Asian Application Developer's Guide

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043-1100
U.S.A.

Part No: 802-7789
June 1997

Contents

Preface

Asian Application Developer's Guide provides application development information specific to the operation of the following three separate Sun® products, which are identified generically as Asian Solaris™ software:

- Korean Solaris software
- Simplified Chinese Solaris software
- Traditional Chinese Solaris software

This documentation complements Sun's standard Common Desktop Environment (CDE) and Solaris base-release product documentation. It includes information that advanced users and developers can use to access and control the Korean or Chinese language-related features of Asian Solaris software.

Who Should Use This Book

You should already be familiar (but need not be expert) with Solaris, the windowing environment you are using, and their documentation.

You should read this manual if:

- You need specific instructions on how to set up Korean or Chinese Solaris language features for users.
- You are a developer who has not used the Korean or Chinese Solaris operating environment before.
- You are a developer who needs information on accessing and controlling the Korean- or Chinese-related features of Asian Solaris software.
- You are an advanced user who wants to use or customize Korean- or Chinese-related features of Asian Solaris software.

- You want information on a variety of Korean- or Chinese-related details internal to the operation of Asian Solaris software.

Before You Read This Book

Before you read this book, read *Common Desktop Environment: Internationalization Programmer's Guide* and *Solaris Internationalization Guide for Developers* for basic internationalization issues. You should also read an overview of your Asian Solaris product and last-minute changes not included in this document. The following documents contain this information:

- The release overview for your Asian Solaris locale
- *Asian Solaris Release and Installation Notes (Intel Platform Edition)*
- *Asian Solaris Release and Installation Notes (SPARC Platform Edition)*
- *Solaris Advanced Installation Guide*
- *Information Library for Solaris (Intel Platform Edition)*
- *Information Library for Solaris (SPARC Platform Edition)*

How This Book Is Organized

Each chapter of *Asian Application Developer's Guide* addresses a different aspect of development for a Korean or Chinese feature of Asian Solaris software. Some chapters give step by step instructions for using or customizing product features.

Chapter 1 describes the Asian Solaris locale environments.

Chapter 2 describes the mechanisms for handling different character sets and multiple code sets.

Chapter 3 describes the `iconv` library, a conversion utility for character-based code.

Chapter 4 describes the fonts and font sets that come with the software and shows basic ways to use them.

Chapter 5 describes the X Window input method architecture.

Chapter 6 explains how to use DPS for Asian font displays in the Asian Solaris operating environment.

Appendix A describes information unique to OpenWindows environment programming.

Appendix B describes utilities useful for backward compatibility.

Related Books

The following books are related to the topic of this book and may contain useful information.

For information on how to use the window system and associated applications, see:

- The user's guide for your Asian Solaris product
- *Solaris User's Guide*

For information on Asian Solaris system administration, see the system administration guide for your Asian Solaris localization.

What Typographic Changes 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

System Environment

Asian Solaris software enables you to set the Korean or Chinese environment or use the English environment:

- A general *locale setting* for all locale-related aspects of the environment, or
- A specific *locale category setting* for a particular aspect of the environment

The general locale setting is called `LC_ALL`. The specific locale category settings each have a name that begins with `LC_` but ends with a different suffix depending on the category. The designation `LC_`*XXX* refers to any one of the locale category names.

Asian Solaris Operating Environment Locales

The Asian Solaris products include the following locales, the English UTF-8 locale, and a number of partial locales (partial locales are locales that do not contain translated messages). For more information, see *Solaris Internationalization Guide for Developers.*

- Korean Solaris includes:
  - `ko` – the Korean environment in EUC (KS C 5601).
  - `ko.UTF-8` – the Korean environment in Universal Multiple-Octet Coded Character Set (UCS) Transmission Format (KS C 5700). This encoding does not support the OpenWindows environment.
- Simplified Chinese Solaris includes `zh` – the Simplified Chinese environment in EUC

- Traditional Chinese Solaris includes:
  - `zh_TW` – the Simplified Chinese environment in EUC.
  - `zh_TW.BIG5` – the Traditional Chinese environment in Big5. This encoding does not support the OpenWindows environment.

Typically, an application uses the value of the `LANG` environment variable to set each category that has not previously been set explicitly using `LC_`*XXX*. In the Korean or Chinese Solaris environments, the value of the `LANG` environment variable is as follows:

- `ko` and `ko.UTF-8` for Korean
- `zh` for Simplified Chinese
- `zh_TW` and `zh_TW.BIG5` for Traditional Chinese
- `C` for the English locale
- `en_US.UTF-8` the locale that handles multiple scripts simultaneously

---

Note - All Asian applications must be internationalized. For details on this process, see *Solaris Internationalization Guide for Developers.*

Asian Locale Files

Korean and Chinese locale-specific files and libraries are located in the following directories:

- Korean Solaris software:
  - `/usr/lib/locale/`
  - `/usr/openwin/lib/locale/ko/`
  - `/usr/openwin/lib/locale/ko.UTF-8/`
  - `/usr/lib/locale/ko`
  - `/usr/lib/locale/ko.UTF8`
- Simplified Chinese Solaris software: `/usr/openwin/lib/locale/zh/`
- Traditional Chinese Solaris software:
  - `/usr/openwin/lib/locale/zh_TW/`
  - `/usr/openwin/lib/locale/zh_TW.BIG5/`

CDE Message Files

For details on how window system messages are generated and how to create message files for your own applications, see *Common Desktop Environment: Internationalization Programmer's Guide*.

Korean Messages

Files in the `/usr/dt/lib/nls/msg/ko` and `/usr/dt/lib/nls/msg/ko.UTF-8` directories with `.cat` filename extensions contain messages used in Korean CDE.

Simplified Chinese Messages

Files in the `/usr/dt/lib/nls/msg/zh/` directory with `.cat` file name extensions contain messages used in Simplified Chinese CDE.

Traditional Chinese Messages

Files in the `/usr/dt/lib/nls/msg/zh_TW/` and `/usr/dt/lib/nls/msg/zh_TW.BIG5/` directories with `.cat` file name extensions contain messages used in Traditional Chinese CDE.

`fonts` Directory

The `/usr/openwin/lib/locale/`*locale*`/X11/fonts` directory holds the Korean or Chinese fonts (*locale* is `ko`, `ko.UTF`, `zh`, `zh_TW`, or `zh_TW.BIG5`). This directory must be in your font path in order to access Korean or Chinese fonts. (The `xsession` script that comes with the Asian Solaris operating environment includes this directory in the font path.) To add a different font directory path dynamically, type:

```
% xset +fp font_directory_path
```

`print` Directory

The `/usr/dt/config/psfonts/`*locale*`/print` directory contains `locale`-specific printing files (*locale* is `ko`, `ko.UTF-8`, `zh`, `zh_TW`, or `zh_TW.BIG5`). Most notable is the `prolog.ps` file, which is used by the CDE tools that come with the Korean or Chinese Solaris operating environment. For information on using `prolog.ps`, refer to the printing facilities chapters in the Asian Solaris user's guide appropriate to your Solaris operating environment.

app-defaults Directory

The /usr/dt/app-defaults/*locale* directory contains locale-specific resource files for CDE applications (*locale* is ko, ko.UTF-8, zh, zh_TW, or zh_TW.BIG5).

CDE Configuration Directory

The /usr/dt/config/*locale* directory contains locale-specific resource files for CDE applications (*locale* is ko, ko.UTF-8, zh, zh_TW, or zh_TW.BIG5).

CDE Action and Data Type Definition Directory

The /usr/dt/appconfig/types/*locale* directory contains locale-specific action and data type definition files for CDE (*locale* is ko, ko.UTF-8, zh, zh_TW, or zh_TW.BIG5).

CHAPTER 2

Language Environment and Character Codes

Asian Solaris software enables you to switch between different languages, or character sets, and can represent very large character sets. Some operating systems use the 7-bit ASCII codeset to represent English language characters. It has become a common assumption that characters equal bytes. Because the Korean and Chinese character sets are very large, more than one byte is required to represent each character. Asian Solaris software provides a mechanism for specifying multiple codesets. These codesets may contain characters of more than one byte. Character sets and codesets are defined as follows:

- A *character set* is a set of elements used for the organization, control, or representation of data. Character sets may be composed of alphabets, ideograms, or other units.
- A *codeset*, or coded character set, is a set of unambiguous rules that establishes a character set and the one-to-one relationship between each character in the character set and its bit representation.

For example, the ASCII codeset contains a bit representation for each uppercase and lowercase alphabetic letter as well as punctuation, numbers, and control codes.

Extended UNIX Code (EUC)

Asian Solaris software implements Extended UNIX® Code (EUC) specified by the SVR4 Multi-National Language Supplement (MNLS), which follows the pattern of ISO 2022 standards. Four single-byte and multibyte codesets can be represented in EUC at both the process level and the file level.

EUC is used as file code for storing data and internally in the CPU and RAM memory. It is composed of one or more bytes and may be accompanied by single-shift characters.

EUC Definition

EUC is composed of one primary codeset and three supplementary codesets. The primary codeset, codeset 0, is used for ASCII. The three supplementary codesets (codesets 1, 2, and 3) can be assigned to different character sets by the user. There is a system default assignment for these codesets.

The primary codeset is defined to use single bytes with the most significant bit (MSB) set to zero. The supplementary codesets can use multiple bytes, and the MSB of each byte is set to one. Codesets 2 and 3 have a preceding single-shift character, known as SS2 (0x8E) in codeset 2 and SS3 (0x8F) in codeset 3. Differentiating between codesets is done as follows: If the MSB is 0, the code is one-byte ASCII. If the MSB is 1, the byte is checked (SS2 or SS3) to determine which codeset it belongs to. The length in bytes of characters from that codeset is retrieved from an ANSI localization table governing character classification, and that number of bytes is read in.

TABLE 2–1 EUC Codeset Representations

Codeset	EUC Representation
codeset 0	0xxxxxxx
codeset 1	1xxxxxxx -or- 1xxxxxxx 1xxxxxxx -or- 1xxxxxxx 1xxxxxxx 1xxxxxxx
codeset 2	SS2 1xxxxxxx -or- SS2 1xxxxxxx 1xxxxxxx -or- SS2 1xxxxxxx 1xxxxxxx 1xxxxxxx
codeset 3	SS3 1xxxxxxx -or- SS3 1xxxxxxx 1xxxxxxx -or- SS3 1xxxxxxx 1xxxxxxx 1xxxxxxx

EUC Special Characters

In accord with ISO 2022 and ISO 6937/3, EUC divides the codeset space into graphic and special characters. Graphic characters are those that have a glyph or shape that can be displayed. Special characters include Control characters, unassigned characters, and the Space and Delete characters. Control characters are characters, other than graphic characters, whose occurrence in a particular context initiates, modifies, or stops a control operation.

TABLE 2–2 Single-Byte Special Character Representations

Special Character	EUC Representation
Space	00100000
Delete	01111111
Control codes (Primary)	000xxxxx
Control codes (Supplementary)	100xxxxx

Wide Character (WC)

The wide character (WC) is defined in Asian Solaris software to be a constant-width four-byte code. It provides a standard character size, which is useful in indexing, interprocess communication, memory management, and other tasks that use character counts and known array sizes.

Note - Wide characters are intended for internal processing only. Applications should not depend on the wide character implementation, but use standard library APIs to handle wide characters.

Korean Solaris Supported Character Sets

Three types of coding conventions are currently supported in the Korean Solaris software:

- N-byte code. This single-byte code has each byte represent a consonant or vowel. These are combined together to build Hangul characters.

- Johap or Packed code. This two-byte code consists of a leading bit followed by three 5-bit fields. These three fields contain the codes for a leading consonant, followed by a vowel, followed by a final consonant (if any) for a Hangul character. This two-byte code is specified in Korean Industry Standard KS C 5601-1992.
- Wansung code. This two-byte code is specified in Korean Industry Standard KS C 5601-1987 for Hangul, Hanja, and other characters. In the Korean Solaris software these KS C 5601-1987 characters are in EUC codeset 1.
- `ko.UTF-8` – Korean Universal Multiple Octet Coded Character Set (UCS) Transmission Format. See "The `ko.UTF-8` Locale " on page 8 for further information.

Korean Solaris software provides code conversion between these four Korean code conventions at three levels of support:

- User commands support file transfers for existing files in different codes.
- Library functions support application development for existing codes.
- STREAMS modules support existing TTY devices using different codes.

The `ko.UTF-8` Locale

The Korean government announced the standard Korean codeset KS C 5700, which is based on Unicode 2.0. KS C 5700 will be widely used in the Korean market, replacing the previous standard, KS C 5601, which is based on ISO 2022.

To comply with this new standard, the `ko.UTF-8` locale was developed. UTF-8 is a file system safe (Universal Character Set Transformation Format) Unicode, which is based on ISO 10646-1/Unicode 2.0.

`ko.UTF-8` supports all the characters of KSC 5601 and 11,172 characters from Johap. `ko.UTF-8` supports all Korean-related Unicode 2.0 characters and fonts. All Unicode characters can be accepted and processed, but some cannot be correctly displayed because of input and output limitations.

`ko.UTF-8` supports the following subset of Unicode:

- Basic Latin and Latin-1 (190 characters) – Row 00 of BMP (Basic Multilingual Plan)
- Symbolic characters – Row 20 to Row 27, and Row 32 of BMP Including box (line) drawing characters that are defined in KS C 5601
- Numerals that are defined in KSC 5601 (20 characters) – Row 21 and Row FF of BMP
- Roman, Greek, Japanese, and Cyrillic alphabet characters that are defined in KS C 5601 (362 characters) – Row 03, Row 04, Row 30 and Row FF of BMP
- Jamo (Hangul alphabet) characters (94 characters) – Row 31 of BMP
- Pre-composed Hangul syllables (11,172 characters) – From Row AC to Row D7 of BMP

- Hanja characters defined in KS C 5601 (4,888 characters) – From Row 4E to Row 9F and from Row F9 to Row FA of BMP

Simplified Chinese Solaris Supported Character Sets

Simplified Chinese Solaris supports the PRC Chinese national standard character set (GB 2312–80). GB 2312–80 consists of 7,445 characters: 3,755 level–1 Hanzi characters, 3,008 level–2 Hanzi characters and Hanzi radicals, Roman characters, Greek and Cyrillic characters, Arabic and Greek numerals, and miscellaneous symbols.

Chinese Solaris software provides code conversion between Chinese code conventions at two levels of support:

- User commands support file transfers for existing files in different codes.
- Library functions support application development for existing codes.

Traditional Chinese Solaris Supported Character Sets

Traditional Chinese Solaris supports the Taiwanese Chinese National Standard CNS 11643–1992 and Big5 character sets. CNS 11643–1992 is a Chinese national standard in Taiwan. It defines 16 planes:

- Plane 1:

  Miscellaneous symbols, Hanzi radicals, and Roman and Greek alphabets, total 684 symbol characters in the range of 0x2121 to 0x427E, and 5,401 most commonly used Hanzi characters in the range of 0x4421 to 0x7D4B.

- Plane 2:

  7,650 secondary commonly-used Hanzi characters in the range of 0x2121 to 0x7244.

- Plane 3:

  A total of 6,148 other Hanzi characters, including some user-defined characters from the original plane 14 characters and different shaped characters in the range 0x2121-0x6246 from the Republic of China's (ROC) Department of Education.

- Plane 4:

  This plane contains a total of 7,298 characters, including some of ISO/IEC 10646 defined CJK Unified Han characters (range: 0x2121-0x6E5C).

- Plane 5:

  This plane contains a total of 8,603 characters that the ROC Department of Education defined as currently-used characters but not included in planes 1 through 4 (range: 0x2121-0x7C51).

- Plane 6:

  This plane contains a total of 6,388 characters that the ROC Department of Education defined as different shaped characters but not included in planes 1 through 5 (range: 0x2121-0x647A).

- Plane 7:

  This plane contains a total of 6,539 characters that the ROC Department of Education defined as different shaped characters but not included in planes 1 through 6 (range: 0x2121-0x6655).

- Plane 8 to 11:

  These planes are not yet defined.

- Plane 12 to 16:

  These planes are for user-defined characters.

Big5 was defined by five major Taiwanese computer vendors (including the Institute of Information Industry) in May of 1984. Although Big5 is not the national standard, it is more widely used than the CNS 11634–1992.

The total number of characters defined in Big5 is 13,523. It is a subset of CNS 11643–1992.

Traditional Chinese Solaris software provides code conversion between Chinese code conventions at three levels of support:

- User commands support file transfers for existing files in different codes.
- Library functions support application development for existing codes.
- STREAMS modules support existing TTY devices using different codes.

CHAPTER 3

The Codeset Conversion Utility

Asian Solaris software provides the `iconv` library as a conversion utility for character-based code conversion.

Code Set Conversion Modules for Korean

Conversion for Korean EUC Character Codes

The following modules perform character-based code conversion on the KS C 5601 character set. They convert KS C 5601 characters, also called Completion code or Wansung, to Combination code (Johap), and vice versa.

For further information, see the `iconv`(3) and `iconv_ko`(5) `man` pages.

TABLE 3–1 Korean `iconv` Code Conversion Modules (`ko` locale)

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

TABLE 3–1 Korean `iconv` Code Conversion Modules (`ko` locale) *(continued)*

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

Conversion for `ko.UTF-8` Character Codes

The following modules perform character-based code conversion on the KS C 5700 character set. They convert KSC 5700 characters between Korean UTF-8, Completion code (Wansung), and Combination code (Johap).

For further information, see the `iconv`(3), `iconv_ko.UTF-8`(5), `iconv_utf`(5) `man` pages.

TABLE 3–2 Common Korean `iconv` Code Conversion Modules (`ko` and `ko.UTF-8` locales)

Code	Symbol	Target Code	Symbol
UTF-8	ko_KR-UTF-8	Wansung	ko_KR-euc
UTF-8	ko_KR-UTF-8	Johap	ko_KR-johap92
UTF-8	ko_KR-UTF-8	Packed	ko_KR-johap
UTF-8	ko_KR-UTF-8	ISO-2022-KR	ko_KR-iso2022-7
Wansung	ko_KR-euc	UTF-8	ko_KR-UTF-8
Johap	ko_KR-johap92	UTF-8	ko_KR-UTF-8
Packed	ko_KR-johap	UTF-8	ko_KR-UTF-8
ISO-2022-KR	ko_KR-iso2022-7 UTF-8	UTF-8	ko_KR-UTF-8

Code Set Conversion Modules for Simplified Chinese

The following code set conversion modules are supported in Simplified Chinese Solaris software. For further information, see the `iconv`(3) and `iconv_zh`(5) `man` pages.

TABLE 3–3 Simplified Chinese `iconv` Code Conversion Modules (`zh` locale)

Code	Symbol	TargetCode	Symbol
GB2312-80	zh_CN.euc	ISO 2022-7	zh_CN.iso2022-7
ISO 2022-7	zh_CN.iso2022-7	GB2312-80	zh_CN.euc
GB2312-80	zh_CN.euc	ISO 2022-CN	zh_CN.iso2022-CN
ISO-2022-CN	zh_CN.iso2022-CN	GB2312-80	zh_CN.euc
UTF-8	UTF-8	GB2312-80	zh_CN.euc
GB2312-80	zh_CN.euc	UTF-8	UTF-8

Code Set Conversion Modules for Traditional Chinese

The following code set conversion modules are supported in Traditional Chinese Solaris. For further information, see the `iconv`(3) and `iconv_zh_TW`(5) `man` pages.

TABLE 3–4 Traditional Chinese `iconv` Code Conversion Modules (`zh_TW` and

TABLE 3–4 Traditional Chinese `iconv` Code Conversion Modules (`zh_TW` and `zh_TW.BIG5` locales) *(continued)*

`zh_TW.BIG5` locales)

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-euc	ISO 2022-CN-EXT	zh_TW-iso2022-CN-EXT
ISO 2022-CN-EX	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	CNS 11643	UTF-8	UTF-8
UTF-8	UTF-8	Big-5	zh_TW-big5
Big-5	zh_TW-big5	UTF-8	UTF-8
UTF-8	UTF-8	ISO 2022-7	zh_TW-iso2022-7
ISO 2022-7	zh_TW-iso2022-7	UTF-8	UTF-8
ISO 2022-CN-EXT	zh_TW-iso2022-CN-EXT	Big-5	zh_TW-big5
Big-5	zh_TW-big5	ISO 2022-CN-EXT	zh_TW-iso2022-CN-EXT

CHAPTER 4

Asian Solaris Fonts

This chapter discusses Korean and Chinese font-related information for developers and programmers.

Korean Fonts

Six typefaces of Korean scalable and bitmap fonts are provided in the Korean Solaris software:

- Kodig
- Myeongjo
- Round Gothic
- Pilki
- Haeso
- Graphic

Kodig and Myeongjo contain Korean characters in accord with the KS C 5601 and KS C 5700 standards. Gothic is an alias of Kodig, and Myeongjo is an alias of Myeongjo.

Korean Solaris software includes bitmap and scalable Korean fonts under the `/usr/openwin/lib/locale/ko/X11/fonts` and `/usr/openwin/lib/locale/ko.UTF-8/X11/font` directories.

Korean bitmap fonts are in the subdirectories:

- `/usr/openwin/lib/locale/ko/X11/fonts/75dpi`
- `/usr/openwin/lib/locale/ko.UTF-8/X11/fonts/75dpi`

  The `fontedit` utility enables users to edit bitmap fonts.

Korean F3 scalable fonts are in the subdirectories:

- `/usr/openwin/lib/locale/ko/X11/fonts/F3`
- `/usr/openwin/lib/locale/ko.UTF-8/X11/fonts/F3`

Korean character ID (CID) scalable fonts are in the subdirectories:

- `/usr/openwin/lib/locale/ko/X11/fonts/cidtype1fonts`
- `/usr/openwin/lib/locale/ko.UTF-8/X11/fonts/cidtype1fonts`

You can use X Font Displayer (`xfd`) to view a Korean font, for example:

```
system% xfd -fn kodig-medium14
```

The `xfd`(1) `man` page provides more information.

All Korean fonts have XLFD font names, and you can use the `xlsfonts` utility to display these names, as in the following example:

```
system% xlsfonts | grep kodig
```

Simplified Chinese Fonts

Simplified Chinese Solaris software provides Song-style fonts, with Simplified Chinese characters in accord with the GB2312-80 standard.

Bitmap Simplified Chinese fonts are provided, located under the `/usr/openwin/lib/locale/zh/X11/fonts` directory. The `fontedit` utility enables users to edit bitmap fonts.

Simplified Chinese Solaris provides TrueType scalable fonts in the following type faces:

- Hei
- Kai
- Song
- Fangsong

You can use X Font Displayer (`xfd`) to view a Simplified Chinese font, for example:

```
system% xfd -fn song-medium14
```

The `xfd`(1) `man` page provides more information.

All Simplified Chinese fonts have XLFD font names, and you can use the `xlsfonts` utility to display these names as follows, for example:

```
system% xlsfonts | grep gb2312
```

Traditional Chinese Fonts

Bitmap, scalable, and PostScript composite Traditional Chinese fonts are provided, located in the `/usr/openwin/lib/locale/zh_TW/X11/fonts` directory.

These bitmap fonts are in:

- `/usr/openwin/lib/locale/zh_TW/X11/fonts/75dpi`
- `/usr/openwin/lib/locale/zh_TW.BIG5/X11/fonts/75dpi`

The `fontedit` utility enables users to edit bitmap fonts.

Traditional Chinese Solaris provides TrueType scalable fonts in the following type faces:

- Hei
- Kai
- Ming

These scalable fonts are in:
`OPENWINHOME/lib/locale/zh_TW/X11/fonts/TrueType`

Chinese PostScript composite fonts are in:
`/usr/openwin/lib/locale/zh_TW/X11/fonts/composite`

You can use X Font Displayer (`xfd`) to view a Traditional Chinese font, for example:

```
system% xfd -fn ming-light14
```

The `xfd`(1) `man` page provides more information.

All Traditional Chinese fonts have XLFD font names, and you can use the `xlsfonts` utility to display these names as in the following example:

```
system% xlsfonts | grep sung
......
system% xlsfonts | grep kai
```

Programmer's Font Set Functions

Asian Solaris software uses the font set mechanisms defined in `X11R6`. Programmers can access the font set using `X11R6` calls such as:

- `XCreateFontSet`()
- `XmbTextEscapement`()
- `XmbTextExtents`()
- `XmbDrawString`()
- `XwcTextEscapement`()
- `XwcTextExtents`()
- `XwcDrawString`(), etc.

Refer to the X11R6 document for a list of text drawing functions also available in the Korean or Chinese Solaris software.

Editing Fonts with `fontedit`

Asian Solaris software provides support for editing bitmap fonts. The Font Editor is in `/usr/openwin/bin/fontedit`.

The Font Editor handles English or Asian characters in Bitmap Distribution Format (BDF), a portable format defined by the MIT X Consortium. BDF font files have the file extension `.bdf`.

However, the Asian Solaris operating environment uses Portable Compiled Format (PCF) for bitmap fonts, and Solaris software provides tools to convert between BDF and PCF.

The process of modifying a font involves the following steps.

1. **Locate an Asian font file in BDF.**

   The encoding should start at 8481(0X2121), as shown in the following example of `myfont16.bdf`.

```
STARTFONT 2.1
COMMENT Sample Font
FONT Myfont-Medium
SIZE 16 75 75
 ...
STARTCHAR C101
ENCODING  8481
 ...
```

2. **Start `fontedit` to modify the BDF file and choose either single-byte encoding for ASCII/English characters or multibyte encoding for Korean or Chinese characters.**

3. **Edit your chosen font characters using the left mouse button to turn pixels on and the middle button to turn pixels off.**

   For information on using Font Editor, see the chapter on using Font Editor with your locale in the Asian Solaris user's guide appropriate to your localization.

4. **Save the edited font.**

5. **Make the edited Asian BDF font(s) usable by the Asian Solaris operating environment by converting to PCF format, as in the following example:**

```
system% bdftopcf Myfont16.bdf > Myfont-Medium16.pcf
system% compress Myfont-Medium16.pcf
```

Adding Bitmap Fonts

This section describes how to add a new bitmap font to your environment. The steps assume the working directory is your font directory.

Note - Solaris 2.6 includes the Font Admin tool, which you may prefer to use for configuring fonts. However, the tools is available only if you install the entire cluster. For information about the Font Admin tool, see *System Administration Guide.*

---

1. **Move the `.pcf` file for the new font into your font directory.**

2. **Run the `mkfontdir` command:**

```
system% mkfontdir
```

3. **If you have an XLFD name for the font, put the definition of this font into `fonts.alias`, as in the following example:**

   For a Korean font:

```
system% cat >> fonts.alias
-new-myfont-medium-r-normal--18-160-75-75-c-160-ksc5601.1987-0
Myfont-Medium16
^D
```

For a Simplified Chinese font:

```
system% cat >> fonts.alias
-new-myfont-medium-r-normal--18-160-75-75-c-160-gb2312.1980-0
Myfont-Medium16
^D
```

For a Traditional Chinese font:

```
system% cat >> fonts.alias
-new-myfont-medium-r-normal--18-160-75-75-c-160-cns11643-16
Myfont-Medium16
^D
```

4. Set the user preference options of the display, as follows:

```
system% xset +fp ‘pwd‘
system% xset fp rehash [[optional]]
```

Viewing Bitmap Fonts

You can view a font by typing:

```
system% xfd -fn Myfont-Medium16
```

With the Korean locale:

```
system% xfd -fn "-new-myfont-medium--r-normal--18-160-75-75-c-
160-ksc5601.1987-0"
```

With the Simplified Chinese locale:

```
system% xfd -fn "-new-myfont-medium--r-normal--18-160-75-75-c-
160-gb2312.1980-0"
```

With the Traditional Chinese locale:

```
system% xfd -fn "-new-myfont-medium--r-normal--18-160-75-75-c-
160-cns11643-16"
```

More on Font Encoding and Editing Fonts

The X11 encoding of Asian characters follows a 7-bit multibyte scheme with the leading bit of each byte set to zero. This differs from EUC use of 8-bit multibyte encoding with the leading bit set to one. BDF defines the encoding to be a positive decimal integer.

For convenience, `fontedit` lets you specify the 7-bit values of the high and low bytes, which are automatically combined into the decimal value used in the BDF file. For example, to locate EUC character 0xA1A2, you would find the following:

TABLE 4–1 Font Encoding

8-bit Encoding	7-bit Encoding	BDF Value
0xA1A2	0x2122	8482

The character appears in its actual size just above the bitmap editing area as you edit it, and immediately shows the effects of turning pixels on and off.

Font Lists

Asian Solaris 2.6 uses font lists to display text. A font defines a set of glyphs that represent the characters in a given character set. A “font set” is a collection of fonts needed to display text for a specific locale or language. A “font list” is a list of fonts, font sets, or a combination of the two. This section describes the Asian Solaris font lists and how to select them when starting an Asian Solaris application. The files in `/usr/dt/app-defaults/`*`locale`*`/*` define the system default font lists, where *locale* is `ko`, `ko.UTF-8`, `zh`, `zh_TW`, or `zh_TW.BIG5`.

Korean Font Lists

In the Asian Solaris 2.6 environment, a Korean font list is composed of one English font representing codeset 0 (ASCII) characters in KS C 5636 or ISO8859-1, and one Korean font representing codeset 1 characters in KS C 5601-1987-0.

The KS C 5636 and ISO8859-1 character sets are nearly identical. The differences are that KS C 5636 uses only the code values from 0 to 127, and the backslash character (whose ISO8859-1 code value is 92) is replaced by the Korean currency symbol. Asian Solaris 2.6 provides some default font lists defined in the application defaults files in `/usr/dt/app-defaults/ko/*`. The following is an excerpt from one of the files, `Dtwm`:

```
Dtwm*icon*fontList: \      -dt-interface system-medium-r-normal-s*ksc*:
```

This portion of the file refers to a font list that contains two fonts previously mentioned that are included in the `ko` locale:

```
English font, for codeset 0 (ASCII) character font display:
-dt-interface system-medium-r-normal-s sans-14-120-75-75-p-60-ksc5636-0
Korean font, for codeset 1 (KS C 5601-1987-0) character font display:
-dt-interface system-medium-r-normal-s sans-14-120-75-75-p-120-ksc5601.1987-0
```

Note that these fonts are defined in the file `/usr/openwin/lib/locale/ko/X11/fonts/75dpi/fonts.alias`.

Starting Applications with a Specific Korean Font List

When you start an Asian Solaris tool at the command line, you can also specify its fonts. Below is an example of a command line argument used to start a new Korean Solaris terminal with a specified font list:

```
system% dtterm  -fn "-dt-interface system-medium-r-normal-s
sans-14-120-75-75-p-60-ksc5636-0;\  -dt-interface
```

(Continuation)

```
system-medium-r-normal-s sans-14-120-75-75-p-120-ksc5601.1987-0:"
```

Note the two delimiters used in the font list. The ";" delimiter is used to separate the font names except for the last font name, which ends with the ":" delimiter. (In the example above, ";" follows the English font name, and the ":" delimiter follows the Korean font name.) Since there are spaces in the long font names, the font list is enclosed in quotation marks.

Korean UTF-8 Font Lists

In the Asian Solaris 2.6 environment, a Korean UTF-8 font list is composed of one English font representing codeset 0 (ASCII) characters in KS C 5636 or ISO8859-1, and one Korean Johap font representing codeset 1 characters in KS C 5601-1992-3. Asian Solaris 2.6 provides some default font lists defined in the application defaults files in `/usr/dt/app-defaults/ko.UTF-8/*`. The following is an excerpt from one of the files, `Dtwm`:

```
Dtwm*icon*fontList: \  -dt-interface system-medium-r-normal-s*ksc*:
```

This portion of the file refers to a font list that contains two fonts previously mentioned that are included in the ko locale:

```
English font, for codeset 0 (ASCII) character font display:
-dt-interface system-medium-r-normal-s sans-14-120-75-75-p-60-ksc5636-0
Korean Johap font, for codeset 1 (KS C 5601-1992-3) character font display:
-dt-interface system-medium-r-normal-s sans-14-120-75-75-p-120-ksc5601.1992-3
```

Note that these fonts are defined in the file `/usr/openwin/lib/locale/ko.UTF-8/X11/fonts/75dpi/fonts.alias`

Starting Applications with a Specific Korean UTF-8 Font List

When you start an Asian Solaris tool at the command line, you can also specify its fonts. Below is an example of a command line argument used to start a new Korean Solaris terminal with a specified font list in the Korean UTF-8 locale environment:

```
system% dtterm  -fn "-dt-interface system-medium-r-normal-s
sans-14-120-75-75-p-60-ksc5636-0;\  -dt-interface
system-medium-r-normal-s sans-14-120-75-75-p-120-ksc5601.1992-3:"
```

Note the two delimiters used in the font list. The ";" delimiter is used to separate the font names except for the last font name, which ends with the ":" delimiter. (In the example above, ";" follows the English font name, and the ":" delimiter follows the Korean UTF-8 font name.) Since there are spaces in the long font names, the font list is enclosed in quotation marks.

Simplified Chinese Font Lists

In the Asian Solaris 2.6 environment, a Simplified Chinese font list is composed of one English font representing codeset 0 (ASCII) characters in `gb1988.1989-0` or `ISO8859-1`, and one Simplified Chinese font representing `gb2312.1980-0` characters.

Simplified Chinese Solaris 2.6 provides some default font lists defined in application defaults files in `/usr/dt/app-defaults/zh/*`. The following is an excerpt from one of the files, `Dtwm`:

```
Dtwm*icon*fontList: \  -dt-interface system-medium-r-normal-s*-*-*-*-*-*-*-*-*:
```

This portion of the file refers to a font list that contains two fonts previously mentioned that are included in the `zh` locale.

```
"-dt-interface system-medium-r-normal-s serif-14-120-75-75-p-60
-gb1988.1989-0""-dt-interface system-medium-r-normal-s serif-14-120-75-75-p-120
-gb2312.1980-0"
```

The first is an English font for codeset 0 (ASCII) character font display. The second is a Simplified Chinese font for codeset 1 (`GB2312.1980`) character font display.

Note that these fonts are defined in the file `/usr/openwin/lib/locale/zh/X11/fonts/75dpi/fonts.alias`.

Starting Applications with a Specific Simplified Chinese Font List

When you start an Asian Solaris tool at the command line, you can also specify its fonts. Below is an example of a command line argument used to start a Simplified Chinese Windows terminal with a specified font list:

```
system% dtterm -fn "-dt-interface system-medium-r-normal-s \
serif-14-120-75-75-p-60-gb1988.1989-0; \
-dt-interface system-medium-r-normal-s \
```

(continued)

```
serif-14-120-75-75-p-120-gb2312.1980-0:"
```

Note the two delimiters used in the font list. The ";" delimiter is used to separate the font names except for the last font name, which ends with the ":" delimiter. (In the example above, ";" follows the English font name, and the ":" delimiter follows the Simplified Chinese font name.) Since there are spaces in the long font names, the font list is enclosed in quotation marks.

Traditional Chinese (zh_TW) Font Lists

In the Asian Solaris 2.6 environment, a Traditional Chinese (zh_TW) font list is composed of one English font, representing ASCII characters in CNS11643-0 or ISO8859-1, and a number of Traditional Chinese fonts representing characters such as CNS11643-1, CNS1643-1, CNS11643-2, and CNS11643-3.

Traditional Chinese Solaris provides some default font lists defined in application defaults files in `/usr/dt/app-defaults/zh_TW/*`. The following is an excerpt from one of these files, `Dtwm`:

```
Dtwm*icon*fontList: \  -dt-interface system-medium-r-normal-s*-*-*-*-*-*-*-*-*:
```

This portion of the file refers to a font list that contains the following fonts, which are defined in `/usr/openwin/lib/locale/zh_TW/X11/fonts/75dpi/fonts.alias`:

```
"-dt-interface system-medium-r-normal-s serif-16-140-75-75-p-70-cns11643-0"
"-dt-interface system-medium-r-normal-s serif-16-140-75-75-p-140-cns11643-1"
"-dt-interface system-medium-r-normal-s serif-16-140-75-75-p-140-cns11643-2"
"-dt-interface system-medium-r-normal-s serif-16-140-75-75-p-140-cns11643-3"
```

The first is the English font for codeset 0 (ASCII) character font display. The rest are Traditional Chinese fonts for codeset 1 (CNS11643) plane 1 character font display, and codeset 2 (CNS11643) plane 2 and plane 3 character font display.

Starting Applications with a Specific Traditional Chinese Font List

When you start an Asian Solaris tool at the command line, you can also specify its fonts. Below is an example of a command line argument used to start a new Traditional Chinese Windows terminal with a specified font list:

```
system% dtterm -fn  "-dt-interface system-medium-r-normal-s \
serif-16-140-75-75-p-70-cns11643-0; \   -dt-interface
system-medium-r-normal-s \   serif-16-140-75-75-p-140-cns11643-1:"
```

Note the two delimiters used in the font list. The ";" delimiter is used to separate the font names except for the last font name, which ends with the ":" delimiter. (In the example above, ";" follows the English font name, and the ":" delimiter follows the Traditional Chinese font name.) Since there are spaces in the long font names, the font list is enclosed in quotation marks.

Traditional Chinese (zh_TW.BIG5) Font Lists

In the Asian Solaris 2.6 environment, a Traditional Chinese zh_TW.BIG5 font list is composed of one English font, representing ASCII characters, and one Traditional Chinese font representing Chinese characters in Big 5.

Traditional Chinese Solaris provides some default font lists defined in an application defaults file in `/usr/dt/app-defaults/zh_TW.BIG5/*`. Below is a part of one of the files, `Dtwm`:

```
Dtwm*icon*fontList: \  -dt-interface system-medium-r-normal-s*-*-*-*-*-*-*-*-*
```

This font list contains the following fonts, defined in `/usr/openwin/lib/locale/zh_TW.BIG5/X11/fonts/75dpi/fonts.alias`:

```
"-dt-interface system-medium-r-normal-s serif-16-140-75-75-p-70-big5-0"
"-dt-interface system-medium-r-normal-s serif-16-140-75-75-p-140-big5-1"
```

The first is an English font for ASCII character font display. The second is a Traditional Chinese Big 5 font.

Starting Applications with a Specific Traditional Chinese Font List

When you start an Asian Solaris tool at the command line, you can also specify its fonts. The following is an example of using a command line argument to start a new Traditional Chinese Windows terminal with a specified font list.

```
system% dtterm -fn "-dt-interface system-medium-r-normal-s \
serif-16-140-75-75-p-70-big5-0; \
-dt-interface system-medium-r-normal-s \
serif-16-140-75-75-p-140-big5-1:"
```

Note the two delimiters used in the font list. The “;” delimiter is used to separate the font names except for the last font name, which ends with the “:” delimiter. (In the example above, “;” follows the English font name, and the “:” delimiter follows the Traditional Chinese font name.) Since there are spaces in the long font names, the font list is enclosed in quotation marks.

Displaying PostScript Fonts

For screen displays, Asian Solaris software provides PostScript fonts through the Display PostScript System (DPS), as explained in Chapter 6.

CHAPTER 5

X Input Method Architecture

This chapter describes the input method architecture of X Window applications in Asian locales. The architecture is based on the X Window System Input Method Specification, which is included in X11R6 documentation, and on the X Window System Input Method Protocol (XIMP).

Overview of the X Window System Input Method (XIM)

The X Window System Input Method (XIM) is a mechanism to input internationalized text, that is, text using other than ASCII characters. Before the X Window System was internationalized, the mapping between keystrokes and input characters was a simple one-to-one mapping. However, the increased demand for internationalized programs forced the abandonment of simple keymapping. For example, mappings between key strokes and Chinese, Korean, or Japanese characters are very different.

Without internationalization, the relevant parts of the X Window system could be depicted as follows:

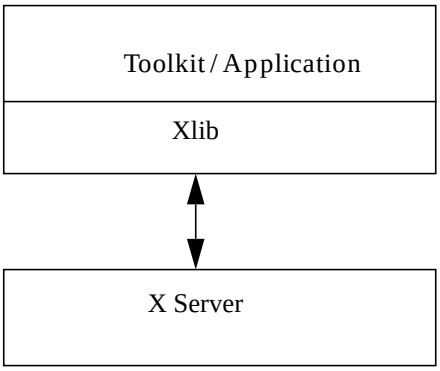

Figure 5–1 X Window system architecture—not internationalized

With internationalization, there is the added facility of an intelligent input method server, which can be either a library or server. Sun's X input method for Asian Solaris facilities is in the form of a server. This can be represented as follows:

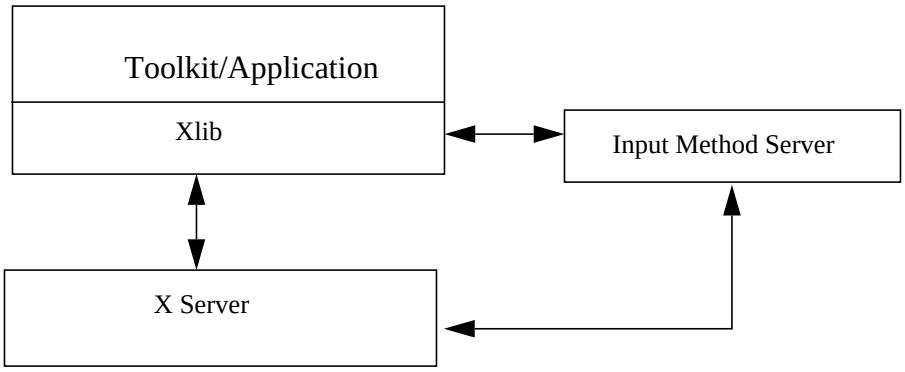

Figure 5–2 Solaris Input Method Server Role

X Window System internationalization involves more than just XIM. It also involves locale support and internationalized text drawing support, such as `XmbDrawString`() and `XwcDrawString`(). The X11R6 documentation provides more information on X Window System internationalization.

Architecture of XIM

This section describes how XIM works.

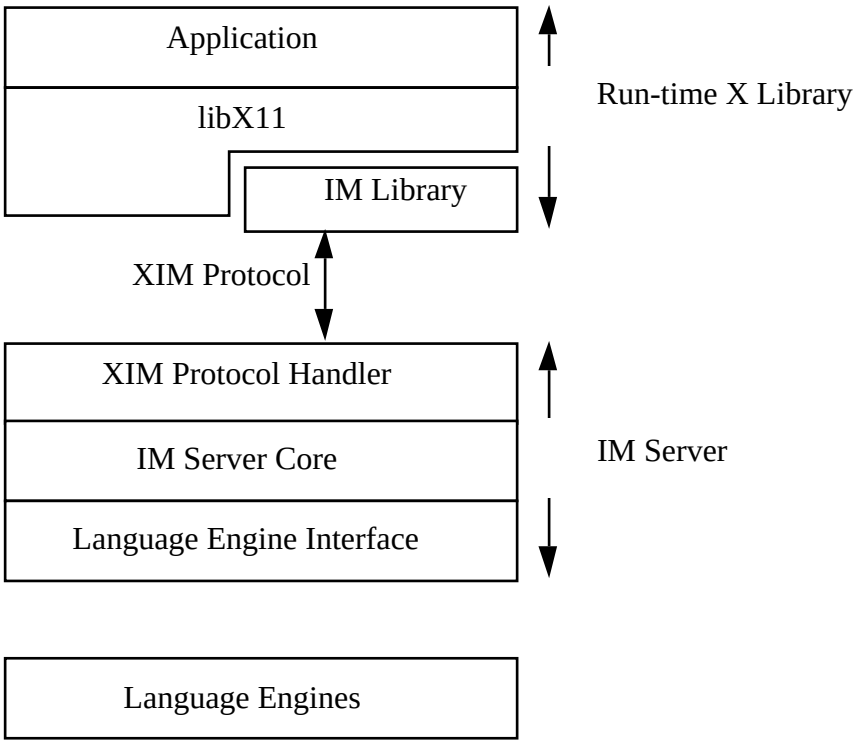

Figure 5–3 XIM Server Model

XIM Protocol Interface

The current XIMP in Solaris software supports the following key event handling solutions:

- `XIMP_FE_TYPE1` (default)
- `XIMP_SYNC_BE_TYPE2`

Front-End IM Server Type1: `XIMP_FE_TYPE1`

All key events go through the IM server only when input conversion mode is ON to compose text.

The Conversion ON key is recognized in the IM library.

The Conversion OFF key is recognized in the IM server.

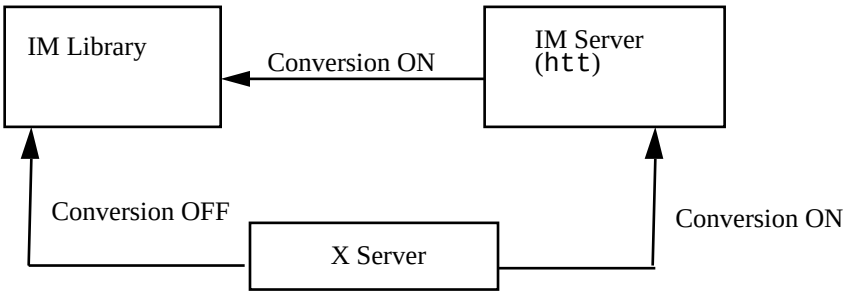

Figure 5–4 Event Flow in `XIMP_FE_TYPE1`

Back-End IM Server Type2: `XIMP_BE_TYPE2`

All key events are sent first to the IM library and then to the IM server. The IM server is responsible for sending unused key events back to the IM library.

The Conversion ON / OFF key is recognized in IM server.

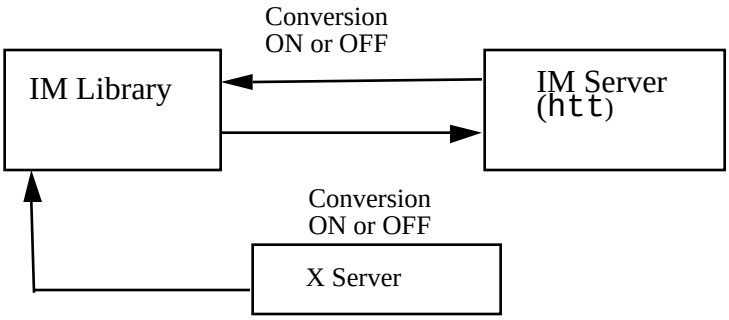

Figure 5–5 Event flow in `XIMP_BE_TYPE2`

For `XIMP_SYNC_BE_TYPE2`, extra synchronization is done on `XIMP_KEYPRESS`.

IM Server

The IM server is implemented as a separate process. The Sun IM server, named `htt`, supports the following Asian locales:

- `ko` and `ko.UTF-8` (Korean)
- `ja` and `ja_JP.PCK` (Japanese)
- `zh` (Simplified Chinese)
- `zh_TW` and `zh_TW.BIG5` (Traditional Chinese)

For Korean or Chinese `htt` usage and options, refer to the `htt`(1) `man` pages and to the input methods chapter in the Asian Solaris user's guide appropriate to your product.

The IM server comprises the following functional components:

1. XIMP

   XIMP is one of the most commonly used protocol interfaces between the IM server and the application side of the X library. Sun's implementation supports the latest version of XIMP, which is XIMP4.0.

2. Event Handling

   Event-handling routines handle all key events coming to the IM server directly from the X server.

3. Language Engine Interface

   `htt` communicates with language engines through its internal interface.

4. Rendering

   Rendering routines support the lookup choice region, non-callback status style, and various preedit area styles:

   - Lookup Choice Region

     Support for lookup choice region

   - Status Area Style

     Support for status callbacks (`XIMStatusCallbacks`)

     Support for geometry management of status (`XIMStatusArea`)

     Support for root window status (`XIMStatusNothing`)

   - Preedit Style

     Support for "on-the-spot" style preedit area (`XIMPreeditCallbacks`)

     Support for "over-the-spot" style preedit area (`XIMPreeditPosition`)

     Support for "off-the-spot" style preedit area (`XIMPreeditArea`)

     Support for "root window" style preedit area (`XIMPreeditNothing`)

Setting Status and Preedit Styles

The X Window System resource entries in the `.Xdefaults` resource file specify the status style and preedit style as described in the following sections.

The default CDE settings are `overTheSpot` and status area.

Status Styles

The Status Area displays which input mode is in effect in the IM server window. You can set the style by entering one of the following lines in the `.Xdefaults` resource file:

TABLE 5–1 Status Styles

Status Styles	X Resource
callback	N / A
status area	`preeditType: overTheSpot`
root window	`preeditType: root`

Preedit Styles

Many scripts require multiple key strokes to form a single display character, syllable, symbol, ideogram, or other item. Adequate space is needed to display input keystrokes for these languages. For the Asian CDE environment, this intermediate display area is called the *preedit area*. The preedit area displays intermediate formations of input keystrokes while they are held in a buffer before being committed and sent to the application.

The available styles of preedit area positioning are discussed in the following sections. You can set the style by placing one of the following lines in the `.Xdefaults` resource file:

TABLE 5–2 Preedit Styles

Preedit Style	X Resource
on-the-spot	N / A
over-the-spot	`*preeditType: overTheSpot`
off-the-spot	`*preeditType: offTheSpot`
root window	`*preeditType: root`

On-the-Spot Style

The on-the-spot style in the preedit area is displayed directly in the application window, and grows to the right of the insertion point as text is typed. If existing text is located to the right of the insertion point, that text is moved further to the right, accommodating the expanding preedit text as it is typed. The following shows an example of this style:

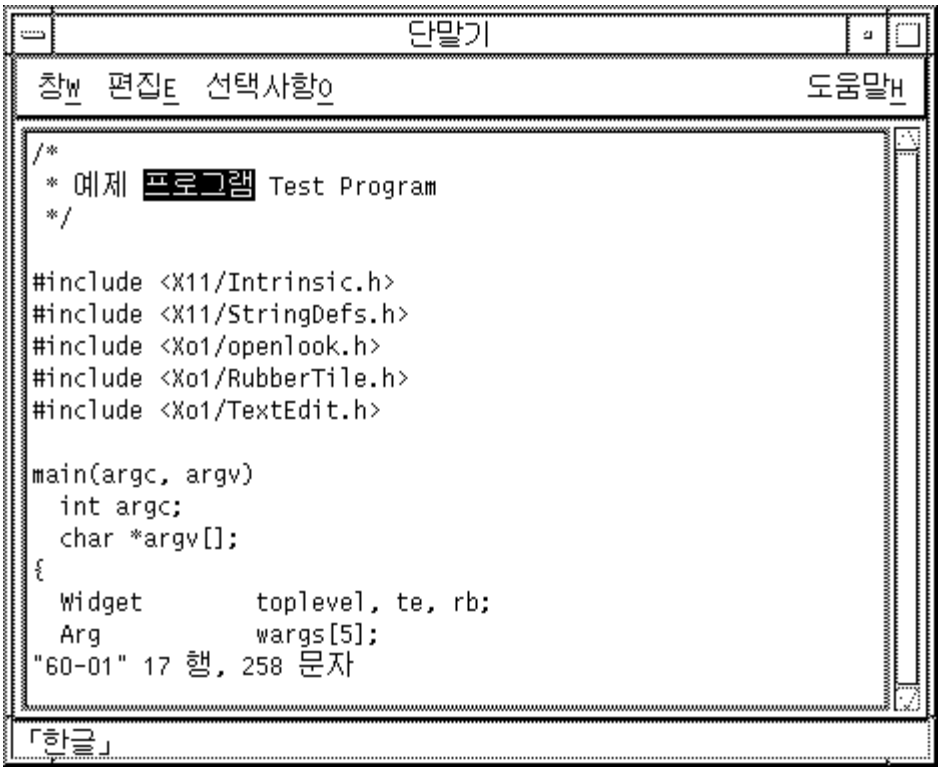

Figure 5–6 On-the-Spot Style: Korean

Figure 5–7 On-the-Spot Style: Simplified Chinese

Figure 5–8 On-the-Spot Style: Traditional Chinese

Over-the-Spot Style

As you type text, an over-the-spot style preedit area expands to the right of the insertion point and overwrites any existing display. After you finish typing and processing the preedit area input (in other words, once the text is converted, committed, or otherwise disposed), the processed preedit text is inserted at the insertion point and the preedit area disappears.

Existing text located to the right of the insertion point appears to be overwritten during typing, but when the processed text is inserted at the insertion point, previously existing text is moved and displayed further to the right. This text movement accommodates the input text being inserted.

Using Over-the-Spot Style with Korean Solaris Software

Before you type in an over-the-spot preedit area, the screen should resemble the following:

Figure 5–9 Over-the-Spot Style Before Korean Text is Typed

1. The following figure shows Korean text added on a line of existing text:

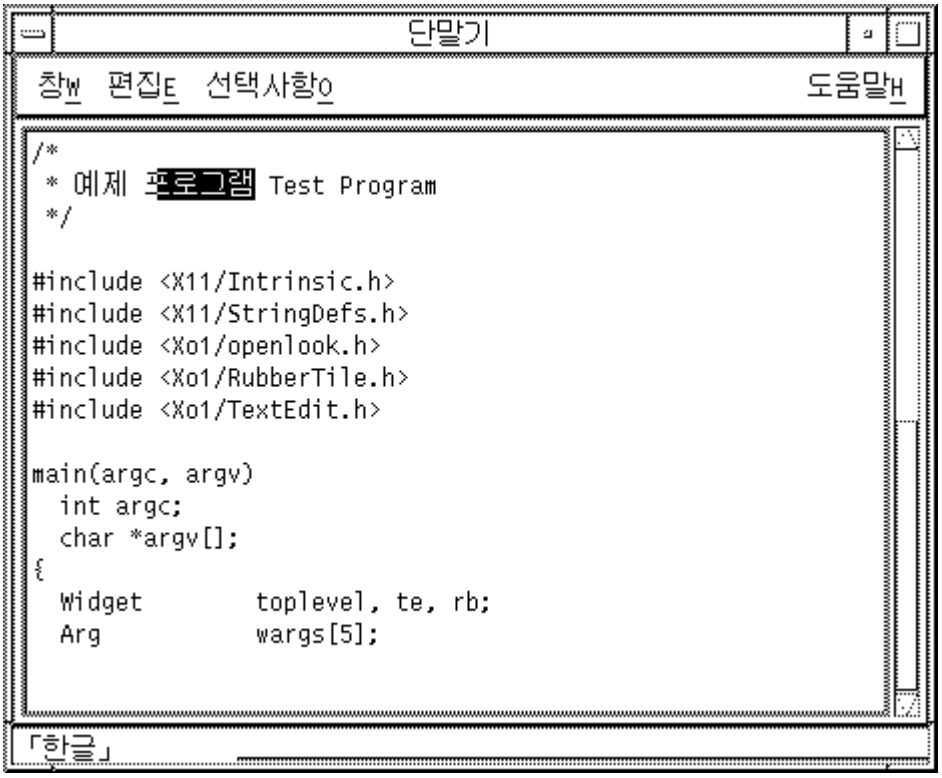

Figure 5–10 Over-the-Spot Style After Korean Input

2. The following figure shows Korean text after it is committed:

Figure 5–11 Over-the-Spot Style After Korean Text is Committed

Using Over-the-Spot Style with Simplified Chinese Solaris Software

Before you type text in an over-the-spot preedit area, the screen should resemble the following:

Figure 5–12 Over-the-Spot Style Before Typing Simplified Chinese Text

1. The following figure shows text added at the insertion point on a line of existing text:

Figure 5–13 Over-the-Spot Style After Typing Simplified Chinese Text

2. The following figure shows Simplified Chinese text after it is committed:

Figure 5–14 Over-the-Spot Style After Simplified Chinese Text is Committed

Using Over-the-Spot Style with Traditional Chinese Solaris Software

Before you type in an over-the-spot preedit area, the screen should resemble the following:

Figure 5–15 Over-the-Spot Style Before Typing Traditional Chinese Text

1. The following figure shows text added at the insertion point on a line of existing text:

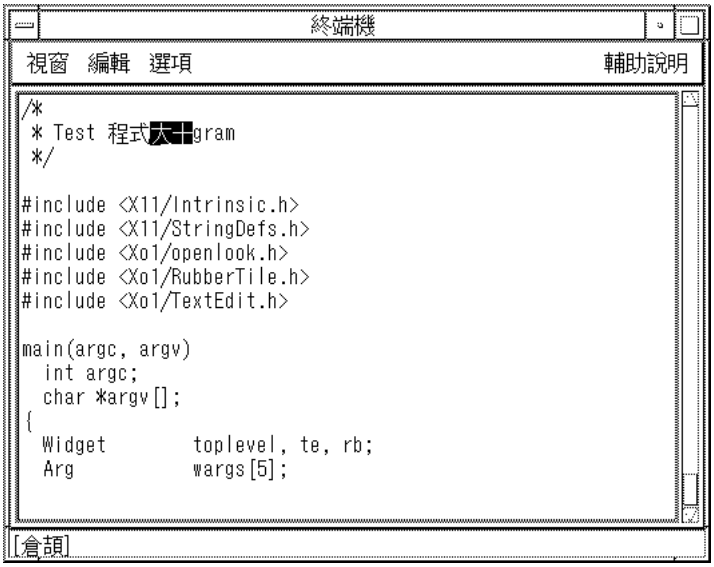

Figure 5–16 Over-the-Spot Style After Traditional Chinese Text is Typed

2. The following figure shows Traditional Chinese text after it is committed:

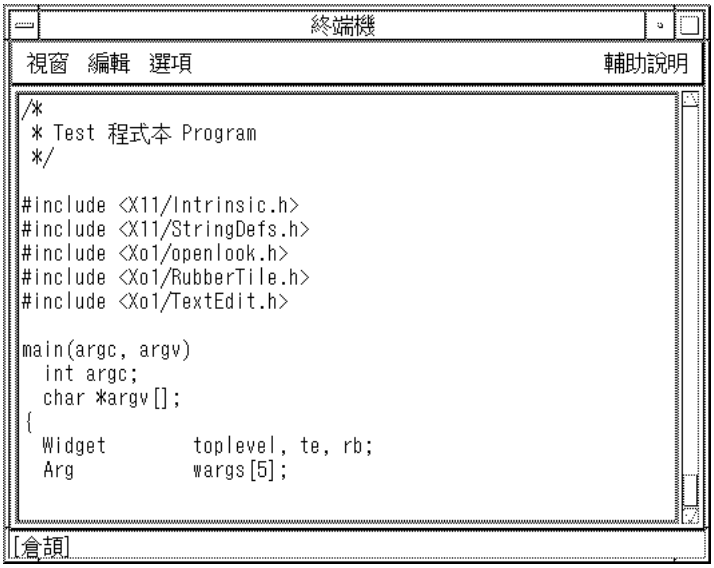

Figure 5–17 Over-the-Spot Style After Traditional Chinese Text is Committed

Off-the-Spot Style

An off-the-spot style preedit area is located inside an application window, but it is not at the insertion point. This style preedit area is a permanent part of the application's visual interface. For ease of viewing and consistency, the proposed

standard location for off-the-spot preedit areas is in the bottom margin of the application window to the right of the status area.

After you finish typing and processing preedit area input (text converted, committed, or otherwise disposed), the text is sent to the insertion point. Text to the right of the insertion point is moved farther to the right, accommodating the preedit text as it is inserted.

Root-Window Style

A root-window style preedit area is provided by the system's workspace. The proposed standard location of the preedit area is at the bottom of the workspace. Input handling works the same as with off-the-spot style preedit areas.

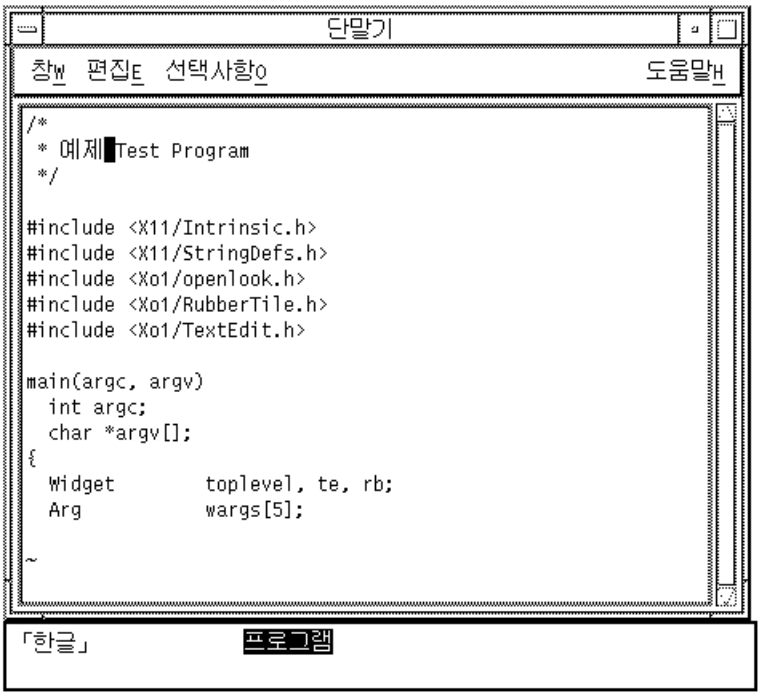

Figure 5–18 Root-Window Style (Korean)

Figure 5–19 Root-Window Style (Simplified Chinese)

Figure 5–20 Root-Window Style (Traditional Chinese)

Toolkit Use and Application Interfaces

Two methods are available for programming applications for Asian Solaris software:

1. Write an application that includes its own direct interface to Xlib.
2. Use Motif and other X toolkits that incorporate an XIM API in their interfaces to Xlib (usually the simpler method).

For more information, see *Common Desktop Environment: Internationalization Programmers Guide*. This guide provides instructions on writing international programs.

To write a program that interfaces directly with `Xlib`, a developer must use the internationalized APIs described earlier in this chapter. For additional information on the APIs, see the X11R6 documentation.

CHAPTER 6

Display PostScript System (DPS)

The Asian Solaris operating environment provides PostScript fonts in the Display PostScript System (DPS). This chapter describes what you need to use DPS in Asian Solaris software. For further details on DPS, see *Programming the Display PostScript System with X*, published by Adobe® Systems.

This chapter is grouped in three main sections for the three localized environments.

Using Korean PostScript Fonts and DPS Facilities

The Korean Solaris operating environment provides the Korean fonts listed in Table 6–1.

TABLE 6–1 Korean Solaris Operating Environment DPS PostScript Fonts

Font Name	Description
Kodig-Medium-COMB-H	Kodig-Medium font, 9/7 composite font encoding for horizontal display of EUC encoded Hangul and Roman text.
Kodig-Medium-COMB-V	Kodig-Medium font, 9/7 composite font encoding for vertical display of EUC encoded Hangul and Roman text.
Kodig-Medium	Kodig-Medium font, an alias of Kodig-Medium-EUC-H font; can be used like a Roman font.

TABLE 6–1 Korean Solaris Operating Environment DPS PostScript Fonts *(continued)*

Font Name	Description
Kodig-Medium-EUC-H	Kodig-Medium font, 9/7 composite font encoding for horizontal display of EUC text; can be used like a Roman font.
Kodig-Medium-EUC-V	Kodig-Medium font, 9/7 composite font encoding for vertical display of EUC text; can be used like a Roman font.
Kodig-Medium-H	Kodig-Medium font, 8/8 composite font encoding for horizontal display of shifted out ISO2022 text.
Kodig-Medium-V	Kodig-Medium font, 8/8 composite font encoding for vertical display of shifted out ISO2022 text.
Myeongjo-Medium-COMB-H	Myeongjo-Medium font, 9/7 composite font encoding for horizontal display of EUC encoded Hangul and Roman text.
Myeongjo-Medium-COMB-V	Myeongjo-Medium font, 9/7 composite font encoding for vertical display of EUC encoded Hangul and Roman text.
Myeongjo-Medium	Myeongjo-Medium font, an alias of Myeongjo-Medium-EUC-H font; can be used like a Roman font.
Myeongjo-Medium-EUC-H	Myeongjo-Medium font, 9/7 composite font encoding for horizontal display of EUC text; can be used like a Roman font.
Myeongjo-Medium-EUC-V	Myeongjo-Medium font, 9/7 composite font encoding for vertical display of EUC text; can be used like a Roman font.
Myeongjo-Medium-H	Myeongjo-Medium font, 8/8 composite font encoding for horizontal display of shifted out ISO2022 text.
Myeongjo-Medium-V	Myeongjo-Medium font, 8/8 composite font encoding for vertical display of shifted out ISO2022 text.

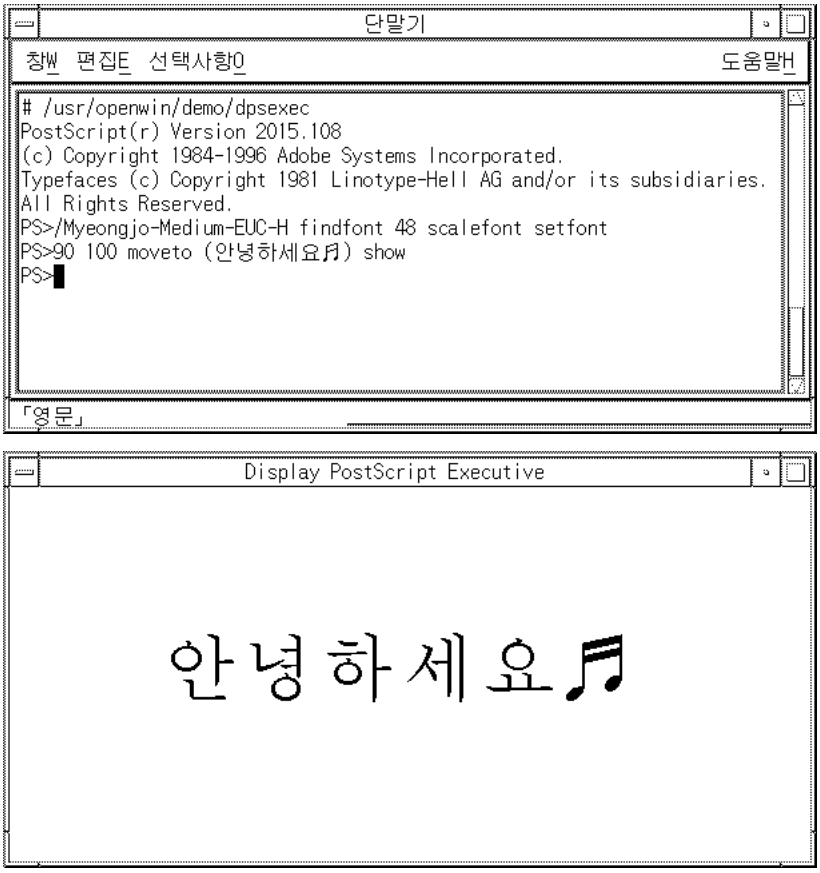

Figure 6–1 Display PostScript Output (Korean)

Of these fonts, you can use the following just as you would use Roman fonts:

- Kodig-Medium-EUC-H
- Kodig-Medium-EUC-V
- Kodig-Medium
- Myeongjo-Medium-EUC-H
- Myeongjo-Medium-EUC-V
- Myeongjo-Medium

Figure 6–1 shows a sample of Kodig-Medium and Myeongjo-Medium text.

You can also use the following Korean fonts just like Roman fonts for an ISO2022 encoded Hangul string, that is, for a pure Hangul string between SO and SI characters with no intermediate ASCII space (0x20) characters:

- Kodig-Medium-H
- Kodig-Medium-V

- Myeongjo-Medium-H
- Myeongjo-Medium-V

Creating Composite Fonts for Korean DPS

You can create composite fonts using any one Roman font with one of the following Korean fonts:

- Kodig-Medium-COMB-H
- Kodig-Medium-COMB-V
- Myeongjo-Medium-COMB-H
- Myeongjo-Medium-COMB-V

For example, the following PostScript code defines a composite font, Times-Italic+Kodig-Medium, which uses Times-Italic for ASCII characters and Kai-Medium horizontal font for Korean characters:

```
/Times-Italic+Kai-Medium
13 dict begin
         /FontName 1 index def
         /FMapType 4 def
         /Encoding [ 0 1 ] def
         /WMode 0 def
         /FontType 0 def
         /FontMatrix [1.0 0.0 0.0 1.0 0.0 0.0] def
         /FDepVector [
                 /Times-Italic findfont
                 /Kodig-Medium-COMB-H findfont
         ] def
 currentdict
 end
 definefont pop
```

Using Korean Fonts in DPS Programming

You can use some Korean fonts just as you use Roman fonts in DPS wrap definitions. The following sample code was used to create the display in Figure 6–2:

```
 defineps PSWDisplayText(char *text)
         /pointSize 50 def
         /Helvetica pointSize selectfont
         (Hello World) stringwidth pop 2 div neg 0 moveto
         (Hello World) show

         /kpSize 40 def
         /Kodig-Medium-KO kpSize selectfont
         (text) stringwidth pop 2 div neg pointSize neg moveto
         (text) show
 endps
```

You can call `PSWDisplayText(`*Korean text*`)` in a C program to display the designated Korean text; for example, as shown in Figure 6–2.

Figure 6–2 Solaris Operating Environment DPS (Korean)

Using Simplified Chinese PostScript Fonts and DPS Facilities

Simplified Chinese Solaris operating environment DPS provides the Simplified Chinese fonts listed in Table 6–2.

TABLE 6–2 Simplified Chinese Solaris Operating Environment PostScript Fonts

Font Name	Description
Song-Medium	Alias of Song-Medium-EUC; can be used like a Roman font
Song-Medium-EUC	Song-Medium font, EUC encoding, horizontal display; can be used like a Roman font
Song-Medium-H	Song-Medium font, horizontal display, to make a composite font with a Roman font

You can use the following two Simplified Chinese fonts just as you would use Roman fonts:

- Song-Medium
- Song-Medium-EUC

Figure 6–3 shows a sample of Song-Medium.

Figure 6–3 Sample Simplified Chinese Text Display PostScript Output

Creating Composite Roman and Simplified Chinese Fonts

You can create composite fonts using any one Roman font and the Simplified Chinese Song-Medium-H font. For example, the following PostScript code defines a composite font, Times-Italic+Song-Medium, which uses Times-Italic for ASCII characters and Song-Medium horizontal font for Simplified Chinese characters:

```
/Times-Italic+Song-Medium
13 dict begin
```

```
        /FontName 1 index def
        /FMapType 4 def
        /Encoding [ 0 1 ] def
        /WMode 0 def
        /FontType 0 def
        /FontMatrix [1.0 0.0 0.0 1.0 0.0 0.0] def
        /FDepVector [
                /Times-Italic findfont
                /Song-Medium-H findfont
        ] def
currentdict
end
definefont pop
```

Using Simplified Chinese Fonts in DPS Programming

You can use Simplified Chinese fonts just as you use Roman fonts in DPS wrap definitions. The following sample code creates the display in Figure 6–4:

```
defineps PSWDisplayText(char *text)
        /pointSize 50 def
        /Helvetica pointSize selectfont
        (Hello World) stringwidth pop 2 div neg 0 moveto
        (Hello World) show

        /cpSize 40 def
        /Song-Medium cpSize selectfont
        (text) stringwidth pop 2 div neg pointSize neg moveto
        (text) show
endps
```

You can call `PSWDisplayText(`*Chinese text*`)` in a C program to display the designated Chinese text. For an example see Figure 6–4.

Figure 6–4 Using Simplified Chinese Solaris Operating Environment DPS

Simplified Chinese Solaris software provides TrueType support in DPS.

Using Traditional Chinese PostScript Fonts and DPS Facilities

The Traditional Chinese Solaris operating environment provides the Chinese PostScript fonts listed in Table 6–3.

TABLE 6–3 Traditional Chinese Solaris Operating Environment PostScript Fonts

Font Name	Description
Kai-Medium	Alias of Kai-Medium-EUC-H.
Kai-Medium-EUC-H	Kai-Medium font, EUC encoding, horizontal display; can be used like a Roman font.
Kai-Medium-EUC-V	Kai-Medium font, EUC encoding, vertical display; can be used like a Roman font.
Kai-Medium-H	Kai-Medium font, horizontal display, to composite with a Roman font.

TABLE 6–3 Traditional Chinese Solaris Operating Environment PostScript Fonts *(continued)*

Font Name	Description
Kai-Medium-V	Kai-Medium font, vertical display, to composite with a Roman font.
Ming-Light	Alias of Ming-Light-EUC-H.
Ming-Light-EUC-H	Ming-Light font, EUC encoding, horizontal display; can be used like a Roman font.
Ming-Light-EUC-V	Ming-Light font, EUC encoding, vertical display; can be used like a Roman font.
Ming-Light-H	Ming-Light font, horizontal display, to composite with a Roman font.
Ming-Light-V	Ming-Light font, vertical display, to composite with a Roman font.

Of the fonts in this table, you can use the following just as you use Roman fonts:

- Kai-Medium
- Kai-Medium-EUC-H
- Kai-Medium-EUC-V
- Ming-Light
- Ming-Light-EUC-H
- Ming-Light-EUC-V

Figure 6–5 shows the use of Kai-Medium and Ming-Light.

Figure 6–5 Display PostScript Output Showing Traditional Chinese Text Sample

Creating Composite Roman and Traditional Chinese Fonts

You can create composite fonts using one Roman font and one of the following Traditional Chinese fonts:

- Ming-Light-H
- Ming-Light-V
- Kai-Medium-H
- Kai-Medium-V

For example, the following PostScript code defines a sample composite font, Times-Italic+Kai-Medium, which uses Times-Italic for ASCII characters and Song-Medium horizontal font for Traditional Chinese characters:

```
/Times-Italic+Kai-Medium
13 dict begin
        /FontName 1 index def
        /FMapType 4 def
        /Encoding [ 0 1 ] def
        /WMode 0 def
        /FontType 0 def
        /FontMatrix [1.0 0.0 0.0 1.0 0.0 0.0] def
        /FDepVector [
                /Times-Italic findfont
                /Kai-Medium-H findfont
        ] def
 currentdict
 end
 definefont pop
```

Using Traditional Chinese Fonts in DPS Programming

You can use Traditional Chinese fonts just as you use Roman fonts in DPS wrap definitions. The following sample code creates the display in Figure 6–6:

```
defineps PSWDisplayText(char *text)
        /pointSize 50 def
        /Helvetica pointSize selectfont
        (Hello World) stringwidth pop 2 div neg 0 moveto
        (Hello World) show

        /cpSize 40 def
        /Kai-Medium cpSize selectfont
        (text) stringwidth pop 2 div neg pointSize neg moveto
        (text) show
endps
```

You can call `PSWDisplayText(`*Chinese text*`)` in a C program to display the designated Chinese text; for example, as shown in Figure 6–6.

Figure 6–6 Using Traditional Chinese Solaris DPS

Simplified Chinese Solaris software provides TrueType support in DPS.

APPENDIX **A**

OpenWindows Information

This appendix contains information on developing applications for the OpenWindows environment.

System Environment

Asian OpenWindows `locale` Environment

Korean or Chinese locale-specific files are in the directory:

- `/usr/openwin/lib/locale/ko/` – for Korean Solaris software
- `/usr/openwin/lib/locale/zh/` – for Simplified Chinese Solaris software
- `/usr/openwin/lib/locale/zh_TW/` – for Traditional Chinese Solaris software

Window Message Files

Refer to *Solaris Internationalization Guide for Developers* for information on how the window system does messaging, how these messages are generated, and how to create message files for applications.

Korean Messages

Files in the `/usr/openwin/lib/locale/ko/LC_MESSAGES` directory with `.mo` filename extensions contain messages used in the Korean OpenWindows environment. They are created from portable message files (*xxx*`.po`).

Simplified Chinese Messages

Files in the `/usr/openwin/lib/locale/zh/LC_MESSAGES` directory with `.mo` filename extensions contain messages used in DeskSet tool windows. They are created from portable message files (*xxx*`.po`).

Traditional Chinese Messages

Files in the `/usr/openwin/lib/locale/zh_TW/LC_MESSAGES` directory with `.mo` filename extensions contain messages used in the Chinese OpenWindows environment.

`fonts` Directory

The `/usr/openwin/lib/locale/`*locale*`/X11/fonts` directory holds the Korean or Chinese fonts. (*locale* is `ko`, `zh`, or `zh_TW`). This directory must be in your font path before you can access Korean or Chinese fonts. The `openwin` script that comes with the Asian Solaris operating environment includes this directory in the font path. To add a different font directory path dynamically, type:

```
% xset +fp font_directory_path
```

`libs` Directory

The `/usr/openwin/lib/locale/`*locale*`/libs` directory is for locale-specific libraries (*locale* is `ko`, `zh`, or `zh_TW`). It contains only the `wdkind.so` library, which provides Korean- or Chinese-specific word selection capabilities.

`xview` Directory

The `/usr/openwin/lib/locale/`*locale*`/xview` directory currently contains two resource files for XView; `.text_extras_menu`, which defines the extra menu under text subwindow, and `defaults`, which defines some default values used by the XView library (*locale* is `ko`, `zh`, or `zh_TW`).

`print` Directory

The `/usr/openwin/lib/locale/`*locale*`/print` directory contains `ko`, `zh`, or `zh_TW` locale-specific printing files (*locale* is `ko`, `zh`, or `zh_TW`). Most notable is the `prolog.ps` file, which is used by the Korean or Chinese OpenWindows DeskSet tools that come with the Korean or Chinese Solaris operating environment. For information on using `prolog.ps`, see the printing facilities chapters in the Asian Solaris user's guide appropriate to your Solaris operating environment.

Font Information

Using Bitmap Fonts with XView Applications

To use your new Korean bitmap font with XView applications, you must add a font set definition in `/usr/openwin/lib/locale/ko/OW_FONT_SETS/OpenWindows.fs`. For example:

```
myfont:definition,\
 -sun-gothic-medium-r-normal--18-160-75-75-c-80-ksc5636-0,\
 -new-myfont-medium-r-normal--18-160-75-75-c-160-ksc5601.1987-0
```

To use your new Simplified Chinese bitmap font with XView applications, you must add a font set definition in `/usr/openwin/lib/locale/zh/OW_FONT_SETS/OpenWindows.fs`. For example:

```
myfont:definition,\
 -sun-song-medium-r-normal--18-160-75-75-c-80-iso8859-1,\
 -new-myfont-medium-r-normal--18-160-75-75-c-160-gb2312.1980-0
```

To use your new Traditional Chinese bitmap font with XView applications, you must add a font set definition in `/usr/openwin/lib/locale/zh_TW/OW_FONT_SETS/OpenWindows.fs`. For example:

```
myfont:definition,\
 -sun-sung-medium-r-normal--18-160-75-75-c-80-cns11643-0,\
 -sun-sung-medium-r-normal--18-160-75-75-c-160-cns11643-1,\
 -sun-sung-medium-r-normal--18-160-75-75-c-160-cns11643-2,\
 -sun-sung-medium-r-normal--18-160-75-75-c-160-cns11643-3,\
 -sun-sung-medium-r-normal--18-160-75-75-c-160-cns11643-4,\
 -sun-sung-medium-r-normal--18-160-75-75-c-160-cns11643-5,\
 -sun-sung-medium-r-normal--18-160-75-75-c-160-cns11643-6,\
 -sun-sung-medium-r-normal--18-160-75-75-c-160-cns11643-7,\
 -sun-sung-medium-r-normal--18-160-75-75-c-160-cns11643-15,\
 -new-myfont-medium-r-normal--18-160-75-75-c-160-cns11643-16
```

Make sure you use the correct English font size with the Asian fonts. Then type:

```
system% shelltool -font myfont
```

See “Font Sets” on page 66 for a description of font sets.

Font Sets

A *font set* is a collection of one or more fonts needed to display characters in a language with multiple character sets. The Asian Solaris font sets are described in the following subsections. The concept of font sets is described more fully in *XView Developer's Notes.*

The `/usr/openwin/lib/locale/`*locale*`/OW_FONT_SETS/OpenWindows.fs` (*locale* = `ko`, `zh`, `zh_TW`) file defines the full Korean or Chinese OpenWindows font set and also sets the following font point size definitions:

- small=12 pts.
- medium=14 pts.
- large=16 pts.
- extra-large=20 pts. (Korean) or 24 pts. (Chinese)

Korean Font Sets

In the Asian OpenWindows environment, a Korean font set is composed of one Roman font (presenting codeset 0 or ASCII characters in ISO8859 standard) and one Korean font (presenting codeset 1 characters in KS C 5601).

The Asian OpenWindows environment provides some default font sets defined in a Korean font set definition file:

`/usr/openwin/lib/locale/ko/OW_FONT_SETS/OpenWindows.fs`

The following is part of this file:

```
! (14 points, proportional)
-sun-kodig-medium-r-normal--16-140-75-75-p-140-korean-
0:definition,\
    -b&h-lucida-medium-r-normal-sans-0-0-0-0-p-0-iso8859-1,\
    -sun-gothic-medium-r-normal--16-140-75-75-c-140-ksc5601.1987-0
```

The following line in the file defines a font set as follows:

```
"-sun-kodig-medium-r-normal--16-140-75-75-p-140-korean-0"
```

The following line describes the Roman font for codeset 0 (ASCII) character font display:

```
"-b&h-lucida-medium-r-normal-sans-0-0-0-0-p-0-iso8859-1"
```

The following line describes the codeset 1 character display, along with the following:

```
"-sun-gothic-medium-r-normal--16-140-75-75-c-140-ksc5601.1987-0"
```

Note the following default font sets defined in the Korean OpenWindows environment:

```
-sun-kodig-medium-r-normal--14-120-75-75-c-120-korean-0
-sun-kodig-medium-r-normal--16-140-75-75-p-140-korean-0
```

```
-sun-kodig-medium-r-normal--16-140-75-75-c-140-korean-0
-sun-kodig-medium-r-normal--18-160-75-75-c-160-korean-0
-sun-kodig-bold-r-normal--16-140-75-75-p-140-korean-0
-sun-kodig-bold-r-normal--16-140-75-75-c-140-korean-0
-sun-kodig-bold-r-normal--18-160-75-75-c-160-korean-0
-sun-myeongjo-medium-r-normal--16-140-75-75-p-140-korean-0
-sun-myeongjo-medium-r-normal--16-140-75-75-c-140-korean-0
-sun-myeongjo-medium-r-normal--22-200-75-75-c-200-korean-0
-sun-myeongjo-medium-r-normal--26-240-75-75-c-240-korean-0
-sun-myeongjo-bold-r-normal--16-140-75-75-p-140-korean-0
-sun-myeongjo-bold-r-normal--16-140-75-75-c-140-korean-0
```

The font set name alias definitions are also provided in `/usr/openwin/lib/locale/ko/OW_FONT_SETS/OpenWindows.fs`.

An alias definition line in the file appears as follows:

```
kodig14: alias, -sun-kodig-medium-r-normal--16-140-75-75-c-140-korean-0
```

You can use `kodig14` instead of `-sun-kodig-medium-r-normal--16-140-75-75-c-140-korean-0` to access the font set.

Font Set	Alias
kodig12	`-sun-kodig-medium-r-normal--14-120-75-75-c-120-korean-0`
kodig14	`-sun-kodig-medium-r-normal--16-140-75-75-c-140-korean-0`
kodig16	`-sun-kodig-medium-r-normal--18-160-75-75-c-160-korean-0`
myeongjo14	`-sun-myeongjo-medium-r-normal--16-140-75-75-c-140-korean-0`
myeongjo16	`-daewoo-myeongjo-medium-r-normal--18-160-75-75-c-160-korean-0`
myeongjo20	`-sun-myeongjo-medium-r-normal--22-200-75-75-c-200-korean-0`
myeongjo24	`-sun-myeongjo-medium-r-normal--26-240-75-75-c-240-korean-0`

For complete definitions of these and other font sets, see the `/usr/openwin/lib/locale/kozh_TW/OW_FONT_SETS/OpenWindows.fs` file and to *XView Developer's Notes.*

Korean Scalable Fonts

The Korean Solaris scalable fonts are Kodig-Medium (alias Gothic-Medium) and Myeongjo-Medium (alias Myoungjo-Medium).

The Korean scalable fonts corresponding to KS C 5601-1987 can be also be accessed from the X side by calling `XCreateFontSet`. For example, the following code creates a font set consisting of scalable Lucida fonts scaled to 33 pixels, and scalable Kodig-Medium font scaled to 33 points.

```
fs = XCreateFontSet(display,
  ‘’-b&h-lucida-medium-r-normal-sans-33-*-*-*-p-*-iso8859-1,\
   -hanyang-kodig-medium-r-normal--*-330-*-*-m-*-ksc5601.1987-0",
      &missing_ptr, &missing_count, &def_string);
```

Starting Windows with a Specific Korean Font Set

A command line that starts an Asian OpenWindows tool can also specify its font. There are several ways you can specify which font set an Asian OpenWindows application uses. When starting an application tool, Asian OpenWindows checks font specifiers in command line arguments (e.g., `-Wt myeongjo14`, or `-font myeongjo14`, or `-fn myeongjo14`). If the font is not specified in the command line, but the shell variable `LANG` or `LC_CTYPE` is set to `ko`, then Asian OpenWindows uses the kodig14 font specification.

The following is an example of a command line argument used to start a new Korean Solaris command tool with a specified font:

```
system% cmdtool -Wt myeongjo14
```

The window display and UNIX processes might behave in a confusing manner if the font setting and locale are mismatched. However, when the current locale is U.S. (C, ASCII), the command uses a long XLED font name and cannot use a font set alias. Short and long font names are explained in the following section.

```
system% cmdtool -font -misc-fixed-medium-r-normal--9-80-100-100-c-
60-iso8859-1
```

Simplified Chinese Font Sets

In the Asian OpenWindows environment, a font set is composed of one Roman font (presenting codeset 0 or ASCII characters in ISO8859 standard) and one Chinese font (presenting codeset 1 characters in GB2312).

The Asian OpenWindows environment provides some default font sets defined in a Chinese font set definition file:

`/usr/openwin/lib/locale/zh/OW_FONT_SETS/OpenWindows.fs`

The following is part of this file:

```
! (14 points, proportional)
-sun-song-medium-r-normal--16-140-75-75-p-140-chinese-
0:definition,\
    -b&h-lucida-medium-r-normal-sans-0-0-0-0-p-0-iso8859-1,\
    -sun-song-medium-r-normal--16-140-75-75-c-140-gb2312.1980-0
```

The following line in the file defines a font set as follows:

```
"-sun-song-medium-r-normal--16-140-75-75-p-140-Chinese-0"
```

The following line describes the Roman font:

```
"-b&h-lucida-medium-r-normal-sans-0-0-0-0-p-0-iso8859-1"
```

The following line describes the codeset 0 (ASCII) character display, along with the following:

```
"-sun-song-medium-r-normal--16-140-75-75-c-140-gb2312.1980-0"
```

Note the following default font sets defined in the Chinese OpenWindows environment:

```
-sun-song-medium-r-normal--14-120-75-75-c-120-chinese-0
-sun-song-medium-r-normal--16-140-75-75-p-140-chinese-0
-sun-song-medium-r-normal--16-140-75-75-c-140-chinese-0
-sun-song-medium-r-normal--18-160-75-75-c-160-chinese-0
-sun-song-medium-r-normal--22-200-75-75-c-200-chinese-0
-sun-song-medium-r-normal--26-240-75-75-c-240-chinese-0
-sun-song-bold-r-normal--16-140-75-75-p-140-chinese-0
-sun-song-bold-r-normal--16-140-75-75-c-140-chinese-0
```

The font set name alias definitions are also provided in `/usr/openwin/lib/locale/zh/OW_FONT_SETS/OpenWindows.fs`.

An alias definition line in the file appears as follows:

```
song14: alias, -sun-song-medium-r-normal--16-140-75-75-c-140-chinese-0
```

You can use `song14` instead of `-sun-song-medium-r-normal--16-140-75-75-c-140-chinese-0` to access the font set.

Font Set	Aliases
song12	-sun-song-medium-r-normal--14-120-75-75-c-120-chinese-0
song14	-sun-song-medium-r-normal--16-140-75-75-c-140-chinese-0
song16	-sun-song-medium-r-normal--18-160-75-75-c-160-chinese-0
song20	-sun-song-medium-r-normal--22-200-75-75-c-200-chinese-0
song24	-sun-song-medium-r-normal--26-240-75-75-c-240-chinese-0

Font Set	Aliases
songb14	-sun-song-bold-r-normal--16-140-75-75-c-140-chinese-0
songb16	-sun-song-bold-r-normal--18-140-75-75-c-160-chinese-0

For complete definitions of these and other font sets, see the `/usr/openwin/lib/locale/zh/OW_FONT_SETS/OpenWindows.fs` file and to *XView Developer's Notes.*

Using a Simplified Chinese Font on the Command Line

A command line that starts a Chinese OpenWindows tool can also specify its font.

When the current locale is `zh`, the command should refer to a font-set name (explained in the following section), for example:

```
system% cmdtool -font song24
```

The flags `-font`, `-Wt`, and `-fn` all work the same in this kind of command. However, when the current locale is C (ASCII), the font name should refer only to the long XLFD name:

```
system% cmdtool -font -misc-fixed-medium-r-normal--9-80-100-100-c-60-iso8859-1
```

Traditional Chinese Font Sets

In the Traditional Chinese OpenWindows environment, a font set is composed of one Roman font (presenting codeset 0 or ASCII characters in ISO8859 standard) and nine Chinese fonts (presenting plane 1, 2, 3, 4, 5, 6, 7, 15, and 16 characters in CNS 11643).

Note - Asian Solaris software provides only plane 1, 2, and 3 characters.

The Asian OpenWindows environment provides some default font sets defined in a Chinese font set definition file:

`/usr/openwin/lib/locale/zh_TW/OW_FONT_SETS/OpenWindows.fs`

The following is part of this file:

```
! (14 points, proportional)
-sun-sung-medium-r-normal--16-140-75-75-p-140-tchinese-0:definition,\
    -b&h-lucida-medium-r-normal-sans-0-0-0-0-p-0-iso8859-1,\
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-1,\
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-2,\
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-3,\
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-4,\
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-5,\
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-6,\
```

```
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-7,\
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-15,\
    -sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-16
```

The following line describes the Roman font:

```
"-sun-sung-medium-r-normal--16-140-75-75-p-140-tchinese-0"
```

The following line describes the codeset 0 (ASCII) character display:

```
"-b&h-lucida-medium-r-normal-sans-0-0-0-0-p-0-iso8859-1"
```

The following line describes the plane 1 (codeset 1) Chinese character display:

```
"-sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-1"
```

The following line describes the plane 2 (codeset 2) Chinese character display:

```
"-sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-2"
```

The following line describes the plane 3 (codeset 2) Chinese character display:

```
"-sun-sung-medium-r-normal--16-140-75-75-c-140-cns11643-3"
```

Note the default font sets defined in the Asian OpenWindows environment:

```
-sun-sung-medium-r-normal--14-120-75-75-c-120-tchinese-0
-sun-sung-medium-r-normal--16-140-75-75-p-140-tchinese-0
-sun-sung-medium-r-normal--16-140-75-75-c-140-tchinese-0
-sun-sung-medium-r-normal--18-160-75-75-c-160-tchinese-0
-sun-sung-medium-r-normal--22-200-75-75-c-200-tchinese-0
-sun-sung-medium-r-normal--26-240-75-75-c-240-tchinese-0
-sun-sung-bold-r-normal--16-140-75-75-p-140-tchinese-0
-sun-sung-bold-r-normal--16-140-75-75-c-140-tchinese-0
```

The font set name alias definitions are also provided in `/usr/openwin/lib/locale/zh_TW/OW_FONT_SETS/OpenWindows.fs`.

The following is an alias definition line in the file:

```
sung14: alias, -sun-sung-medium-r-normal--16-140-75-75-c-140-tchinese-0
```

You can use `sung14` instead of `-sun-sung-medium-r-normal--16-140-75-75-c-140-tchinese-0` to access the font set.

TABLE A–1 Predefined Font Set Aliases

Font Set	Aliases
sung12	-sun-sung-medium-r-normal--14-120-75-75-c-120-tchinese-0
sung14	-sun-sung-medium-r-normal--16-140-75-75-c-140-tchinese-0
sung16	-sun-sung-medium-r-normal--18-160-75-75-c-160-tchinese-0
sung20	-sun-sung-medium-r-normal--22-200-75-75-c-200-tchinese-0
sung24	-sun-sung-medium-r-normal--26-240-75-75-c-240-tchinese-0

For the complete definitions of these and other font sets, see the `/usr/openwin/lib/locale/zh_TW/OW_FONT_SETS/OpenWindows.fs` file and *XView Developer's Notes*.

Traditional Chinese Scalable Fonts

The scalable fonts corresponding to CNS 11643 can be also be accessed from the X side by calling `XCreateFontSet`. For example, the following code creates a font set consisting of scalable Lucida fonts scaled to 33 pixels, and scalable Ming-Light Chinese fonts for planes 1, 2, and 3 scaled to 33 points.

```
fs = XCreateFontSet(display,
       "-b&h-lucida-medium-r-normal-sans-33-*-*-*-p-*-iso8859-1,\
       -dynalab-ming-light-r-normal--*-330-*-*-m-*-cns11643-1,\
       -dynalab-ming-light-r-normal--*-330-*-*-m-*-cns11643-2,\
       -dynalab-ming-light-r-normal--*-330-*-*-m-*-cns11643-3’’,
       &missing_ptr,
       &missing_count,
       &def_string);
```

Using a Traditional Chinese Font on the Command Line

A command line that starts an Asian OpenWindows tool can also specify its font.

When the current locale is `zh_TW`, the command should refer to a font-set name (explained in the following section), for example:

```
system% cmdtool -font sung24
```

The flags `-font`, `-Wt`, and `-fn` all work the same in this kind of command. However, when the current locale is C (ASCII), the font name should refer only to the long XLFD name:

```
system% cmdtool -font -misc-fixed-medium-r-normal--9-80-100-100-c- 60-iso8859-1
```

XIM

Setting Status and Preedit Styles

OpenWindows resource entries in the `.OWdefaults` resource file set the status area style and preedit area style as described in the following sections.

Default OpenWindows settings are `onTheSpot` and `imDisplaysInClient`.

Status Styles

The Status Area displays which input mode is in effect in the IM server window. You can specify the mode by adding one of the following lines to your `.OWdefaults` resource file:

```
OpenWindows.StatusStyle.locale: clientDisplays
OpenWindows.StatusStyle.locale: imDisplaysInClient
OpenWindows.StatusStyle.locale: imDisplaysInRoot
OpenWindows.StatusStyle.locale: none
```

Preedit Styles

Many languages require multiple key strokes to form a single display character, syllable, symbol, ideogram, or other item. The status area needs to display input keystrokes for these languages. For the Asian OpenWindows environment, this intermediate display area is called the *preedit area.* The preedit area displays intermediate formations of input keystrokes while they are held in a buffer before being committed, that is, before being sent to the insertion point in the text stream to the next program operation or the application.

The four available styles of preedit area positioning are discussed in the following sections. You can set the style by placing one of the following lines in the `.OWdefaults` resource file:

```
OpenWindows.PreeditStyle.locale: onTheSpot
OpenWindows.PreeditStyle.locale: overTheSpot
OpenWindows.PreeditStyle.locale: rootWindow
OpenWindows.PreeditStyle.locale: none
```

The preedit areas work the same internally; only their relation to the rest of the display differs.

APPENDIX **B**

Backward Compatibility Information

This appendix contains information for making programs backward-compatible with earlier version of Asian Solaris Software. Every utility described is supported, but for this version of Solaris, you are encouraged to use the XPG4 internationalization APIs as described in *Solaris Internationalization Guide for Developers*.

Asian Locale-Specific Utilities

These utilities test various aspects of the Korean or Chinese national standard character sets. Except Korean isksc, they also assume that the character being tested is part of the national standard character set:

- Korean: KS C 5601
- Traditional Chinese: CNS 11643
- Simplified Chinese: GB-2312-80

The arguments for the functions in these tables must be a character in WC, `wchar_t`. For more information, see the appropriate `man` page for your locale:

- Korean—`kctype`(3x)
- Simplified Chinese—`cctype`(3x)
- Traditional Chinese—`hctype`(3x)

TABLE B–1 Korean Character Classification Functions

Utility	Description
`isksc`	Returns true if it is in the KS C 5601 character set.
`iskroman`	Returns true if it is a Roman character as defined by the KS C 5636 character set.
`iskromannum`	Returns true if it is a Roman numeral symbol in the KS C 5601 character set.
`isksymbol`	Returns true if it is a Latin symbol or special character in the KS C 5601 character set.
`iskparen`	Returns true if it is a right or left parenthesis in the KS C 5601 character set.
`isklatin`	Returns true if it is a Latin letter character in the KS C 5601 character set.
`iskletter`	Returns true if it is a Korean vowel or consonant in the KS C 5601 character set.
`iskline`	Returns true if it is a ruled line symbol in the KS C 5601 character set.
`iskunit`	Returns true if is a unit character in KS C 5601.
`isksci`	Returns true if is a scientific symbol in KS C 5601.
`iskgen`	Returns true if it is a graphic or general symbol in the KS C 5601 character set.
`iskgreek`	Returns true if it is a Greek character in the KS C 5601 character set.
`iskrussian`	Returns true if it is a Russian character in the KS C 5601 character set.
`iskuser`	Returns true if the character is in the user-defined area of the KS C 5601 character set.
`iskhanja`	Returns true if it is an ideogram in KS C 5601.
`iskhangul`	Returns true if it is a Hangul phonogram in KS C 5601.

TABLE B–1 Korean Character Classification Functions *(continued)*

Utility	Description
`iskkata`	Returns true if it is a Japanese Katakana character in the KS C 5601 character set.
`iskhira`	Returns true if it is a Japanese Hiragana character in the KS C 5601 character set.

TABLE B–2 Simplified Chinese Character Classification Functions

Routine	Description
`ischanzi`	Returns true if it is a Hanzi ideogram in GB-2312-80.
`iscaccent`	Returns true if it is an accent notation in GB-2312-80.
`iscphonetic`	Returns true if it is a phonetic symbol in GB-2312-80.
`iscpinyin`	Returns true if it is a Pinyin symbol in GB-2312-80.
`iscalpha`	Returns true if it is a Roman alphabetic in GB-2312-80.
`iscdigit`	Returns true if it is a Roman digit in GB-2312-80.
`iscnumber`	Returns true if it is a number in GB-2312-80.
`isclower`	Returns true if it is a Roman lowercase in GB-2312-80.
`iscupper`	Returns true if it is a Roman uppercase in GB-2312-80.
`iscblank`	Returns true if it is a white space character from GB-2312-80.
`iscspace`	Returns true if it is a space character from GB-2312-80.
`iscgen`	Returns true if it is a graphic or general symbol in GB-2312-80.
`iscsci`	Returns true if it is a scientific symbol in GB-2312-80.

TABLE B–2 Simplified Chinese Character Classification Functions *(continued)*

Routine	Description
`iscline`	Returns true if it is a ruled line symbol in GB-2312-80.
`iscunit`	Returns true if it is a unit character in GB-2312-80.
`iscparen`	Returns true if it is a right or left parenthesis in GB-2312-80.
`iscpunct`	Returns true if it is a punctuation character in GB-2312-80.
`iscgreek`	Returns true if it is a Greek character in GB-2312-80.
`iscrussian`	Returns true if it is a Russian character in GB-2312-80.
`iscspecial`	Returns true if it is a Greek or Russian character in GB-2312-80.
`ischira`	Returns true if it is a Japanese Hiragana character in GB-2312-80.
`isckata`	Returns true if it is a Japanese Katakana character in GB-2312-80.

For Simplified Chinese, two additional routines, `iscgb` and `isceuc` (Table B–3), test for characters from the GB-2312-80 character set. The `iscgb` routine expects a wide character, and `isceuc` expects a GB-2312-80 character in EUC format. For more information, see the `cctype`(3x) `man` page.

TABLE B–3 General Simplified Chinese General Character Classification Functions

Routine	Description
`iscgb`	Returns true if it is in GB-2312-80.
`isceuc`	Returns true if it is a GB-2312-80 character in EUC format.

TABLE B–4 Traditional Chinese Character Classification Functions

Utility	Description
`ishalpha`	Returns true if it is a Roman character in the CNS 11643 character set.
`ishupper`	Returns true if it is an uppercase Roman character as defined by the CNS 11643 character set.
`ishlower`	Returns true if it is a lower case Roman character in the CNS 11643 character set.
`ishdigit`	Returns true if it is a number in the CNS 11643 character set.
`ishspace`	Returns true if it is the space character in the CNS 11643 character set.
`ishpunct`	Returns true if it is a punctuation character in the CNS 11643 character set.
`ishparen`	Returns true if it is a right or left parenthesis in the CNS 11643 character set.
`ishphontone`	Returns true if is a Mandarin phonetic tone.
`ishradical`	Returns true if is a Chinese character radical.
`ishline`	Returns true if it is a ruled line symbol in the CNS 11643 character set.
`ishunit`	Returns true if it is a unit character in the CNS 11643 character set.
`ishsci`	Returns true if it is a scientific symbol in the CNS 11643 character set.
`ishgen`	Returns true if it is a general symbol in the CNS 11643 character set.
`ishgreek`	Returns true if it is a Greek character in CNS 11643 character set.

Asian-Specific Utilities

This section describes functions for wide character and string input and output, character classification, and conversion functions for the Korean or Chinese character

sets. Asian Solaris software implements a wide character library for handling Korean or Chinese character codes according to industry standards.

Routines that have Korean or Chinese language-specific dependency are in their own language-specific library, which is linked with the corresponding C compiler option:

- Korean Solaris `libkle` is linked with `-lkle`
- Simplified Chinese Solaris `libcle` is linked with `-lcle`
- Traditional Chinese Solaris `libhle` is linked with `-lhle`

Refer to the appropriate `man` pages for more information.

Asian Solaris software defines WC as a constant-width, four-byte code. WC uses the ANSI C data type `wchar_t`, which Solaris software defines in `wchar.h` as follows:

```
typedef long wchar_h;
```

In Solaris software, `long` is four bytes.

Conversion Utilities

The conversion functions described in this section are available, but you should use `iconv( )` as a standard function.

Asian Solaris software provides facilities for various conversions, for example:

- Characters within a codeset, such as converting uppercase ASCII to lowercase.
- Between different conventions for national standard character sets, such as:
  - Between Combination and Completion code, both KS C 5601-1987 and KS C 5601-1992.
  - Between GB and EUC.
  - Between CNS 11643 code and Big5.
- Between code formats (such as converting between EUC and WC).

Programs using the general multibyte conversion utilities should include the header files `widec.h` and `wctype.h`.

- Korean Solaris specific routines (such as isk*xxx*) are declared in `ko/xctype.h`.
- Simplified Chinese Solaris specific routines (such as isc*xxx*) are declared in `zh/xctype.h`.
- Traditional Chinese Solaris specific routines (such as ish*xxx*) are declared in `zh_TW/xctype.h`.

Programs using general multibyte conversion utilities should include three header files: `wctype.h`, `widec.h`, plus one of the following locale-specific files:

- `ko/xctype.h` (Korean)

- `zh/xctype.h` (Simplified Chinese)
- `zh_TW/xctype.h` (Traditional Chinese)

The *locale*`/xctype.h` file declares the Korean or Chinese locale-specific routines, which have names of the form:

- isk*xxxx* (Korean)
- isc*xxxx* (Simplified Chinese)
- ish*xxxx* (Traditional Chinese)

As with classification functions described in the previous section, the use of these previously mentioned functions can be controlled by the `setlocale` function (described elsewhere in this and other chapters).

Locale-specific conversion routines (such as Korean `comptopack` or Chinese `cgbtoeuc` are contained in a locale-specific library:

- `libkle` (Korean)
- `libcle` (Simplified Chinese)
- `libhle` (Traditional Chinese)

This library can be linked during compilation using the C compiler option:

- `–lkle` (Korean)
- `–lcle` (Simplified Chinese)
- `–lhle` (Traditional Chinese)

Conversion Within a Codeset

The multibyte conversion functions are similar to the one-byte conversion functions `toupper` and `tolower`. These functions convert wide-characters to other wide characters. For more information on conversion routines, see the `man` pages for `wconv`(3) for all locales and for:

- `kconv`(3)—Korean
- `cconv`(3)—Simplified Chinese
- `hconv`(3)—Traditional Chinese

The following routines are in the regular Chinese C library:

TABLE B–5 Simplified Chinese Case Conversion Functions (declared in `zh/xctype.h`)

Function	Description
`tocupper`	Converts codeset 1 Roman lowercase to uppercase
`toclower`	Converts codeset 1 Roman uppercase to lowercase

TABLE B–6 Traditional Chinese Case Conversion Functions (declared in `zh_TW/xctype.h`)

Function	Description
`tohupper`	Converts codeset 1 Roman lowercase to uppercase
`tohlower`	Converts codeset 1 Roman uppercase to lowercase

Conversion Between Simplified Chinese Codesets

In the Simplified Chinese character sets, the Roman characters and numbers in codeset 0 are repeated in codeset 1. The following functions test wide characters.

TABLE B–7 Simplified Chinese Codeset Conversion Functions

Function	Description
`atocgb`	Converts alphabetic or numeric characters in ASCII (codeset 0) to the corresponding characters in GB-2312-80 (codeset 1).
`cgbtoa`	Converts alphabetic or numeric characters in GB-2312-80 (codeset 1) to the corresponding characters in ASCII (codeset 0).

For further information on these functions, see the `man` page for `cconv( )`(3x).

Conversion for Korean Character Codes

The following routines perform character-based code conversion on the KS C 5601 character set. They convert characters between Completion code (or EUC format) and Combination code (or Packed code).

To use these routines, the library `kle` must be linked using the C compiler option `–lkle`. For more information, see the `kconv`(3x) `man` page.

TABLE B–8 Korean Code Conversion Functions

Routine	Description
`comptopack`	Converts a character in Completion code to Combination (Packed) code of KS C 5601-1987.
`packtocomp`	Converts a character in Combination (Packed) code of KS C 5601-1987 to Completion code.
`wansuntojohap`	Converts a character in Completion code to Combination (Packed) code of KS C 5601-1992.
`packtocomp`	Converts a character in Combination (Packed) code of KS C 5601-1992 to Completion code .

Conversion for Simplified Chinese Character Codes

The following routines do character-based code conversion on the GB-2312-80 character set. They convert characters and strings between EUC format and GB-2312-80 format. To use these routines, the library lib`cle` must be linked using the C compiler option `–lcle`. For further information, see the `cconv`(3x) man page.

TABLE B–9 Simplified Chinese Character-Based Functions

Function	Description
`cgbtoeuc`	Converts a character in GB-2312-80 format (7 bit) to EUC format
`scgbtoeuc`	Converts a string in GB-2312-80 format (7 bit) to EUC format
`sncgbtoeuc`	Converts part of a string in GB-2312-80 format (7 bit) to EUC format

TABLE B–9 Simplified Chinese Character-Based Functions *(continued)*

Function	Description
`euctocgb`	Converts a character in EUC format to GB-2312-80 format (7 bit)
`seuctocgb`	Converts a string in EUC format in GB-2312-80 format (7 bit)
`sneuctocgb`	Converts a part of a string in EUC to GB-2312-80 format (7 bit)

Conversion for Traditional Chinese Character Codes

The following routines perform character-based code conversion on the CNS-11643 character set. They convert CNS-11643 characters between CNS-11643, EUC, and Big5 formats. To use these routines, the library `hle` must be linked using the C compiler option `–lhle`. For more information, see the `hconv`(3x) `man` page.

TABLE B–10 Traditional Chinese Character-Based Functions

Function	Description
`cbig5toeuc`	Converts Big5 character to EUC character.
`ccnstoeuc`	Converts CNS character to EUC character.
`ceuctobig5`	Converts EUC character to Big5 character.
`ceuctocns`	Converts EUC character to CNS character.

TABLE B–11 Traditional Chinese String-Based Functions

Function	Description
`big5toeuc`	Converts Big5 string to EUC string.
`cnstoeuc`	Converts CNS string to EUC string.
`euctobig5`	Converts EUC string to Big5 string.
`euctocns`	Converts EUC string to Big5 string.

Index