drop method
File Manager	Graphical way to navigate local and remote file systems: view, copy, or move files and documents, launch applications by point and click

TABLE A-1 CDE Deskset Tools *(continued)*

DeskSet Tool	Features
Printer	An intuitive interface to UNIX printing utilities
Tape Tool	Interface to UNIX tape archiving and retrieval utilities
Performance Meters	Allows system use to be monitored graphically
Audio Tool	An application to record, playback, and edit audio files
Image Tool	View files in popular graphic formats such as PostScript or TIFF
Text Editor	An interactive text editor with mouse-based graphical interface
Snapshot	Capture a black-and-white, grayscale, or color snapshot of the screen
Clock	Displays the current time in a window or icon for any time zone around the world
Icon Editor	A pixel editor that allows creation of customized icon images
Command/Shell Tool	Standard UNIX shell that accepts SunOSTM system software commands
sdtudctool	Tool for registering user-defined characters

TABLE A-2 Asian Printing Tools

Asian Printing Tools	Features
xetops	Used in EUC/CSI locales to print Asian text files: <code>xetops Asian_Text_File lp</code>
mp	Used in UTF-8 locales to print all UTF-8 characters: <code>mp UTF-8_File lp</code>

For more information, see the appropriate `man` pages.

A.2 Simplified Chinese Solaris 8 Operating Environment Features

Table A-3 to Table A-6 give an overview of the Simplified Chinese Solaris 8 operating environment features.

TABLE A-3 Simplified Chinese Codesets

Locale Name	Description	Supported Character Set
zh	Simplified Chinese (EUC)	GB 2312-1980
zh.GBK	Simplified Chinese (GBK)	GBK
zh.UTF-8	Simplified Chinese (UTF-8)	Unicode 3.0

TABLE A-4 Simplified Chinese Input Methods

Locale Name	Input Methods
zh, zh.GBK, zh.UTF-8	New QuanPin
zh, zh.GBK, zh.UTF-8	New ShuangPin
zh, zh.GBK, zh.UTF-8	Quanpy
zh	PinYin
zh	Stroke
zh	Golden
zh	Intelligent PinYin
zh	Simplified Chinese Symbol
zh.GBK, zh.UTF-8	GBK Code
zh.GBK, zh.UTF-8	Japanese
zh.GBK, zh.UTF-8	Hanja

TABLE A-4 Simplified Chinese Input Methods *(continued)*

Locale Name	Input Methods
zh.GBK, zh.UTF-8	Zhuyin
zh.UTF-8	Unicode Hex and Unicode Octal

TABLE A-5 Simplified Chinese Fonts

Locale Name	Full Family Name	Format
zh, zh.GBK	Fangson	TrueType
zh, zh.GBK	Hei	TrueType
zh, zh.GBK	Kai	TrueType
zh, zh.GBK	Song	TrueType
zh, zh.GBK	Song	PCF (12,14,16,20,24)

TABLE A-6 Simplified Chinese Codeset Conversions

Locale Name	Codeset Conversion Supported
zh, zh.GBK, zh.UTF-8	GB2312-80
zh, zh.GBK, zh.UTF-8	ISO-2022-7
zh, zh.GBK, zh.UTF-8	ISO-2022-CN
zh, zh.GBK, zh.UTF-8	UTF-8
zh, zh.GBK, zh.UTF-8	GBK
zh, zh.GBK, zh.UTF-8	BIG5
zh, zh.GBK, zh.UTF-8	HZ-GB-2312

A.3 Traditional Chinese Solaris 8 Operating Environment Features

Table A-7 to Table A-10 give an overview of the Traditional Chinese Solaris 8 operating environment features.

TABLE A-7 Traditional Chinese Codesets

Locale Name	Description	Supported Character Set
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

TABLE A-8 Traditional Chinese Input Methods

Locale Name	Input Methods
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	Chuyin
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	I-Tien
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	Telecode
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	TsangChieh
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	CheinI
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	NeiMA
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	ChuangHsing

TABLE A-8 Traditional Chinese Input Methods *(continued)*

Locale Name	Input Methods
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	Array
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	BoShiaMy
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	DaYi
zh_TW.UTF-8	Unicode Hex and Unicode Octal

TABLE A-9 Traditional Chinese Fonts

Locale Name	Full Family Name	Format
zh_TW, zh_TW.BIG5	Hei	TrueType
zh_TW, zh_TW.BIG5	Kai	TrueType
zh_TW, zh_TW.BIG5	Ming	TrueType
zh_TW, zh_TW.BIG5	Ming	PCF (12,14,16,20,24)

TABLE A-10 Traditional Chinese Codeset Conversions

Locale Name	Codeset Conversion Supported
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	CNS 11643
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	BIG5
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	ISO-2022-7
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	ISO-2022-CN-EXT

TABLE A-10 Traditional Chinese Codeset Conversions *(continued)*

Locale Name	Codeset Conversion Supported
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	UTF-8
zh_TW, zh_TW.BIG5, zh_TW.UTF-8	BIG5 Plus

A.4 Japanese Solaris 8 Operating Environment Features

Table A-11 to Table A-14 give an overview of the Japanese Solaris 8 operating environment features.

TABLE A-11 Japanese Codesets

Locale Name	Description	Supported Character Set
ja	Japanese (EUC)	JIS x 0201, JIS x 0208, JIS x 0212, UDC, VDC
ja_JP.PCK	Japanese (PCK)	JIS x 0201, JIS x 0208, UDC, VDC
ja_JP.UTF-8	Japanese (UTF-8)	Unicode 3.0

TABLE A-12 Japanese Input Methods

Locale Name	Input Methods
ja, ja_JP.PCK, ja_JP.UTF-8	ATOK12
ja, ja_JP.PCK, ja_JP.UTF-8	Wnn6
ja, ja_JP.PCK, ja_JP.UTF-8	cs00

TABLE A-12 Japanese Input Methods *(continued)*

Locale Name	Input Methods
ja, ja_JP.PCK, ja_JP.UTF-8	ATOK8
ja_JP.UTF-8	Unicode Hex and Unicode Octal

TABLE A-13 Japanese Fonts

Full Family Name	Format
hg gothic b	True Type
hg mincho 1	True Type
heiseimin	True Type
gothic	PCF (12,14,16,20,24)
minchou	PCF (12,14,16,20,24)
hg gothic b	PCF (12,14,16,20,24)
hg minchou 1	PCF (12,14,16,20,24)
heiseimin	PCF (12,14,16,20,24)

TABLE A-14 Japanese Codeset Conversions

Locale Name	Codeset Conversion Supported
ja, ja_JP.PCK, ja_JP.UTF-8	eucJP
ja, ja_JP.PCK, ja_JP.UTF-8	SJIS
ja, ja_JP.PCK, ja_JP.UTF-8	PCK
ja, ja_JP.PCK, ja_JP.UTF-8	ISO-2022-JP
ja, ja_JP.PCK, ja_JP.UTF-8	UTF-8
ja, ja_JP.PCK, ja_JP.UTF-8	JIS7

TABLE A-14 Japanese Codeset Conversions *(continued)*

Locale Name	Codeset Conversion Supported
ja, ja_JP.PCK, ja_JP.UTF-8	jis
ja, ja_JP.PCK, ja_JP.UTF-8	ibmj
ja, ja_JP.PCK, ja_JP.UTF-8	UTF-8-Java
ja, ja_JP.PCK, ja_JP.UTF-8	ibmj-EBCDIK
ja, ja_JP.PCK, ja_JP.UTF-8	ibm-930
ja, ja_JP.PCK, ja_JP.UTF-8	ibm-931
ja, ja_JP.PCK, ja_JP.UTF-8	ibm-939
ja, ja_JP.PCK, ja_JP.UTF-8	ibm-5026
ja, ja_JP.PCK, ja_JP.UTF-8	ibm-5035
ja, ja_JP.PCK, ja_JP.UTF-8	ms-932
ja, ja_JP.PCK, ja_JP.UTF-8	UTF-8-ms932

A.5 Korean Solaris 8 Operating Environment Features

Table A-15 to Table A-18 give an overview of the Korean Solaris 8 operating environment features.

TABLE A-15 Korean Codesets

Locale Name	Description	Supported Character Set
ko	Korean (EUC)	KS C 5601-1992
ko.UTF-8	Korean (UTF-8)	Unicode 3.0

TABLE A-16 Korean Input Methods

Locale Name	Input Methods
ko, ko.UTF-8	Hangul 2-BeolSik (1 set of consonants and 1 set of vowels)
ko, ko.UTF-8	Hangul-Hanja conversion
ko, ko.UTF-8	Special character
ko.UTF-8	Unicode Hex and Unicode Octal

TABLE A-17 Korean Fonts

Locale Name	Full Family Name	Format
ko, ko.UTF-8	Gothic	TrueType
ko, ko.UTF-8	Haeso	TrueType
ko, ko.UTF-8	Kodig	TrueType
ko, ko.UTF-8	Myeongijo	TrueType
ko, ko.UTF-8	Roundgothic	TrueType
ko, ko.UTF-8	Gothic	PCF (14,16,18,20,24)
ko, ko.UTF-8	Graphic	PCF (14,16,18,20,24)
ko, ko.UTF-8	Haeso	PCF (14,16,18,20,24)
ko, ko.UTF-8	Kodig	PCF (14,16,18,20,24)
ko, ko.UTF-8	Myeongijo	PCF (14,16,18,20,24)
ko, ko.UTF-8	Pilki	PCF (14,16,18,20,24)
ko, ko.UTF-8	Roundgothic	PCF (14,16,18,20,24)

TABLE A-18 Korean Codeset Conversions

Locale Name	Codeset Conversion Supported
ko, ko.UTF-8	KSC 5601-1987
ko, ko.UTF-8	ISO 646
ko, ko.UTF-8	UTF-8
ko, ko.UTF-8	IBM CP933
ko, ko.UTF-8	ISO 2022-KR
ko, ko.UTF-8	KSC 5601-1987-Johap
ko, ko.UTF-8	KSC 5601-1992-Johap
ko, ko.UTF-8	Unified Hangul

A.5.1 Korean Dictionary Tools

Hanja Tool expands the capabilities of the standard Korean Solaris operating environment hangul-hanja conversion mode by adding hanja ideograms and managing available lookup choices for hangul-hanja conversion. It works only with words of more than one syllable.