JFP Reference Manual 1 : User Commands

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–0657–10
December 2002

021225@5115

Contents

Preface

Both novice users and those familar with the SunOS operating system can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Overview

The following contains a brief description of each man page section and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character-set tables.
- Section 6 contains available games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.

- Section 9 provides reference information needed to write device drivers in the kernel environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).
- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer can include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man`(1) for more information about man pages in general.

NAME	This section gives the names of the commands or functions documented, followed by a brief description of what they do.
SYNOPSIS	This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required. The following special characters are used in this section: [ ] Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified. . . . Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "`filename` . . ." . | Separator. Only one of the arguments separated by this character can be specified at a time. { } Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

PROTOCOL	This section occurs only in subsection 3R to indicate the protocol description file.
DESCRIPTION	This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, and functions are described under USAGE.
IOCTL	This section appears on pages in Section 7 only. Only the device class that supplies appropriate parameters to the `ioctl`(2) system call is called `ioctl` and generates its own heading. `ioctl` calls for a specific device are listed alphabetically (on the man page for that specific device). `ioctl` calls are used for a particular class of devices all of which have an `io` ending, such as `mtio`(7I).
OPTIONS	This secton lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.
OPERANDS	This section lists the command operands and describes how they affect the actions of the command.
OUTPUT	This section describes the output – standard output, standard error, or output files – generated by the command.
RETURN VALUES	If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.
ERRORS	On failure, most functions place an error code in the global variable `errno` indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than

	one condition can cause the same error, each condition is described in a separate paragraph under the error code.
USAGE	This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality: Commands Modifiers Variables Expressions Input Grammar
EXAMPLES	This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as `example%`, or if the user must be superuser, `example#`. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections.
ENVIRONMENT VARIABLES	This section lists any environment variables that the command or function affects, followed by a brief description of the effect.
EXIT STATUS	This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions.
FILES	This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.
ATTRIBUTES	This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes`(5) for more information.
SEE ALSO	This section lists references to other man pages, in-house documentation, and outside publications.

DIAGNOSTICS	This section lists diagnostic messages with a brief explanation of the condition causing the error.
WARNINGS	This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.
NOTES	This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.
BUGS	This section describes known bugs and, wherever possible, suggests workarounds.

JFP Reference Manual 1 : User Commands

NAME

Intro_jfp, intro_jfp – introduction to JFP commands and application programs

AVAILABILITY

This section indicates which package contains the commands being described on this page. To be able to use the command, the indicated package must have been installed with the operating system. For information on how to add a package see `pkgadd`(1).

DESCRIPTION

This section describes, in alphabetical order, JFP commands available with this operating system.

OTHER SECTIONS

See these sections of the *JFP Reference Manual* for more information.

- Section 1M in this manual for JFP system maintenance commands.
- Section 4 of this manual for information on JFP file formats.
- Section 5 of this manual for descriptions of publicly available JFP files and miscellaneous information pages.

Manual Page Command Syntax

Unless otherwise noted, commands described in the SYNOPSIS section of a manual page accept options and other arguments according to the following syntax and should be interpreted as explained below.

name [–*option*...] [*cmdarg*...] where:

[ ]	Surround an *option* or *cmdarg* that is not required.
...	Indicates multiple occurrences of the *option* or *cmdarg*.
name	The name of an executable file.
{ }	The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.
option	(Always preceded by a ''–''.) *noargletter*... or, *argletter optarg*[,...]
noargletter	A single letter representing an option without an option-argument. Note that more than one *noargletter* option can be grouped after one ''–'' (Rule 5, below).
argletter	A single letter representing an option requiring an option-argument.
optarg	An option-argument (character string) satisfying a preceding *argletter*. Note that groups of *optargs* following an *argletter* must be separated by commas, or separated by a tab or space character and quoted (Rule 8, below).
cmdarg	Path name (or other command argument) *not* beginning with ''–'', or ''–'' by itself indicating the standard input.

Command Syntax Standard: Rules

These command syntax rules are not followed by all current commands, but all new commands will obey them. `getopts`(1) should be used by all shell procedures to parse positional parameters and to check for legal options. It supports Rules 3-10 below. The enforcement of the other rules must be done by the command itself.

1. Command names (*name* above) must be between two and nine characters long.
2. Command names must include only lower-case letters and digits.
3. Option names (*option* above) must be one character long.
4. All options must be preceded by ''–''.
5. Options with no arguments may be grouped after a single ''–''.
6. The first option-argument (*optarg* above) following an option must be preceded by a tab or space character.
7. Option-arguments cannot be optional.
8. Groups of option-arguments following an option must either be separated by commas or separated by tab or space character and quoted (–o xxx,z,yy or –o "xxx z yy").
9. All options must precede operands (*cmdarg* above) on the command line.
10. ''—'' may be used to indicate the end of the options.
11. The order of the options relative to one another should not matter.
12. The relative order of the operands (*cmdarg* above) may affect their significance in ways determined by the command with which they appear.
13. ''–'' preceded and followed by a space character should only be used to mean standard input.

ATTRIBUTES

See `attributes`(5) for a discussion of the attributes listed in this section.

SEE ALSO

`getopts`(1), `wait`(1), `exit`(2), `getopt`(3C), `attributes`(5)

DIAGNOSTICS

Upon termination, each command returns two bytes of status, one supplied by the system and giving the cause for termination, and (in the case of ''normal'' termination) one supplied by the program [see `exit`(2)]. The former byte is 0 for normal termination; the latter is customarily 0 for successful execution and non-zero to indicate troubles such as erroneous parameters, or bad or inaccessible data. It is called variously ''exit code'', ''exit status'', or ''return code'', and is described only where special conventions are involved.

WARNINGS

Some commands produce unexpected results when processing files containing null characters. These commands often treat text input lines as strings and therefore become confused upon encountering a null character (the string terminator) within a line.

LIST OF COMMANDS

Name	Description
`Intro_jfp`(1)	introduction to JFP commands and application programs
`atok12`(1)	ATOK12 Japanese language input system
`atok12migd`(1)	Merges an ATOK8 dictionary to ATOK12 dictionary
`atok12migs`(1)	Migrates the style setting from ATOK8 to ATOK12

atok12setup(1)	Set up ATOK12 for Japanese input in X environment
cs00toatok(1)	conversion cs00 user dictionary to ATOK user dictionary
euctoibmj(1)	Code conversion between Japanese EUC and IBM-Japanese
euctojis(1)	See jistoeuc(1)
euctosj(1)	See jistoeuc(1)
evftobdf(1)	convert evfont file to BDF format
ibmjtoeuc(1)	See euctoibmj(1)
jistoeuc(1)	Code conversion between JIS, PC kanji, and Japanese EUC
jistosj(1)	See jistoeuc(1)
jpostprint(1)	PostScript translator for Japanese text files
jprconv(1)	Filter for printing Japanese text on a dot-matrix Kanji printer or Japanese language page printer
jtops(1)	postscript filter for printing Japanese characters on Sun Laser Writer or Japanese postscript printer
jtty(1)	set Japanese terminal characteristics
kanji(1)	show the list of Kanji codes
kkcvtocs00(1)	conversion from kkcv user dictionary to cs00 user directory
sdtudctool(1)	Solaris gaiji tool
sdtudc_convert(1)	User defined character conversion utility
sdtudc_extract(1)	User defined character conversion utility
sdtudc_extract_ps(1)	User defined character conversion utility
sdtudc_register(1)	Intermediate utility to register user-defined characters
sjtoeuc(1)	See jistoeuc(1)
sjtojis(1)	See jistoeuc(1)
uum(1)	Kana-Kanji conversion front end processor
Wnn6(1)	Wnn6 Japanese language input system
wnn6setup(1)	Set up Wnn6 for Japanese input in X environment
wnnatod(1)	Convert an EUC text dictionary to a binary dictionary
wnnbushu(1)	Wnn6 radical input utility

wnndictutil(1)	Dictionary utility
wnndtoa(1)	Convert a binary dictionary to an EUC text dictionary
wnnenvutil(1)	Environment setting utility
wnnotow(1)	User dictionary converter
wnnstat(1)	Print the status of Wnn6 Kana-Kanji conversion server
wnntouch(1)	Rewrite and format the file header according to the inode.
xjsi(1)	Wnn6 Kana-Kanji conversion server/htt interface module

NAME

atok12 – ATOK12 Japanese language input system

DESCRIPTION

ATOK12 provides a method to input Japanese language in a desktop environment.

You can change the setup for ATOK12 by running `atok12setup`(1). When you are logged in to the Japanese desktop again, ATOK12 becomes available and ATOK palette window will open.

You can choose any Japanese language input method you like by using environment configuration utility accessable from ATOK palette window.

You can also register and use your own Kana-Kanji conversion dictionary by using dictionary utility accessable from ATOK palette window.

ATOK12 is available from X Window applications, and client applications that uses IIIMP (Internet-Internet Input Method Protocol), such as Java2 applications with Swing interface, for example.

Dictionaries for Kana-Kanji conversion can be managed centrally on a server.

SEE ALSO

`atok12setup`(1), `java`(1)

Japanese Input System Summary & Transition

ATOK12 User's Guide

NAME

atok12migd – Merges an ATOK8 dictionary to ATOK12 dictionary

SYNOPSIS

`atok12migd` `[-h` *part_of_speech*`]` *atok8_dic atok12_dic*

AVAILABILITY

JSatsvu

DESCRIPTION

`atok12migd` merges the contents of an ATOK8 dictionaries with those of an ATOK12 dictionary. `atok12migd` merges only user-registered words with the ATOK8 dictionaries. ATOK8 system-provided words are not merged. The contents of a source ATOK8 dictionary is neither destroyed nor modified through merge processing.

OPTIONS

`-h` *part_of_speech* Specifies the part_of_speech. The words to be merged can be limited by specifying the par_of_speech. Specify the part_of_speech using ATOK8's part_of_speech number. When specifying multiple parts_of_speech, delimit them with a plus sign (+). When "`all`" is specified, all parts_of_speech are merged.

(Example)

```
1
2+3
3+4+8+9
all
```

ATOK8 Parts of Speech Number

Number	Name
1	General nouns
2	Proper nouns (person name)
3	Proper nouns (location name)
4	Proper nouns (organization name)
5	Proper nouns (general)
6	Nouns (sa-gyou irregular)
7	Nouns (za-gyou irregular)
8	Nouns (adjective verbs)
9	Independent words
10	Tankanji (single Kanji)
11	Rentaishi (non-conjugative adjectives)

Number	Name
12	Conjunctions
13	Interjections
14	Prefixes
15	Noun suffixes
16	Numerals
17	Ka-gyou godan katsuyou (consonant-stem) verbs
18	Ga-gyou godan katsuyou (consonant-stem) verbs
19	Sa-gyou godan katsuyou (consonant-stem) verbs
20	Ta-gyou godan katsuyou (consonant-stem) verbs
21	Na-gyou godan katsuyou (consonant-stem) verbs
22	Ha-gyou godan katsuyou (consonant-stem) verbs
23	Ba-gyou godan katsuyou (consonant-stem) verbs
24	Ma-gyou godan katsuyou (consonant-stem) verbs
25	Ra-gyou godan katsuyou (consonant-stem) verbs
26	Wa-gyou godan katsuyou (consonant-stem) verbs
27	Ichidan katsuyou verbs
28	Ka-gyou irregular verbs
29	Sa-gyou irregular verbs
30	Za-gyou irregular verbs
31	Adjectives
32	Adjective verbs
33	Adverbs

atok8_dic ATOK8's dictionary.

atok12_dic ATOK12's dictionary

ENVIRONMENT VARIABLES

`LANG`, `LC_CTYPE` `atok12migd` operates only when the `LC_CTYPE` category is set to locale `ja` or an equivalent locale. For usage of the above environment variables, see the `environ`(5) man page.

EXIT STATUS

This command returns the following exit codes:

`0` Successful completion

`1` Unknown option

`127` Other errors

FILES

`/usr/bin/atok12migd` ATOK8 to ATOK12 dictionary merge command

SEE ALSO

`environ`(5)

Japanese Input System Summary & Transition

ATOK12 User's Guide

NOTES

This command ends when `SIGTERM` is sent to the process.

`atok12migd` cannot handle files whose name or contents contain characters of EUC code set 3.

NAME

atok12migs – Migrates the style setting from ATOK8 to ATOK12

SYNOPSIS

atok12migs [-k | -r] *atok8_ucf atok12_sty*

AVAILABILITY

JSatsvu

DESCRIPTION

The atok12migs command extracts the information of function-to-key bindings and Romaji-to-Kana mappings from ATOK8 configurtaion file *atok8_ucf*, converts it into ATOK12-style file format, then creates *atok12_sty* with that information.

atok12migs will not check for existing *atok12_sty*. If it exists, atok12migs will overwrite the existing *atok12_sty* with the new one.

OPTIONS

The following options are supported.

-k	Converts only function-to-key bindings, and uses ATOK12's default for Romaji-to-Kana mappings.
-r	Converts only Romaji-to-Kana mappings, and uses ATOK12's default for function-to-key mappings.

OPERANDS

The following operands are supported.

atok8_ucf	ATOK8 configuration file
atok12_sty	ATOK12-style file created by atok12migs

ENVIRONMENT VARIABLES

LANG, LC_CTYPE

atok12migs operates only when the LC_CTYPE category is set to locale ja or an equivalent locale. For usage of the above environment variables, see the environ(5) man page.

EXIT STATUS

This command returns the following exit codes:

0	Successful completion
1	Unknown option
127	Other errors

FILES

/usr/bin/atok12migs	Command for migrating style setting from ATOK8 to ATOK12

SEE ALSO

environ(5)

Japanese Input System Summary & Transition

ATOK12 User's Guide

NAME

atok12setup – Set up ATOK12 for Japanese input in X environment

SYNOPSIS

`atok12setup`

AVAILABILITY

JSatsvw

DESCRIPTION

`atok12setup` sets up ATOK12 for use as Japanese input system in X environment, to be started when you log in. Values set by `atok12setup` will be in effect when you log in next time.

`atok12setup` modifies the following files.

`$HOME/.dtprofile` A script that is run when you log in to the Common Desktop Environment (CDE). Lines to launch `htt`(1) with `atok12`(1) are added.

If `$HOME/.dtprofile` already exist, and `wnn6setup`(1) provided with Solaris 2.6 or later is set up to start as Japanese input system, `atok12setup` overrides the settings to use ATOK12 as Japanese input system.

Lines to set up ATOK12 are placed between comment lines in `$HOME/.dtprofile` as follows.

```
###== - Generated by atok12setup to launch XIM for Japanese. == BEGIN == -==###
. . .
New settings to start a Japanese input system . . .
. . .
###== - Generated by atok12setup to launch XIM for Japanese. == END == -==###
```

The above setting lines are modified by executing `atok12setup(1)` or `wnn6setup`(1). That is, if you edit the setting lines between comment lines, then execute `atok12setup(1)` or `wnn6setup`(1) changes to these lines will be lost. You should not make any changes to these lines.

FILES

`$HOME/.dtprofile`

SEE ALSO

`dtlogin`(1), `wnn6setup`(1)

NOTES

If you edit `$HOME/.dtprofile` to set up the Japanese input system, and execute the `atok12setup` command, you may not be able to use ATOK12 when you log in. In this case, remove the setting lines you added in `$HOME/.dtprofile` for the Japanese input system.

NAME

cs00toatok – conversion cs00 user dictionary to ATOK user dictionary

SYNOPSIS

`cs00toatok` [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

`cs00toatok` is a filter that converts cs00 word-list-file to ATOK word-list-file. `cs00toatok` is used for the purpose of using cs00 user words on ATOK12 as well as on cs00.

`cs00toatok` reads file(s) specified by *filename(s)*. If no *filenames* are given, `cs00toatok` reads a file from the standard input. The contents of the files must be the format of cs00 word-list-file. `cs00toatok` writes ATOK word-list-file to the standard output. `cs00toatok` converts each word according to the following rules.

`Kana reading (Phonetic), Kanji word`
Does not change any character and the length of Kana reading and Kanji word of each source word.

`Part-of-speech (Hinshi)`
Converts Hinshi of source word according to the table shown below.

Part-of-speech in cs00		Part-of-speech in ATOK	
:N1	noun1	01	common noun
:N2	noun2	01	common noun
:M1	person's name1	02	proper noun
:M2	person's name2	02	proper noun
:T1	place name1	02	proper noun
:T2	place name2	02	proper noun
:NM	numeral	13	numeral
:NN	supplemental numeral	12	suffix
:PR	prefix	11	prefix
:SF	suffix	12	suffix
:AD	adverb	29	adverb
:CN	conjunction	09	conjunction
:RT	participial adjective	08	participial adjective
:AJ	adjective	27	adjective
:AV	adjective verb	28	adjective verb

:SH	S-series irregular conjugation verb	03	noun form of S-series irregular conjugation verb
:ZH	Z-series irregular conjugation verb	04	noun form of Z-series irregular conjugation verb
:1V	single conjugation verb	23	single conjugation verb
:KV	K-series five conjugation verb	14	K-series five conjugation verb
:GV	G-series five conjugation verb	15	G-series five conjugation verb
:SV	S-series five conjugation verb	16	S-series five conjugation verb
:TV	T-series five conjugation verb	17	T-series five conjugation verb
:NV	N-series five conjugation verb	18	N-series five conjugation verb
:BV	B-series five conjugation verb	19	B-series five conjugation verb
:MV	M-series five conjugation verb	20	M-series five conjugation verb
:RV	R-series five conjugation verb	21	R-series five conjugation verb
:WV	W-series five conjugation verb	22	W-series five conjugation verb
:UN	no classification	-	-
:TK	single kanji	07	single kanji
:BS	clause	-	-

Words with the part of speech "no classification" (:UN) or "clause" (:BS) need `-a` option to be put out. Also, a source word with multiple parts of speech is converted to plural words, each of which has the each part of speech.

OPTIONS

`-a` Put out words whose part of speech are "no classification" or "clause" as words whose parts of speech are unknown, in addition to words put out by default.

NOTES

`cs00toatok` does not change Kana reading and Kanji word of any word. Therefore, note below.

- A word may not be registered to ATOK user dictionary with the characters or the length of kana reading and kanji word of the word.
- If the edge of a word is ‘’’ (Zenkaku single quote) or ‘"’ (Zenkaku double quote), a new word stripped the edge of characters from the word is registered.
- If Kanji word of a word contains ‘,’ (Zenkaku comma), the word cannot be registered to ATOK user dictionary.

Use ATOK12 dictionary utility to register a word-list-file to ATOK dictionary. For detail, refer to *ATOK12 User's Guide*.

SEE ALSO

`atok12`(1), `atok8wordlist`(4)

NAME

euctoibmj, ibmjtoeuc – Code conversion between Japanese EUC and IBM-Japanese

SYNOPSIS

euctoibmj [-t] [-u *code*] [-U] [*filename*...]

ibmjtoeuc [-u *code*] [-U] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

euctoibmj converts the contents of the specified *filename*s from ASCII/ Japanese EUC to EBCDIC/IBM–Japanese. ibmjtoeuc converts the contents of the specified *filename*s from EBCDIC/IBM–Japanese to ASCII/ Japanese EUC. The both commands write the resultant code to stdout. If *filename* is not given, input characters are read from the standard input.

For Japanese language handling, the euctoibmj/ibmjtoeucj pair of commands provide conversion only between the two code standards. Code conversion among Japanese EUC, JIS, and PC kanji are supported by another set of commands, jistoeuc(1) family or iconv(1).

OPTIONS

-u *code* With this option specified, characters in one code set that do not have corresponding characters in the other are mapped to the *code* given in four–digit hexadecimal HOST CODE of IBM Japanese (for euctoibmj) or in four–digit JIS Ku–Ten code (for ibmjtoeuc). Without this option, such characters are mapped to HOST CODE 4040 (for euctoibmj) or JIS Ku–Ten code 0101 (for ibmjtoeuc).

-U The output is not buffered (The default is buffered output).

-t With this option specified, euctoibmj translates Half–Size Katakana (Code Set 2) in Japanese EUC to the corresponding characters in Code Set 1 prior to conversion. Without this option, Code Set 2 characters in Japanese EUC are processed to the illegal character.

ENVIRONMENT VARIABLES

The environment variables LC_CTYPE and LANG control the character classification throughout these commands. For euctoibmj and ibmjtoeuc to work correctly, one or both of the environment variables must be set to ja or an equivalent locale. On entry to these commands, these environment variables are checked in the following order: LC_CTYPE and LANG. When a valid value is found, remaining environment variables for character classification are ignored.

FILES

/usr/lib/jcodetables/ibmj-euc
Code conversion table for IBM Japanese.

SEE ALSO

iconv(1), jistoeuc(1), iconv_ja(5)

DIAGNOSTICS

unexpected data encountered in input.
Illegal character code is found in input file.

BUGS

The ASCII/EBCDIC conversion table are taken from the 256 character standard in the CACM Nov, 1968. The conversion, while less blessed as a standard, corresponds better to certain IBM print train convertions. There is no universal solution.

The Japanese EUC/IBM Japanese conversion table is based on the IBM Kanji codebook (4th edition – September 1987), JIS X 0201, and JIS X 0208–1983.

If JIS X 0212 caracter set is specified as input, `euctoibmj` can not support the conversion correctly.

NAME

jistoeuc, jistosj, euctojis, euctosj, sjtojis, sjtoeuc – Code conversion between JIS, PC kanji, and Japanese EUC

SYNOPSIS

jistoeuc [-8] [-U] [*filename*...]

jistosj [-8] [-U] [*filename*...]

euctojis [-8] [-U] [*filename*...]

euctosj [-U] [*filename*...]

sjtojis [-8] [-U] [*filename*...]

sjtoeuc [-U] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

For Japanese language handling, the `jistoeuc` family provides conversion between different code standards. `command` [*filename* . . .] does the specified conversion on the contents of the input *filename*s and writes it to `stdout`.

If *filename* is not given, it reads and converts characters from the standard input.

`jistoeuc`	converts JIS to Japanese EUC
`jistosj`	converts JIS to PC kanji
`euctojis`	converts Japanese EUC to JIS
`euctosj`	converts Japanese EUC to PC kanji
`sjtojis`	converts PC kanji to JIS
`sjtoeuc`	converts PC kanji to Japanese EUC

OPTIONS

`-8`	With this option specified, the commands `jistoeuc`, `jistosj`, `sjtojis`, and `sjtoeuc`, can support JIS X 0201 (Half-Size Katakana). This 8-bit JIS code does not use ISO Shift-In and Shift-Out escape sequences.
`-U`	The output is not buffered (The default is buffered output).

SEE ALSO

`iconv`(1), `iconv_ja`(5)

NOTES

`jistoeuc` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in escape – `\E$B`, `\E$(B`, `\E$@`
JIS X 0212 shift-in escape – `\E$(D`
JIS X 0201 Roman shift-in escape – `\E(J`, `\E(H`
ASCII shift-in escape – `\E(B`

`euctojis` and `sjtojis` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in – `\E$B`

JIS X 0212 shift-in – \E$(D (except when sjtojis command is specified)
JIS X 0201 Roman shift-in – \E(J

This command does not check whether or not each code in the input file is correct. Conversion with PC kanji is not based on *TOG Japanese Vendors Council (TOG/JVC) Recommended Code Set Conversion Specification between Japanese EUC and Shift–JIS*. The iconv(1) utility provides these functions. See iconv(1) and iconv_ja(5) for more information.

BUGS

If JIS X 0212 character set is specified as input, jistosj and euctosj can not support the conversion correctly. euctosj, sjtoeuc, jistosj, and sjtojis can support conversion correctly only if JIS X 0208 1 ku – 84 ku is specified as input.

NAME

jistoeuc, jistosj, euctojis, euctosj, sjtojis, sjtoeuc – Code conversion between JIS, PC kanji, and Japanese EUC

SYNOPSIS

jistoeuc [-8] [-U] [*filename*...]

jistosj [-8] [-U] [*filename*...]

euctojis [-8] [-U] [*filename*...]

euctosj [-U] [*filename*...]

sjtojis [-8] [-U] [*filename*...]

sjtoeuc [-U] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

For Japanese language handling, the `jistoeuc` family provides conversion between different code standards. `command` [*filename* . . .] does the specified conversion on the contents of the input *filename*s and writes it to `stdout`.

If *filename* is not given, it reads and converts characters from the standard input.

`jistoeuc`	converts JIS to Japanese EUC
`jistosj`	converts JIS to PC kanji
`euctojis`	converts Japanese EUC to JIS
`euctosj`	converts Japanese EUC to PC kanji
`sjtojis`	converts PC kanji to JIS
`sjtoeuc`	converts PC kanji to Japanese EUC

OPTIONS

`-8`	With this option specified, the commands `jistoeuc`, `jistosj`, `sjtojis`, and `sjtoeuc`, can support JIS X 0201 (Half-Size Katakana). This 8-bit JIS code does not use ISO Shift-In and Shift-Out escape sequences.
`-U`	The output is not buffered (The default is buffered output).

SEE ALSO

`iconv`(1), `iconv_ja`(5)

NOTES

`jistoeuc` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in escape – `\E$B`, `\E$(B`, `\E$@`
JIS X 0212 shift-in escape – `\E$(D`
JIS X 0201 Roman shift-in escape – `\E(J`, `\E(H`
ASCII shift-in escape – `\E(B`

`euctojis` and `sjtojis` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in – `\E$B`

JIS X 0212 shift-in – \E$(D (except when sjtojis command is specified)
JIS X 0201 Roman shift-in – \E(J

This command does not check whether or not each code in the input file is correct. Conversion with PC kanji is not based on *TOG Japanese Vendors Council (TOG/JVC) Recommended Code Set Conversion Specification between Japanese EUC and Shift–JIS*. The iconv(1) utility provides these functions. See iconv(1) and iconv_ja(5) for more information.

BUGS

If JIS X 0212 character set is specified as input, jistosj and euctosj can not support the conversion correctly. euctosj, sjtoeuc, jistosj, and sjtojis can support conversion correctly only if JIS X 0208 1 ku – 84 ku is specified as input.

NAME

evftobdf – convert evfont file to BDF format

SYNOPSIS

`evftobdf` `[-t] [-p` *`propertyfile`*`] `*`filename`*`...`

AVAILABILITY

SUNWjfpu

DESCRIPTION

`evftobdf` converts the evfont format file to X11 BDF 2.1 format file. `evftobdf` is typically used to generate fonts for use with the X11/NeWS window system.

OPTIONS

`-t` Prints a properties of the BDF fonts on standard output; a reformatted font file is not dumped. The filename must be BDF format file.

`-p`*`propertyfile`* Takes in BDF properties from other file. The property file take format like BDF, and it can define 2 properties which is seperated by "{" and "}" block in one file.

NAME

euctoibmj, ibmjtoeuc – Code conversion between Japanese EUC and IBM-Japanese

SYNOPSIS

euctoibmj [-t] [-u *code*] [-U] [*filename*...]

ibmjtoeuc [-u *code*] [-U] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

euctoibmj converts the contents of the specified *filename*s from ASCII/ Japanese EUC to EBCDIC/IBM–Japanese. ibmjtoeuc converts the contents of the specified *filename*s from EBCDIC/IBM–Japanese to ASCII/ Japanese EUC. The both commands write the resultant code to stdout. If *filename* is not given, input characters are read from the standard input.

For Japanese language handling, the euctoibmj/ibmjtoeucj pair of commands provide conversion only between the two code standards. Code conversion among Japanese EUC, JIS, and PC kanji are supported by another set of commands, jistoeuc(1) family or iconv(1).

OPTIONS

-u *code* With this option specified, characters in one code set that do not have corresponding characters in the other are mapped to the *code* given in four–digit hexadecimal HOST CODE of IBM Japanese (for euctoibmj) or in four–digit JIS Ku–Ten code (for ibmjtoeuc). Without this option, such characters are mapped to HOST CODE 4040 (for euctoibmj) or JIS Ku–Ten code 0101 (for ibmjtoeuc).

-U The output is not buffered (The default is buffered output).

-t With this option specified, euctoibmj translates Half–Size Katakana (Code Set 2) in Japanese EUC to the corresponding characters in Code Set 1 prior to conversion. Without this option, Code Set 2 characters in Japanese EUC are processed to the illegal character.

ENVIRONMENT VARIABLES

The environment variables LC_CTYPE and LANG control the character classification throughout these commands. For euctoibmj and ibmjtoeuc to work correctly, one or both of the environment variables must be set to ja or an equivalent locale. On entry to these commands, these environment variables are checked in the following order: LC_CTYPE and LANG. When a valid value is found, remaining environment variables for character classification are ignored.

FILES

/usr/lib/jcodetables/ibmj-euc
Code conversion table for IBM Japanese.

SEE ALSO

iconv(1), jistoeuc(1), iconv_ja(5)

DIAGNOSTICS

unexpected data encountered in input.
Illegal character code is found in input file.

BUGS

The ASCII/EBCDIC conversion table are taken from the 256 character standard in the CACM Nov, 1968. The conversion, while less blessed as a standard, corresponds better to certain IBM print train convertions. There is no universal solution.

The Japanese EUC/IBM Japanese conversion table is based on the IBM Kanji codebook (4th edition – September 1987), JIS X 0201, and JIS X 0208–1983.

If JIS X 0212 caracter set is specified as input, `euctoibmj` can not support the conversion correctly.

NAME

jistoeuc, jistosj, euctojis, euctosj, sjtojis, sjtoeuc – Code conversion between JIS, PC kanji, and Japanese EUC

SYNOPSIS

jistoeuc [-8] [-U] [*filename*...]

jistosj [-8] [-U] [*filename*...]

euctojis [-8] [-U] [*filename*...]

euctosj [-U] [*filename*...]

sjtojis [-8] [-U] [*filename*...]

sjtoeuc [-U] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

For Japanese language handling, the `jistoeuc` family provides conversion between different code standards. `command` [*filename* . . .] does the specified conversion on the contents of the input *filename*s and writes it to `stdout`.

If *filename* is not given, it reads and converts characters from the standard input.

`jistoeuc`	converts JIS to Japanese EUC
`jistosj`	converts JIS to PC kanji
`euctojis`	converts Japanese EUC to JIS
`euctosj`	converts Japanese EUC to PC kanji
`sjtojis`	converts PC kanji to JIS
`sjtoeuc`	converts PC kanji to Japanese EUC

OPTIONS

`-8`	With this option specified, the commands `jistoeuc`, `jistosj`, `sjtojis`, and `sjtoeuc`, can support JIS X 0201 (Half-Size Katakana). This 8-bit JIS code does not use ISO Shift-In and Shift-Out escape sequences.
`-U`	The output is not buffered (The default is buffered output).

SEE ALSO

`iconv`(1), `iconv_ja`(5)

NOTES

`jistoeuc` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in escape – `\E$B`, `\E$(B`, `\E$@`
JIS X 0212 shift-in escape – `\E$(D`
JIS X 0201 Roman shift-in escape – `\E(J`, `\E(H`
ASCII shift-in escape – `\E(B`

`euctojis` and `sjtojis` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in – `\E$B`

JIS X 0212 shift-in – `\E$(D` (except when `sjtojis` command is specified)
JIS X 0201 Roman shift-in – `\E(J`

This command does not check whether or not each code in the input file is correct. Conversion with PC kanji is not based on *TOG Japanese Vendors Council (TOG/JVC) Recommended Code Set Conversion Specification between Japanese EUC and Shift–JIS*. The `iconv(1)` utility provides these functions. See `iconv`(1) and `iconv_ja`(5) for more information.

BUGS

If JIS X 0212 character set is specified as input, `jistosj` and `euctosj` can not support the conversion correctly. `euctosj`, `sjtoeuc`, `jistosj`, and `sjtojis` can support conversion correctly only if JIS X 0208 1 ku – 84 ku is specified as input.

NAME

jistoeuc, jistosj, euctojis, euctosj, sjtojis, sjtoeuc – Code conversion between JIS, PC kanji, and Japanese EUC

SYNOPSIS

jistoeuc [-8] [-U] [*filename*...]

jistosj [-8] [-U] [*filename*...]

euctojis [-8] [-U] [*filename*...]

euctosj [-U] [*filename*...]

sjtojis [-8] [-U] [*filename*...]

sjtoeuc [-U] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

For Japanese language handling, the `jistoeuc` family provides conversion between different code standards. `command` [*filename* . . .] does the specified conversion on the contents of the input *filename*s and writes it to `stdout`.

If *filename* is not given, it reads and converts characters from the standard input.

`jistoeuc`	converts JIS to Japanese EUC
`jistosj`	converts JIS to PC kanji
`euctojis`	converts Japanese EUC to JIS
`euctosj`	converts Japanese EUC to PC kanji
`sjtojis`	converts PC kanji to JIS
`sjtoeuc`	converts PC kanji to Japanese EUC

OPTIONS

`-8`	With this option specified, the commands `jistoeuc`, `jistosj`, `sjtojis`, and `sjtoeuc`, can support JIS X 0201 (Half-Size Katakana). This 8-bit JIS code does not use ISO Shift-In and Shift-Out escape sequences.
`-U`	The output is not buffered (The default is buffered output).

SEE ALSO

`iconv`(1), `iconv_ja`(5)

NOTES

`jistoeuc` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in escape – `\E$B`, `\E$(B`, `\E$@`
JIS X 0212 shift-in escape – `\E$(D`
JIS X 0201 Roman shift-in escape – `\E(J`, `\E(H`
ASCII shift-in escape – `\E(B`

`euctojis` and `sjtojis` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in – `\E$B`

JIS X 0212 shift-in – `\E$(D` (except when `sjtojis` command is specified)
JIS X 0201 Roman shift-in – `\E(J`

This command does not check whether or not each code in the input file is correct. Conversion with PC kanji is not based on *TOG Japanese Vendors Council (TOG/JVC) Recommended Code Set Conversion Specification between Japanese EUC and Shift–JIS*. The `iconv(1)` utility provides these functions. See `iconv`(1) and `iconv_ja`(5) for more information.

BUGS

If JIS X 0212 character set is specified as input, `jistosj` and `euctosj` can not support the conversion correctly. `euctosj`, `sjtoeuc`, `jistosj`, and `sjtojis` can support conversion correctly only if JIS X 0208 1 ku – 84 ku is specified as input.

NAME

jpostprint – PostScript translator for Japanese text files

SYNOPSIS

jpostprint [-c *num*] [-f *name*] [-l *num*] [-m *num*] [-n *num*] [-o *list*] [-p *mode*] [-r *num*] [-s *num*] [-t *num*] [-x *num*] [-y *num*] [-u *path*] [*filename*...]

/usr/lib/lp/postscript/jpostprint

AVAILABILITY

SUNWjfpu

DESCRIPTION

The `jpostprint` translates Japanese characters on the standard input into Japanese PostScript and writes the results on the standard output. If no *filename*s are specified, or if – is one of the input *filename*s, the standard input is read. When the input character code includes UDC (User Defined Character), JIS X 0212, IBM extention character, or IBM extention character NEC selected, these fonts are also printed. And other fonts is printed by printer's fonts. UDC is created by using `sdtudctool`(1). SUNWjcs3f is needed for printing JIS X 0212, IBM extention character, or IBM extention character NEC selected.

OPTIONS

`-c` *num* — Print *num* copies of each page. By default, only one copy is printed.

`-f` *name* — Specify the font for printing. The following fonts can be specified as *name* to change fonts for JIS X 0208 or JIS X 0201 Katakana character set.

Ryumin-Light (same as Ryumin-Light-H)
Ryumin-Light-H
GothicBBB-Medium (same as GothicBBB-Medium-H)
GothicBBB-Medium-H
Ryumin-Light-V
GothicBBB-Medium-V

The following fonts may be specified for ASCII characters.

JIS X 0201 Roman character set	Ryumin-Light.Hankaku GothicBBB-Medium.Hankaku
English font	Courier LucidaSans-Typewriter

The following example shows how to specify the multiple fonts.

GothicBBB-Medium.Hankaku+GothicBBB-Medium

	By default, Courier is specified for ASCII characters, and Ryumin-Light-H is specified for JIS X 0208 or JIS X 0201 Katakana characters. It is impossible to change the fonts for JIS X 0212, VDC (Vender Defined Character), and UDC.
`-l` *num*	Set the length of a page to *num* lines. By default, *num* is 66. Setting *num* to 0 is allowed, and will cause `jpostprint` to guess a value, based on the point size that's being used.
`-m` *num*	Magnify each logical page by the factor *num*. Pages are scaled uniformly about the origin, which is located near the upper left corner of each page. The default magnification is 1.0.
`-n` *num*	Print *num* logical pages on each piece of paper, where *num* can be any positive integer. By default, *num* is set to 1.
`-o` *list*	Print pages whose numbers are given in the comma-separated *list*. The *list* contains single numbers *N* and ranges *N1 – N2*. A missing *N1* means the lowest numbered page, a missing *N2* means the highest. The page range is an expression of logical pages rather than physical sheets of paper. For example, if you are printing two logical pages to a sheet, and you specified a range of 4, then two sheets of paper would print, containing four page layouts.
`-p` *mode*	Print *files* in either portrait or landscape *mode*. Only the first character of *mode* is significant. The default *mode* is portrait. The *mode* is an expression for logical page rather than physical page. For example, portrait for logical page two correspond to landscape for physical page.
`-r` *num*	Selects carriage return behavior. Carriage returns are ignored if *num* is 0, cause a return to column 1 if *num* is 1, and generate a newline if *num* is 2. The default *num* is 0.
`-s` *num*	Print *files* using point size *num*. When printing in landscape mode *num* is scaled by a factor that depends on the imaging area of the device. The default size for portrait mode is 10. Note that increasing point size increases virtual image size, so you either need to load larger paper, or use the `-l0` option to scale the number of lines per page.
`-t` *num*	Assume tabs are set every *num* columns, starting with the first column. By default, tabs are set every `8` columns.
`-x` *num*	Translate the origin *num* inches along the positive x axis. The default coordinate system has the origin fixed near the upper left corner of the page, with positive x to the right and positive y down the page. Positive *num* moves everything to the right. The default offset is 0.25 inches.
`-y` *num*	Translate the origin *num* inches along the positive y axis. Positive *num* moves text up the page. The default offset is –0.25 inches.

`-u` *path* — Print *path* using UDC font. By default, it refers `$HOME/.Xlocale/`*locale*`/fonts/UDC/Type1/UDC`*xx*`.pfa` first, and then `fontpath udc` in `jpostprint.conf` file.

A new logical page is started after 66 lines have been printed on the current page, or whenever an ' `\f` ' character is read. The number of lines per page can be changed using the `-l` option. Unprintable characters are ignored.

EXAMPLES

EXAMPLE 1

To print *file1* and *file2* in landscape mode, issue the following command:

```
example% jpostprint -pland file1 file2
```

EXAMPLE 2

To print two logical pages on each physical page in portrait mode:

```
example% jpostprint -n2 file
```

EXAMPLE 3

To print UDC1.pfa, . . . , UDC20.pfa of UDC font in `/usr/local/UDC` directory:

```
example% jpostprint -u " /usr/local/UDC/UDC%d.pfa"
```

ENVIRONMENT VARIABLES

The locale has to be set to ja, ja_JP.eucJP, ja_JP.PCK, or ja_JP.UTF-8 in your environment.

FILES

`/usr/lib/lp/postscript/forms.ps`

`/usr/lib/lp/postscript/jpostprint.ps`

`/usr/lib/lp/postscript/jpostprint.conf`

SEE ALSO

`jtops`(1), `postprint`(1), `sdtudctool`(1), `lpfilter`(1M)

NOTES

If `-n` option specified, PostScript file properly shows only for printing. UDC font is only used by Type1 font `sdtudctool` printed, and JIS X 0212 font is only used by Type1 font in SUNWjcs3f.

If the locale is set to ja_JP.UTF-8, `jpostprint` supports 6400 and less characters for UDC font, and the characters that can be supported in Japanese EUC for other fonts.

NAME

jprconv – Filter for printing Japanese text on a dot-matrix Kanji printer or Japanese language page printer

SYNOPSIS

jprconv [-T *terminfo*] [-r]

/usr/lib/lp/text/jprconv

AVAILABILITY

SUNWjfpu

DESCRIPTION

jprconv is a filter for printing Japanese text on a dot-matrix Kanji printer (EPSON VP-5085 or NEC PC-PR201) or Japanese language page printer (Canon LASERSHOT). Control codes for each printer are as follows:

Database	Control code
EPSON VP-5085	ESC/P24-J84 of EPSON ESC/P
NEC PR201	NEC 201PL
Canon LASERSHOT	LIPS-complied control code

If the above control codes are supported, Japanese text can be printed on another printer.

jprconv reads Japanese characters from the standard input, converts them to each control code, and writes to the standard output. If the input character code includes any user-defined characters, JIS X 0212, IBM Extended characters, or NEC-selective IBM Extended characters, these fonts are also printed. For the other characters, fonts installed on the printer are used.

You can use sdtudctool to define user-specific characters (see sdtudctool(1)). To print JIS X 0212, IBM Extended characters, or NEC-selective IBM Extended characters, the SUNWjcs3f package is also needed.

OPTIONS

-T Specifies to use *terminfo* database. Any one of the following must be spefified.

canon-ls-408	In case of LIPS format
nec-pr201	In case of NEC 201PL format
epson-vp5085	In case of ESC/P24-J84 format

-r Does not convert NL to CR-NL when printing. By default it is converted.

EXAMPLES

To print *file1* in the ESC/P24-J84 format, type:

```
example% jprconv -T epson-vp5085 < file1
```

ENVIRONMENT VARIABLES

To run this command, the locale must be set to ja, ja_JP.PCK or ja_JP.UTF-8.

FILES

`/usr/lib/lp/text/jprconv.conf`

`/usr/share/lib/terminfo/e/epson-vp5085`

`/usr/share/lib/terminfo/n/nec-pr201`

`/usr/share/lib/terminfo/c/canon-ls-a408`

SEE ALSO

`jtops`(1), `jpostprint`(1), `sdtudctool`(1), `lpfilter`(1M)

Japanese Environment User's Guide

NOTES

In general, it is not necessary to use the `jprconv` because `jprconv` is used as a filter on the printer server side. For the setting on the printer server side, see *Japanese Environment User's Guide*.

Only Japanese characters are printable in the ja_JP.UTF-8 locale.

NAME

jtops – postscript filter for printing Japanese characters on Sun Laser Writer or Japanese postscript printer

SYNOPSIS

jtops [-12rRjJvwWmg] [-ln] [-s *size*] [-f *font*] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

jtops is a filter for converting Japanese characters to Japanese postscript output which uses Kanji font on a printer side. Input from *stdin* is converted and sent to *stdout*.

If there is no *filename*, the standard input is read.

By default, it forms font size 10 and 66 lines per page for the portrait form.

Before checking options specified in command lines, it interprets the strings in the JTOPS environment variable as options.

OPTIONS

-1	1 column output (by default).
-2	2 columns output.
-r	Rotate for the landscape form.
-R	Output in the portrait form (by default).
-v	Use Kanji-fonts of printer (for the Japanese PostScript printer by default).
-l *n*	Specify the number of lines per page as *n* (by default 66).
-m	Use Ming style as Japanese fonts (by default).
-g	Use Gothic style as Japanese fonts. As alphabetic fonts use Courier-Bold style unless -f option is specified.
-f *font*	Specify alphabetic fonts in *font*. By default, use Courier style without -g option, otherwise Courier-Bold style.
-s *size*	Set font size to *size*. When the fontsize is specified with -s, the lines per page are calculated as follows: portrait: 720 / (fontsize + 1) landscape: 550 / (fontsize + 1)
-j	Use half size alphanumeric of Japanese fonts as alphanumeric fonts.
-J	Use Courier as alphanumeric fonts(by default).
-W	Control the ratio of alphanumeric character and Japanese to be 1:2, except changeable width of alphanumeric fonts(by default).
-w	Not control the ratio of alphanumeric character and Japanese.

EXAMPLES

EXAMPLE 1

```
example% jtops -j -l40 -s11 filename | lpr
example% pr -l120 filename | jtops -l120 -s5 | lpr
example% jtops -2r filename | lpr
```

ENVIRONMENT VARIABLES

The locale has to be set to ja, ja_JP.eucJP, ja_JP.PCK, or ja_JP.UTF-8 in your environment.

SEE ALSO

`expand`(1), `lp`(1), `pr`(1), `lpr`(1B), `lpfilter`(1M)

NOTES

`jtops` supports the following character sets;

- JIS X 0201 figure character set for Roman
- JIS X 0201 figure character set for Katakana
- JIS X 0208

NAME

jtty – set Japanese terminal characteristics

SYNOPSIS

jtty [-c *y* | *n*] [-i *c*] [-o *c*]

AVAILABILITY

SUNWjfpu

DESCRIPTION

For modules in the Stream that do code conversion on the JIS character set (EUC-JIS, Shift-JIS, 7-bit JIS, and 8-bit JIS), the `jtty` command is used to turn conversion on and off.

Additionally it will set or report the values that the conversion module uses for the third character of the JIS announcement sequence. This character is part of the three-character ISO sequence for introducing JIS characters.

When used with no parameters, `jtty` reports the values of the JIS input and output escape characters.

OPTIONS

`-c` *y* Turn code conversion on.

`-c` *n* Turn code conversion off.

`-i` *c* Set the third character of the JIS input escape sequence to '*c*'.

`-o` *c* Set the third character of the JIS output escape sequence to '*c*'.

SEE ALSO

`stty`(1), `jaio`(7)

NAME

kanji – show the list of Kanji codes

SYNOPSIS

kanji [-j] [-s] [-e] [-k] [-K *n*] [-HK *n*] [-h]

AVAILABILITY

SUNWjfpu

DESCRIPTION

The kanji command shows the list of printable characters in the current locale. When invoked without an option, encoding numbers of current locale are shown along with the characters.

OPTIONS

-j	JIS code numbers are shown.
-s	PC kanji code numbers are shown. This option is available only in a locale equivalent to "ja" locale.
-e	EUC code numbers are shown. This option is available only in a locale equivalent to "ja_JP.PCK" locale.
-k	JIS Kuten numbers are shown.
-K*n*	In a locale equivalent to "ja" locale, this shows only those characters from row of JIS X 0208 character set *n* in Kuten code. In a locale equivalent to "ja_JP.PCK" locale, this shows only those characters from row of printable characters set *n* in Kuten code.
-HK *n*	This shows only those characters from row of JIS X 0212 character set *n* in Kuten code. This option is available only in a locale equivalent to ja locale.
-h	Help for this command is shown.

SEE ALSO

PCK(5), eucJP(5)

NAME

kkcvtocs00 – conversion from kkcv user dictionary to cs00 user directory

SYNOPSIS

kkcvtocs00 [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

`kkcvtocs00` is a utility that is necessary to use the words of your user dictionary of `cs00` on JFP system as well as one of `kkcv` on JLE system. `kkcvtocs00` can be used as a filter; if *filename* is given, it is used as an input. Otherwise, the standard input is used for it. The standard output is always used as an output. The input format should be the format of "jiritsugo-file" in `kkcvdicedit` which is specified for a line. `kkcvtocs00` shows an error message for a line that is not on the format, and ignore the line.

EXAMPLES

EXAMPLE 1

Create your user dictionary of `cs00` from one of user dictionary of `kkcv` as follows.

1. In advance, on the JLE system, create "jiritsugo-file" from your user dictionary of `kkcv`.

   ```
   JLE% kkcvdicedit extract kkcv_u.dic kkcv_u.list
   ```

2. On the JFP system, convert this "jiritsugo-file" into "tango-list-file" of `udicm`.

   ```
   JFP% kkcvtocs00 kkcv_u.list > cs00_u.list
   ```

3. Create your user dictionary of `cs00` from this "tango-list-file".

   ```
   JFP% udicm create /usr/lib/mle/ja/cs00/cs00_m.dic cs00_u.dic cs00_u.list
   ```

SEE ALSO

`mdicm`(1), `udicm`(1)

DIAGNOSTICS

The only purpose of `kkcvtocs00` is to support of inheritance of user dictionary words from `kkcv` to `cs00`. `kkcvtocs00` assumes that "jiritsugo-file"(s) will be given as an input file. Therefore, its error process is not always strict.

Currently, the following error and warning messages are defined.

`Error: file` *filename* `line` *number*`: is not kkcvdicedit jiritsugo format, ignored.`

`Warning: file` *filename* `line` *number*`: "Invalid hinshi(s) have found. They are converted into ":UN".`

NAME

sdtudctool – Solaris gaiji tool

SYNOPSIS

`/usr/dt/bin/sdtudctool` `[-f` *file*`]`

DESCRIPTION

Beginning from Solaris 2.6, user–defined characters can be handled as separate font files without editing existing font files.

`sdtudctool` is a tool for registering user–defined characters at the above areas as separate font files on the Solaris Common Desktop Environment (Solaris CDE). Using `sdtudctool`, it is possible to register, all at once, user–defined characters for bitmap fonts of various sizes used on-screen. It can even be used to register user–defined characters for outline fonts. Fonts that are created on Solaris CDE can be also used on OpenWindows under Solaris 2.6 and later.

When the directory in which user–defined characters are saved is specified in DTUDCFONTPATH environment variable, `stdudctool` will edit font files in this directory. Otherwise, it will edit font files in the following directories. (If DTUDCFONTPATH is not specified or the directory is not existed, directories will be created automatically).

Directories which include font files for user–defined characters under Solaris 2.6 and later.

For end user `$HOME/.Xlocale/<`*locale*`>/fonts/UDC/`

`Bitmaps`

`Type1`

`CID`

For superuser `$OPENWINHOME/lib/locale/<`*locale*`>/X11/fonts/UDC/`

`Bitmaps`

`Type1`

`CID`

Note that if user–defined characters have registered by `fontedit` or `fontmanager` and user want to re–use them, those characters will need to be ported to the new environment. See NOTES below, `sdtudc_extract`(1), and `sdtudc_convert`(1).

Screen Configuration

`sdtudctool` consists of the following 3 basic windows.

- editor
- list
- reference list

It also includes optional dialog and print list dialog screens and so on.

editor

When `sdtudctool` is started, the following window appears first. If user–defined characters are usable, they will automatically be loaded and displayed.

The screen consists of the following items.

Edit Screen	Screen for editing user–defined characters.
Reference Screen	Screen for displaying a character to use as reference when creating a user–defined character. This screen may be displayed to the right of the Edit Screen either by selecting [Display] -> [Reference] or by pressing the [Reference] button at the right on the toolbar.
Confirmation Screens	Two screens displayed at lower left. The left screen displays the outline of the character, while the right screen displays a bitmap image of the character.
Draw Menu	You can draw graphics by selecting one of the Draw menus at left of edit screen and dragging the mouse on the edit screen. However, how to operate drawing break lines and polygons are different from other drawing. You can click select button at every start and end point of a line segment on the edit screen, and new lie segment would be started drawing from the point. Fix the drawing by double click at the last line segment. The following menus are available:

Free–hand	Draws free–hand lines.
Straight line	Draws straight lines.
Break line	Draws break lines.
Polygons	Draws polygons.
Rectangles	Draws rectangles.
Circles	Draws circles.
Eraser	Erases the region indicated by the cursor. However, it is not selected for outline fonts and then it must be selected edit menu for erasing.
Area specification	Specifies the region within which to edit. It must be specified the region first when user uses the command from edit menues. Objects in the specified region will be displayed small rectangles as control points. To move objects, specify the region and drag the mouse on the center of the object.

For other editing , see EDIT below.

Save button — Select this button to register a user–defined character which has been created on the Edit Screen. Once selected, the user–defined character registered will be displayed in the list.

Next button — Sets the character to be edited to the next user–defined character in the list.

Previous button — Sets the character to be edited to the previous user–defined character in the list.

The following options are displayed on the toolbar.

[Outline] — Sets the display or edit mode for the Edit Screen to outline mode.

[Bitmap] — Sets the display or edit mode for the Edit Screen to bitmap mode.

[Reference] — Displays/Undisplays a Reference Screen to the right of the Edit Screen.

[List] — Displays a list of characters contained in the font file being edited.

The following menus are displayed on the menu bar.

[File]

[Open the user–defined characters. . .]
Loads the font files for user–defined characters under Solaris 2.6 and later and displays them. If the `DTUDCFONTPATH` environment variable is not valid, `sdtudctool` will open files in the following directories.

For end user `$HOME/.Xlocale/<locale>/fonts/UDC/`

```
Bitmaps
Type1
CID
```

For superuser `$OPENWINHOME/lib/locale/<locale>/X11/fonts/UDC/`

```
Bitmaps
Type1
CID
```

[Open. . .]
Used to load a specified font file. Registered files will not be saved as separate font files, and they will be written into the font files read directly. The font types that can be loaded are following;

BDE format
PCF format
Type1 format (not editable)

Type3 format (not editable)

[Save]
If the font files for user–defined characters under Solaris 2.6 and later was loaded, this saves it as a user–defined character and `sdtudctool` does setting required for using of it.

The setting required is setting font path and saving it. The setting font path can be omitted and see OPTION below.

If loaded by directly specifying a font file, changes are directly reflected in that font file. It is not Setting font path and saving it.

[Save as. . .]
Used when saving results of editing as a font file of a different name. This menu cannot be selected if the font files for user–defined characters under Solaris 2.6 and later was loaded. The font types that can be loaded are following;

BDE format
PCF format

[Open Dict Tool. . .]
Starts `sdtudc_register` that is intermediate utility for registering user–defined characters. `sdtudc_register` displays user–defined characters registered, and you can input and register the user–defined characters in dictionary in this window. See `sdtudc_register`(1).

[Options]
The following options can be specified.

Grid size	Specifies the grid size on editting for outline mode by point.
Align to grid	Specifies to align to grid the drawing location on editting for outline mode.
Set font path	Adds the directory which saved the user–defined characters. See `xset`(1).
Save font path	Sets font path to save the user–defined characters to the following files.
	For end user: `$HOME/.OWfontpath`
	For superuser: `$OPENWINHOME/lib/locale/`<*locale*>`/OWfontpath`
	Generation size of the bitmap file: Specifies the size of the bitmap font file that generated automatically when the user–defined characters is saved. It can generate the following sizes.

12, 14, 16, 20, 24

[Exit]
Exits `sdtudctool`.

[Edit] Select the object after the region was specified. To move objects, drag the mouse on the center of the object.

[Undo] Returns to the immediately preceding status.

[Cut] Cuts the selected region and places it in a buffer. You can draw the contents of the buffer on a edit screen if paste operation will be done after cut.

[Copy] Copies the selected region. You can draw the contents of the buffer on a edit screen if paste operation will be done after copy.

[Paste] Pastes the contents of the buffer.

[Delete] Deletes editing contents. The contents are not placed in a buffer, so paste operation is not available.

[Rotate] Rotates the selected region.

[Diagonal] Converts the selected region to a diagonal region.

[Reverse]

For outline mode: Reverses the direction of the selected font path. `sdtudctool` draws by non–zero winding number rule for outline mode. Path direction is following;

Draws in the specified order.
free–hand, straight line, break line, polygons

Draws right–handed revolution.
rectangles, circles

For example, if you create whitewashed circles, draw a small circle in a large circle and reverse the small circle. Then the small circle reverses.

For bitmap mode:
Reverses black and white within the selected region.

See *PostScript reference manual* (Adobe Systems, Inc.) for non–zero winding number rule.

[Display]

[Fill] Fills the region enclosed by an outline when in outline mode. The resulting image is the one actually used when displaying the outline font.

[Control] Displays control points when drawing outlines. Area specification is available only if the specified region includes all control points in outline mode.

[Drag display] Specifies the method to use to display intermediate images when changing the selected region (with Rotate or Diagonal).

[Grid] Displays a grid on the Edit Screen. Grid can be changed the size by selecting [Grid size] in opetion dialog in outline mode.

[Reference] Opens the Reference Screen.

[Help]

[Summary]
Displays the `sdtudctool`(1) this manual page.

[Using help]
Starts AnswerBook.

[About the Solaris gaiji tool]
Displays the version number for the Solaris gaiji tool.

list This will give a list of characters registered in the font file. If you want to find out what type of characters are registered in a font file read in by `sdtudctool`.

The following items are displayed on the toolbar.

[Page] Move to the desired page by moving the slider.

[Left arrow] Go to the previous page.

[Right arrow] Go to the next page.

The following items are displayed on the menu bar.

[File]

[Print. . .] Opens the print dialog. Prints a list of user–defined character registered in the print dialog. The size of printing characters are 15 and 30 points. It is available to specify as follows;

printer name Specifies the output printer name.

range

this page Prints a list of user–defined character which is displayed currently.

all pages Print a list of all user–defined character which is loaded currently.

[Close] Closes the list window.

[Edit]

[Undo]	Returns to the immediately preceding status.
[Cut]	Cuts the selected region and places it in a buffer.
[Copy]	Copies the selected region in a buffer.
[Paste]	Pastes the contents of the buffer at the specified position.
[Delete]	Deletes the specified character.

[Display]

[Next page]	Goes forward one page if there is a next one.
[Previous page]	Goes backward one page if there is a previous one.
[Size]	Changes the font size being displayed. Size cannot be changed if a bitmap font file is loaded.
[Code]	Changes the code format of the character being displayed.

reference list

This is used when you wish to reference another character while editing a user–defined character. The character reference list window may be displayed by selecting the button at the upper left on the toolbar of the user–defined character editor and then the [Display] -> [Reference] buttons.

The following items are displayed on the toolbar.

[Page]	Move to the desired page by moving the slider.
[Left arrow]	Go to the previous page.
[Right arrow]	Go to the next page.

The following items are displayed on the menu bar.

[File]

[Open]	Opens a file selection list for specifying the font to access. You can specify BDF format, PCF format, Type1 format and Type3 format.
[Select from installed fonts]	Displays a list of fonts which can be used by the system. Select the desired font to view in on the reference list.
[Close]	Closes the reference list.

[Display]

[Next page]	Goes forward one page if there is a next one.
[Previous page]	Goes backward one page if there is a previous one.
[Size]	Changes the font size being displayed. Size cannot be changed if a bitmap font file is loaded.
[Code]	Changes the code format of the character being displayed.
[Other functions]	Solaris gaiji tool supports drag & drop operation; on the list, from the list to edit screen, from reference list to edit screen, and reference list to the list. You can drag & drop by selecting characters and drowing it to the target location.

OPTIONS

`-f` *file* Specifies the font file to be edited.

USAGE

How to start Solaris gaiji tool is opening [desktop apprication] within apprication manager and selecting [Solaris gaiji tool], or entering the following command within terminal emulator and so on.

```
sun% /usr/dt/bin/sdtudctool
```

RESOURCES

`utRefXFont` Specifies the font displayed in the reference window by XLFD name. Default is `-sun-gothic-medium-r-normal--16-140-75-75-c-140-jisx0208.1983-0 .`

`utUDCPrefix` Specifies the prefix of bitmap/outline font file in which the user–defined character is saved. Default is `UDC`.

The font file in which the user–defined character is saved is as follows.

For bitmap font file:
<the value of `utUDCPrefix`><`utBDFUDCSize`>`.pcf`

For outline font file:
<the value of `utUDCPrefix`><`utCIDUDCBase`>–;<`utCIDUDCRange`>`.ps`

For example, the following font files are saved by default in Japanese.

For bitmap font file:
`UDC{12,14,16,24}.pcf`

For Type1 font file:
`UDC[1-20].pfa`

For outline font file of JIS encording:

```
NewGothicBBB-Medium-{H,V}.ps
NewGothicBBB-Medium-Hojo-{H,V}.ps
NewRyumin-Light-{H,V}.ps
NewRyumin-Light-Hojo-{H,V}.ps
```

For outline font file of EUC encording:

```
NewGothicBBB-Medium-EUC-{H,V}.ps
NewGothicBBB-Medium-Hojo-EUC-{H,V}.ps
NewRyumin-Light-EUC-{H,V}.ps
NewRyumin-Light-Hojo-EUC-{H,V}.ps
```

For outline font file of SJIS encording:

```
NewGothicBBB-Medium-RKSJ-{H,V}.ps
NewRyumin-Light-RKSJ-{H,V}.ps
```

For outline font file of 83pv–RKSJ encording:

```
NewGothicBBB-Medium-83pv-RKSJ-{H,V}.ps
NewRyumin-Light-83pv-RKSJ-{H,V}.ps
```

`utUDCBDFSize` — Specifies the size for the bitmap font from which to load user–defined characters. Defaults are 12, 14, 16, 20 and 24.

`utUDCBDFCR:` — Specifies `Charset Registry` for the user–defined character font. Default is `sunudcja.1997`.

`utUDCBDFCE:` — Specifies `Charset encoding` for the user–defined character font. Default is `0`.

`utUDCCIDPrefix` — Specifies the prefix of outline font file from which to create and load user–defined characters. Default is `New`.

`utUDCCIDBase` — Specifies the outline font from which to load user–defined characters. Default is `GothicBBB-Medium,Ryumin-Light`.

`utUDCBDFAliasTo` — Specifies an alias for bitmap font in which the user–defined character is saved. Defaults are

```
-sun-gothic-bold-r-normal-*,\
-sun-gothic-medium-r-normal-*,\
-dt-interface
system-medium-r-normal-*-*-*-*-*-m-*,\
-dt-interface
user-medium-r-normal-*-*-*-*-*-m-*.
```

SEE ALSO `sdtudc_convert`(1), `sdtudc_extract`(1), `sdtudc_register`(1), `sdtudc_map`(4)

NOTES Be sure to perform the following porting procedures if there are user–defined characters which have been registered using `fontedit` , `type3creator` , and `fontmanager` under Solaris 2.5.1 and earlier environments. The following examples are for the default environment in `ja` locale for end user.

- If user–defined characters have been registered in an existing font file using `fontedit`

  Use `sdtudc_extract` to extract the user–defined characters already registered and output them as a separate font file. Be sure the font file name is of the format UDC<*font size*>.bdf. Next, move the generated file to the directory in which user–defined character are being saved, start `sdtudctool`, and save the file.

  The folowings are the examples of the default environment for end user in `ja` locale.

  Example : When extracting user–defined characters from `gotm14.pcf`

  ```
  % sdtudc_extract gotm14.pcf > UDC14.bdf
  % bdftopcf -o UDC14.pcf UDC14.bdf
  % mkdir -p    ~/.Xlocale/ja/fonts/UDC/Bitmaps
  % mv UDC14.pcf    ~/.Xlocale/ja/fonts/UDC/Bitmaps
  % /usr/dt/bin/sdtudctool
  ```

- If user–defined characters have been registered in an existing font file using `type3creator` and `fontmanager`

  Use `sdtudc_extract` to extract the user–defined characters already registered and output them as a separate font file. Next, move the generated file to the directory in which user–defined character are being saved, start `sdtudctool`, and save the file.

  Example : When using `UDC.ps` created using `fontmanager`

  ```
  % sdtudc_extract UDC.ps
        UDC1.pfa
        UDC2.pfa
          . . .
  % mkdir -p   ~/.Xlocale/ja/fonts/UDC/Type1
  % mv UDC*.pfa   ~/.Xlocale/ja/fonts/UDC/Type1
  % /usr/dt/bin/sdtudctool
  ```

NAME

sdtudc_convert – User defined character conversion utility

SYNOPSIS

`/usr/dt/bin/sdtudc_convert` [`-f` *map_file*] [*text_file*]

AVAILABILITY

SUNWudct

DESCRIPTION

`sdtudc_convert` is a utility which converts code points within a text file to the code points specified in a map file and prints the results on the standard output.

`sdtudc_convert` is supported only in Japanese EUC (`ja`, `japanese`) locale under Solaris 2.6 and later.

NAME

sdtudc_extract – User defined character conversion utility

SYNOPSIS

/usr/dt/bin/sdtudc_extract [-f *map_file*] [-p *prefix*] *font_file*

AVAILABILITY

SUNWudct

DESCRIPTION

Beginning from Solaris 2.6, user–defined characters can be handled as separate font files without editing existing font files.

`sdtudc_extract` is a utility for producing glyph output from font file to the standard output. Beginning from Solaris 8, `sdtudc_extract_ps` is removed and its functionalities are merged into `sdtudc_extract`.

Supported types of font file and output are as follows.

- Bitmap font file (`.bdf`, `.bdf.Z`, `.pcf`, `.pcf.Z`)

  `sdtudc_extract` extracts glyphs that correspond to the code point specified in the map file, converts their code point, and outputs them to the standard output in bdf.

  When a bitmap font file is given to *font_file*, `sdtudc_extract` converts code point of codeset 1 9 ku – 15 ku to codeset 1 85 ku – 91 ku using map file listed below by default. You can change the range and target by modifying the map file.

  map file :

  a9a1,a9ff f5a1
  aaa1,aaff f6a1
  aba1,abff f7a1
  aca1,acff f8a1
  ada1,adff f9a1
  aea1,aeff faa1
  afa1,afff fba1

  When no map file is given, search for `sdtudc_map` is made at first in the directory `sdtudc_extract` was executed, and if it is not found there, `/usr/dt/config/$LANG/sdtudc_map` is accessed. If you wish to modify the map file, copy `/usr/dt/config/$LANG/sdtudc_map` to the working directory and edit the copy. For more informaiton about map file, See `sdtudc_map`(4).

- User-defined character font file (`.ps`) created with `type3creator` and `fontmanager`, and user-defined character font file for Windows9X/NT (`.ttf`)

  By default, `sdtudc_extract` extracts up to twenty Tyep1 font files with the prefix UDC per each 'ku' according to the total number of 'ku's in the regions for user-defined characters in Solaris 2.6. The following is the example where font file created with font manager is `font.ps`.

```
% sdtudc_extract font.ps
UDC1.pfa ... Done
UDC2.pfa ... Done
```

```
      :
UDC20.pfa ... Done
```

After the procedure described above, move the extracted Type1 font file to the fixed directory. For more information, see the description for the porting of user-defined character in `sdtudctool`(1)

OPTIONS

Available options are shown below.

`-f` *map_file* — Specifies the map file to use for conversion.

`-p` *prefix* — Specifies the prefix of Type1 font file to be extracted. If not specified, UDC is used as default.

EXIT STATUS

Exit status is returned as follows.

`0` — All input files were output normally.

`>0` — An error occurred.

FILES

`/usr/dt/config/$LANG/sdtudc_map`

SEE ALSO

`sdtudctool`(1), `sdtudc_convert`(1), `sdtudc_map`(4)

NAME

sdtudc_extract_ps – User defined character conversion utility

SYNOPSIS

/usr/dt/bin/sdtudc_extract_ps [-p *prefix*] *font_file*

AVAILABILITY

SUNWudct

DESCRIPTION

`sdtudc_extract_ps` is a utility which extracts user–defined characters created using `type3creator` and `fontmanager` as Type1 font format font files.

Beginning from Solaris 2.6, user–defined characters can be handled as separate font files without editing existing font files. They are defined in the following regions.

user–defined characters regions

ja/japanese	0xf5a1 - 0xfefe	JIS X 0208-1990 85 ku - 94 ku
	0x8ff5a1- 0x8ffefe	JIS X 0212-1990 85 ku - 94 ku

ja_JP.PCK	0xf040 - 0xf4fc	JIS X 0208-1990 85 ku - 94 ku
	0xf540 - 0xf9fc	JIS X 0212-1990 85 ku - 94 ku

So, if user–defined characters have registered by `type3creator` and `fontmanager`, and user want to re–use them under Solaris 2.6 and later, extract the the user–defined characters already registered and output them as a separate font file.

By default, `sdtudc_extract_ps` extracts until 20 font files which have UDC as prefix in every ku. This number is the total of the region for user–defined characters under Solaris 2.6. The following assumes the created font files by use of fontmanager to `font.ps`.

```
% sdtudc_extract_ps font.ps
UDC1.pfa . . . Done
UDC2.pfa . . . Done
     :
UDC20.pfa . . . Done
```

Next, move the extracted Type1 font files to the fixed directory (see `sdtudctool`(1)).

OPTIONS

Available options are shown below.

`-p` *prefix* — Specifies the prefix of the Type1 font file to be extracted. If it is not specified, `UDC` is used by default.

EXIT STATUS

Exit status is returned as follows.

`0` — All input files were output normally.

`>0` — An error occurred.

FILES `/usr/dt/config/$LANG/sdtudc_map`

SEE ALSO `sdtudctool`(1), `sdtudc_convert`(1), `sdtudc_extract`(1), `sdtudc_map`(4)

NAME

sdtudc_register – Intermediate utility to register user-defined characters

SYNOPSIS

`/usr/dt/bin/sdtudc_register`

AVAILABILITY

SUNWudct

DESCRIPTION

`sdtudc_register` is an intermediate utility that is launched from `sdtudctool` to register user-defined characters in a dictionary. `sdtudc_register` cannot function by itself. Be sure to launch `sdtudctool` first.

`sdtudc_register` displays a button to specify the kind of dictionary for registration and menu to change character code for display. A list of user-defined characters registered using `sdtudctool` is also displayed as code points and glyphs. One or more dictionaries can be specified at the same time.

`sdtudc_register` is currently supported only in Japanese locales (`ja`, `ja_JP.PCK`, `ja_JP.UTF-8`).

NAME

jistoeuc, jistosj, euctojis, euctosj, sjtojis, sjtoeuc – Code conversion between JIS, PC kanji, and Japanese EUC

SYNOPSIS

jistoeuc [-8] [-U] [*filename*...]

jistosj [-8] [-U] [*filename*...]

euctojis [-8] [-U] [*filename*...]

euctosj [-U] [*filename*...]

sjtojis [-8] [-U] [*filename*...]

sjtoeuc [-U] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

For Japanese language handling, the `jistoeuc` family provides conversion between different code standards. `command` [*filename* . . .] does the specified conversion on the contents of the input *filename*s and writes it to `stdout`.

If *filename* is not given, it reads and converts characters from the standard input.

`jistoeuc`	converts JIS to Japanese EUC
`jistosj`	converts JIS to PC kanji
`euctojis`	converts Japanese EUC to JIS
`euctosj`	converts Japanese EUC to PC kanji
`sjtojis`	converts PC kanji to JIS
`sjtoeuc`	converts PC kanji to Japanese EUC

OPTIONS

`-8`	With this option specified, the commands `jistoeuc`, `jistosj`, `sjtojis`, and `sjtoeuc`, can support JIS X 0201 (Half-Size Katakana). This 8-bit JIS code does not use ISO Shift-In and Shift-Out escape sequences.
`-U`	The output is not buffered (The default is buffered output).

SEE ALSO

`iconv`(1), `iconv_ja`(5)

NOTES

`jistoeuc` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in escape – `\E$B`, `\E$(B`, `\E$@`
JIS X 0212 shift-in escape – `\E$(D`
JIS X 0201 Roman shift-in escape – `\E(J`, `\E(H`
ASCII shift-in escape – `\E(B`

`euctojis` and `sjtojis` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in – `\E$B`

JIS X 0212 shift-in – `\E$(D` (except when `sjtojis` command is specified)
JIS X 0201 Roman shift-in – `\E(J`

This command does not check whether or not each code in the input file is correct. Conversion with PC kanji is not based on *TOG Japanese Vendors Council (TOG/JVC) Recommended Code Set Conversion Specification between Japanese EUC and Shift–JIS*. The `iconv(1)` utility provides these functions. See `iconv`(1) and `iconv_ja`(5) for more information.

BUGS

If JIS X 0212 character set is specified as input, `jistosj` and `euctosj` can not support the conversion correctly. `euctosj`, `sjtoeuc`, `jistosj`, and `sjtojis` can support conversion correctly only if JIS X 0208 1 ku – 84 ku is specified as input.

NAME

jistoeuc, jistosj, euctojis, euctosj, sjtojis, sjtoeuc – Code conversion between JIS, PC kanji, and Japanese EUC

SYNOPSIS

jistoeuc [-8] [-U] [*filename*...]

jistosj [-8] [-U] [*filename*...]

euctojis [-8] [-U] [*filename*...]

euctosj [-U] [*filename*...]

sjtojis [-8] [-U] [*filename*...]

sjtoeuc [-U] [*filename*...]

AVAILABILITY

SUNWjfpu

DESCRIPTION

For Japanese language handling, the `jistoeuc` family provides conversion between different code standards. `command` [*filename* . . .] does the specified conversion on the contents of the input *filename*s and writes it to `stdout`.

If *filename* is not given, it reads and converts characters from the standard input.

`jistoeuc`	converts JIS to Japanese EUC
`jistosj`	converts JIS to PC kanji
`euctojis`	converts Japanese EUC to JIS
`euctosj`	converts Japanese EUC to PC kanji
`sjtojis`	converts PC kanji to JIS
`sjtoeuc`	converts PC kanji to Japanese EUC

OPTIONS

`-8`	With this option specified, the commands `jistoeuc`, `jistosj`, `sjtojis`, and `sjtoeuc`, can support JIS X 0201 (Half-Size Katakana). This 8-bit JIS code does not use ISO Shift-In and Shift-Out escape sequences.
`-U`	The output is not buffered (The default is buffered output).

SEE ALSO

`iconv`(1), `iconv_ja`(5)

NOTES

`jistoeuc` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in escape – `\E$B`, `\E$(B`, `\E$@`
JIS X 0212 shift-in escape – `\E$(D`
JIS X 0201 Roman shift-in escape – `\E(J`, `\E(H`
ASCII shift-in escape – `\E(B`

`euctojis` and `sjtojis` can handle shift-in escape sequences for the following character sets:

JIS X 0208 shift-in – `\E$B`

JIS X 0212 shift-in – \E$(D (except when sjtojis command is specified)
JIS X 0201 Roman shift-in – \E(J

This command does not check whether or not each code in the input file is correct. Conversion with PC kanji is not based on *TOG Japanese Vendors Council (TOG/JVC) Recommended Code Set Conversion Specification between Japanese EUC and Shift–JIS*. The iconv(1) utility provides these functions. See iconv(1) and iconv_ja(5) for more information.

BUGS

If JIS X 0212 character set is specified as input, jistosj and euctosj can not support the conversion correctly. euctosj, sjtoeuc, jistosj, and sjtojis can support conversion correctly only if JIS X 0208 1 ku – 84 ku is specified as input.

NAME

uum – Kana-Kanji conversion front end processor

SYNOPSIS

/usr/bin/uum [-J | -U | -S | -T] [-j | -u | -s | -t] [-h | -H] [-x | -X] [-k *filename*] [-c *filename*] [-r *filename*] [-D *hostname*] [-n *username*] [-l *number*]

DESCRIPTION

uum command provides a Japanese language I/O environment on the terminal. When started, uum searches and reads initialization files in the following order.

1. Files under the directory set in the environment variable UUMRC
2. $HOME/.uumrc
3. /etc/lib/locale/ja/wnn/ja/uumrc
4. /usr/lib/locale/ja/wnn/ja/uumrc

uum can connect to the Kana-Kanji conversion server jserver running on the machine from which uum started or another machine over the network. Even when uum cannot connect to the server, it can start to complete certain operations that do not require the communication with the Kana-Kanji conversion server. When a conversion key is pressed, an attempt will be made to automatically connect to jserver if uum has not already connected to it.

OPTIONS

The following options are available.

-H — Starts with the conversion status ON (default).

-h — Starts with the conversion status OFF.

-U — Uses UJIS code on the application (the virtual terminal) (default).

-J — Uses JIS code on the application (the virtual terminal).

-S — Uses shift-JIS code on the application (the virtual terminal).

-T — Uses UTF-8 code on the application (the virtual terminal).

-u — Uses UJIS code on the terminal.(default)

-j — Uses JIS code on the terminal.

-s — Uses shift-JIS code on the terminal.

-t — Uses UTF-8 code on the terminal.

-X — Starts with terminal flow control ON (default).

-x — Starts with terminal flow control OFF.

-k *filename* — Specifies the key binding definition file. If omitted, the difinition file is searched in the following order:

1. File name specified with the setuumkey entry in the initialization file uumrc
2. /usr/lib/locale/ja/wnn/ja/uumkey

-c *filename* — Specifies the key code conversion table file. If omitted, the table file is searched in the following order:

1. File name specified with the `setconvkey` entry in the initialization file `uumrc`
2. `/usr/lib/locale/ja/wnn/cvt_key_tbl`

`-r` *filename* — Specifies the mode definition file for roman character-Kana conversion (see wnn_automaton(4)). If a directory name is given, the mode file under the directory will be used as the mode file. If omitted, the difinition file is searched in the following order:

1. File name specified with the `setrkfile` entry in the initialization file `uumrc`
2. `/usr/lib/locale/ja/wnn/ja/rk/mode`

`-l` *number* — Specifies the number of lines to be used in Kana-Kanji conversion (larger than 0, and smaller than the number of display lines - 1). The default is 1.

`-D` *hostname* — Specifies the host name of the Kana-Kanji conversion server (`jserver`). If omitted, the conversion server is searched in the following order:

1. Environment variable `JSERVER`
2. `localhost`
3. UNIX domain socket

`-n` *username* — Specifies the user name to use for the environment name for `Wnn6`. If omitted, the user name is searched in the following order:

1. Environment variable `WNNUSER`
2. User name of the user who started `uum`

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncu

SEE ALSO

`jserver`(1M), `uumkey`(4), `uumrc`(4), `wnn_automaton`(4), `wnn_cvt_key_tbl`(4), `wnn_mode`(4)

NOTES

`uum` uses one virtual terminal and cannot be started unless a virtual terminal can be obtained. It does not be started either if the initialization file, key binding definition file, or roman character-Kana conversion mode definition table file cannot be found.

Dictionaries are administered under the envirnonment name of Wnn6.

NAME

Wnn6 – Wnn6 Japanese language input system

DESCRIPTION

Wnn6 provides a method to input Japanese language in a desktop environment or on a terminal.

You need to use `htt`(1) or `xjsi`(1) to input Japanese language in a desktop environment, and `uum`(1) on a serial-connected terminal.

If you are logged in to the Japanese desktop, you can use Wnn6 as Japanese language input system by default. If you have changed the default to use another Japanese language input system, you can use Wnn6 by running `wnn6setup`(1).

You can choose any Japanese language input method you like (see the GUI utility `wnnenvutil`(1)).

You can also register and use your own Kana-Kanji conversion dictionary (see the GUI utility `wnndictutil`(1)).

Wnn6 is based on a client-server model, and the server supports multiple clients (users) at the same time.

Dictionaries for Kana-Kanji conversion can be managed centrally on a server.

SEE ALSO

`htt`(1), `xjsi`(1), `uum`(1), `wnn6setup`(1), `wnnatod`(1), `wnnbushu`(1), `wnndictutil`(1), `wnndtoa`(1), `wnnenvutil`(1), `wnnotow`(1), `wnnstat`(1), `wnntouch`(1),

`jserver`(1M), `wnnaccess`(1M), `wnnds`(1M), `wnnkill`(1M), `wnnoffline`(1M), `wnnudmerge`(1M), `dpkeyserv`(1M), `dpkeystat`(1M),

`wnnenvrc`(4), `wnnhosts`(4), `jserverrc`(4), `uumkey`(4), `uumrc`(4), `wnn_2A_CTRL`(4), `wnn_2B_ROMKANA`(4), `wnn_automaton`(4), `wnn_cvt_key_tbl`(4), `wnn_cvt_xim_tbl`(4), `wnn_hinsi.data`(4), `wnn_mode`(4), `wnn_serverdefs`(4), `wnn_ximrc`(4)

Japanese Input System Summary & Transition

Wnn6 User's Guide

Wnn6 Advanced User's and System Administrator's Guide

NAME

wnn6setup – Set up Wnn6 for Japanese input in X environment

SYNOPSIS

wnn6setup

DESCRIPTION

wnn6setup sets up Wnn6 for use as Japanese input system in X environment, to be started when you log in. Values set by wnn6setup will be in effect when you log in next time.

wnn6setup modifies the following files.

$HOME/.dtprofile
A script that is run when you log in to the Common Desktop Environment (CDE). Lines to launch htt(1) for Wnn6 are added.

$HOME/.Xlocale/ja/app-defaults/Htt
A resource file that htt(1) refers. Settings to use xjsi(1) as interface module are added.

If $HOME/.dtprofile already exist, and atok12setup(1) provided with Japanese Solaris 2.6 is used to configure Japanese input system, wnn6setup overrides such settings to use Wnn6 as Japanese input system.

Lines to set up Wnn6 are placed between comment lines in $HOME/.dtprofile as follows.

```
###== - Generated by wnn6setup to launch japanese XIM. == BEGIN == -==###
. . .
New settings to start a Japanese input system . . .
. . .
###== - Generated by wnn6setup to launch japanese XIM. == END == -==###
```

The above setting lines are modified by executing atok12setup(1). That is, if you edit the setting lines between comment lines, then execute atok12setup(1), changes to these lines will be lost. You should not make any changes to these lines.

FILES

$HOME/.dtprofile

$HOME/.Xlocale/ja/app-defaults/Htt

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncx

SEE ALSO

atok12setup(1), dtlogin(1), htt(1), xjsi(1)

NOTES

If you edit $HOME/.dtprofile to set up the Japanese input system, and execute the wnn6setup command, you may not be able to use Wnn6 when you log in. In this case, remove the setting lines you added in $HOME/.dtprofile for the Japanese input system.

NAME

wnnatod – Convert an EUC text dictionary to a binary dictionary

SYNOPSIS

/usr/bin/wnnatod [-s *num*] [-R] [-S] [-U] [-r] [-N] [-n] [-P *filename*] [-p *filename*] [-I] [-e] [-h *filename*] *binary_dictionary_filename*

DESCRIPTION

wnnatod reads a Japanese EUC text dictionary from the standard input, converts it to a binary dictionary and writes it to the specified *binary_dictionary_filename*.

OPTIONS

The following options are available.

-s *num*	Specifies the amount of memory to allocate (in words). *num* should be a little over the number of words in the dictionary. Normally you do not need to specify this option. The default is 70,000. If wnnatod fails, notifying memory shortage, retry the command with -s option.
-R	Converts the EUC text dictionary to a reverse-searchable binary dictionary (default).
-S	Converts the EUC text dictionary to a fixed-format dictionary.
-U	Converts the EUC text dictionary to an editable dictionary.
-r	Reverses the order of Kana and Kanji when converting the EUC text dictionary.
-N	Sets the dictionary password to "*".
-n	Sets the frequency password to "*".
-P *filename*	Specifies the file name of the dictionary password.
-p *filename*	Specifies the file name of the frequency password.
-I	Creates a system dictionary.
-e	Registers an entry's reading (Hiragana) as word in the binary dictionary if the reading and the word are the same (that is, the word consists of only Hiragana). With this option, you cannot convert a text dictionary to a reverse-searchable binary dictionary.
-h *filename*	Specifies the file name that contains part of speech information.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncu

SEE ALSO

wnndictutil(1), wnndtoa(1), wnnotow(1), wnntouch(1)

NAME

wnnbushu – Wnn6 radical input utility

SYNOPSIS

/usr/openwin/bin/wnnbushu `-s -D` *jserver_name* `[-h] [-b` *filename*`]` `[-f` *filename*`]`

DESCRIPTION

The `wnnbushu` utility provides a feature of inputting radical to `xjsi`. Normally `wnnbushu` is launched by xjsi. The utility searches for Kanji based on radical names and total stroke counts, lists the candidates, and allows you to select one.

OPTIONS

The following options are available.

`-s` Allows the utility to run on a stand-alone basis, not from `xjsi`.

`-D` *jserver_name* Specifies the host name of `jserver` to search for Kanji based on radical names or total stroke counts. This option cannot be omitted.

`-h` Outputs online help messages.

`-b` *filename* Specifies the file name of the radical dictionary to search for Kanji based on radical names or total stroke counts. If this option is not specified, the `/usr/lib/locale/ja/wnn/ja/dic/bushu/bushu.dic` file is used.

`-f` *filename* Specifies the file name of the attribute words dictionary to search for Kanji based on radical names or total stroke counts. If this option is not specified, the `/usr/lib/locale/ja/wnn/ja/dic/bushu/bushu.fzk` file is used.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncx

SEE ALSO

`xjsi`(1)

NAME

wnndictutil – Dictionary utility

SYNOPSIS

/usr/openwin/bin/wnndictutil [-D *jserver_name*] [-E *user_name*] [*toolkit options*]

DESCRIPTION

wnndictutil provides dictionary editing features for Wnn6 such as:

- Word registration (single words or batch)
- Word deletions (single words or batch)
- Word attribute information (usage, comments, frequency)
- Dictionary merge
- User dictionary, learning feature, restoring

OPTIONS

The following options are available.

-D *jserver_name* Specifies the host name on which Kana–Kanji conversion server jserver is launched. If this option is omitted, the following are searched in the order listed to determine the Kana Kanji conversion server.

1. Environment variable JSERVER
2. Resource Dictutil.serverName
3. jserver (host name)
4. UNIX domain socket

-E *user_name* Specifies the user name to be used as the environment name of Wnn6. If this option is omitted, the following are searched in the order listed to determine the user name.

1. Environment variable WNNUSER
2. Resource Wnndictutil.userName
3. Name of the user who run wnndictutil

toolkit options Specifies the standard options for X Toolkit initialization functions.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncx

SEE ALSO

wnnatod(1), wnndtoa(1), wnnotow(1), xjsi(1)

NAME

wnndtoa – Convert a binary dictionary to an EUC text dictionary

SYNOPSIS

/usr/bin/wnndtoa [-n] [-s] [-e | -E] [-h *filename*]
binary_dictionary_filename [*frequency_filename. ..*]

DESCRIPTION

wnndtoa converts the specified binary dictionary *binary_dictionary_filename* to a text dictionary file (in Japanese EUC) and writes it to the standard output.

If you specify one or more frequency files (*frequency_filename*), the frequency information affects the text dictionary.

OPTIONS

The following options are available.

-n — Sorts entries by reading (in the order of long sound symbol, Hiragana (full-width characters), and alphanumerics (ASCII characters)).

-s — Adds serial numbers to the entries.

-e — Expands entries to special representation. For example, space characters and tabs are expanded to octal representation (default).

-E — Does not expand entries to special representation. For example, space characters and tabs are not expanded to octal representation

-h *filename* — Specifies the part of speech file name. The default is /usr/lib/locale/ja/wnn/ja/hinsi.data .

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncu

SEE ALSO

wnnatod(1), wnndictutil(1), wnn_hinsi.data(4)

NAME

wnnenvutil – Environment setting utility

SYNOPSIS

`/usr/openwin/bin/wnnenvutil`

DESCRIPTION

`wnnenvutil` provides the customization feature of the operating environment in Wnn6. You can easily set up the environment without special knowledge.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncx

SEE ALSO

`uum`(1), `xjsi`(1), `jserver`(1M), `wnnds`(1M)

NAME

wnnotow – User dictionary converter

SYNOPSIS

/usr/bin/wnnotow [-i *input_file*] [-f *format_file*] [-o *output_file*] [-l *log_file*] [-h *frequency_value*]

DESCRIPTION

wnnotow converts a text dictionary created with another Japanese input system (ATOK7, ATOK8, cs00,EGBRIDGE, VJE-Delta) to the Wnn6 text dictionary format.

OPTIONS

The following options are available.

-i *input_file* Specifies the name of the dictionary text to be converted. The text code must be EUC. The standard input is read if this option is omitted.

-f *format_file* Specifies the name of the file defining the format used in the input file. The text code must be EUC. The format files corresponding to each FEP are as follows.

Directory: /usr/lib/locale/ja/wnn/ja/otow.format/

Files: atok7-wnn6.fmt (ATOK7)
atok8-wnn6.fmt(ATOK8)
cs00-wnn6.fmt(cs00) egbridge-wnn6.fmt (EGBRIDGE) vje-wnn6.fmt(VJE–Delta)

The ATOK7 format is used if this option is omitted.

-o *output_file* Writes the conversion results in the specified file. The output character code must be Japanese EUC. The output is written to the standard output if this option is omitted.

-l *log_file* Specifies the name of the log file to save error information if unconvertable words are found. The output character code must be Japanese EUC. No log file will be created if this option is omitted.

-h *frequency_value* Specifies the frequency value to applied for the output file. A frequency value of 1 will be applied if this option is omitted.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncu

SEE ALSO

wnnatod(1), wnndictutil(1)

NAME

wnnstat – Print the status of Wnn6 Kana-Kanji conversion server

SYNOPSIS

/usr/bin/wnnstat [-w] [-e] [-E] [-f] [-F] [-d] [-D] [-J | -U | -S | -T] [-L *language*] [*hostname*]

DESCRIPTION

wnnstat prints the status of Wnn6 Kana-Kanji conversion server (jserver) running on the specified host (*hostname*). If the host name is omitted, one from which wnnstat was launched is assumed.

OPTIONS

The following options are available.

-w Prints the user names, host names, socket numbers, and environment numbers.

-e Prints the environment numbers, environment names, and number of references.

-E Prints the environment numbers, environment names, number of references, auxiliary words, number of dictionaries (dictionary numbers) and file names.

-f Prints the dictionary file identifier (Fid), type, location, number of references, and file names.

-F Prints the dictionary file identifier (Fid), type, location, number of references, and file names.

-d Prints the dictionary numbers, types, nicknames, the dictionary file identifier (Fid), and file names.

-D Prints the dictionary numbers, types, number of words, update disables, frequency file update disables, usage disables, priorities, [nicknames], the dictionary file identifier (Fid), file names, and [(frequency: frequency_file_name)] [password, (frequency_password)].

-U Prints in Japanese EUC (UJIS).

-J Prints in JIS code.

-S Prints in SJIS code.

-T Prints in UTF-8 code.

-L *language* Prints the status of the server that supports the language specified with *language* . -L ja should be specified for Solaris (Japanese version) releases.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncu

SEE ALSO

jserver(1M)

NAME

wnntouch – Rewrite and format the file header according to the inode.

SYNOPSIS

`/usr/bin/wnntouch` *binary_filename ...*

DESCRIPTION

`wnntouch` rewrites the header of the binary dictionary or auxiliary word specified with *binary_filename* and formats it according to the inode information.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncu

SEE ALSO

`wnnatod`(1)

NAME

xjsi – Wnn6 Kana-Kanji conversion server/htt interface module

SYNOPSIS

htt -if xjsi -so -nosm [*htt_options*...]

DESCRIPTION

xjsi is an interface module between Wnn6 Kana–Kanji conversion server jserver and X input server htt. xjsi is read as shared library by htt.

SETUP FILES

The following setup files are read by xjsi when it is invoked:

ximrc — Sets the X Input Method (XIM) Kana–Kanji conversion interface. xjsi determines the path name for ximrc in the following order.

1. ximrcName resource
2. XIMRC environment variable
3. $HOME/.Wnn6/ximrc
4. /etc/lib/locale/ja/wnn/ximrc
5. /usr/lib/locale/ja/wnn/ximrc

uumrc — Sets the standard Kana–Kanji conversion interface. xjsi determines the path name for uumrc in the following order.

1. setuumrc entry in ximrc file
2. @HOME/.Wnn6/uumrc
3. /etc/lib/locale/ja/wnn/ja/uumrc
4. /usr/lib/locale/ja/wnn/ja/uumrc

wnnenvrc — Sets the parameter Kana–Kanji conversion dictionary and conversion. xjsi determines the path name for wnnenvrc in the following order.

1. setconvenv entry in uumrc file
2. $HOME/.Wnn6/wnnenvrc
3. /etc/lib/locale/ja/wnn/ja/wnnenvrc
4. /usr/lib/locale/ja/wnn/ja/wnnenvrc

uumkey — Sets the key assignment. xjsi determines the path name for uumkey in the following order.

1. setuumkey entry in uumrc
2. /etc/lib/locale/ja/wnn/ja/uumkey
3. /usr/lib/locale/ja/wnn/ja/uumkey

rk/mode — Sets the mode definition table for Roman characters–Kana conversion. xjsi determines the path name for rk/mode in the following order.

1. setrkfile entry in uumrc file
2. /usr/lib/locale/ja/wnn/ja/rk/mode

cvt_xim_tbl — Sets the X key code conversion table. xjsi determines the path name for X key code conversion table in the following order.

1. cvtximName resource
2. /usr/lib/locale/ja/wnn/cvt_xim_tbl

RESOURCES

`xjsi` provides the following resource names and resource classes under the name `xjsi` and the class name `Xjsi`.

`serverName` (`ServerName`)
Specifies the name of Kana–Kanji conversion server machine. `xjsi` determines the Kana–Kanji conversion server to connect in the following order.

1. Resource setting
2. `JSERVER` environment variable
3. `setconvenv` entry in `wnnenvrc`
4. local host
5. UNIX domain socket

`userName` (`UserName`)
Specifies the user name as the Wnn6 environment name. `xjsi` determines the user name used for the Wnn6 environment name in the following order.

1. Resource specification
2. Environment variable WNNUSER
3. Name of the user who launched htt

`ximrcName` (`XimrcName`)
Specifies the path name of the `xjsi` initialization file `ximrc`. For the details of `ximrc` file, see `wnn_ximrc`(4) man page.

`cvtximName` (`CvtximName`)
Specifies the path name of the X key code conversion table.

When using `xjsi` in the multi-screen system, the following resources are set independently for each screen. The sub-name and sub-class name for each screen is `screen`*N* and `Screen`*N* respectively. Set *N* to a screen number.

`foreground` (`Foreground`)
Specifies the color of text. The default is `black`.

`background` (`Background`)
Specifies the color of background. The default is `white`.

`borderColor` (`BorderColor`)
Specifies the color of borders. The default is `black`.

`fontSet` (`FontSet`)
Specifies the list of font names. The format of the list is *fontname* { , *fontname*}.

FILES

`/usr/openwin/lib/locale/ja/xjsi` interface module

`ximrc` Setting file for XIM Kana–Kanji conversion interface

`uumrc` Setting file for the standard Kana–Kanji conversion interface

`wnnenvrc` Parameter setting file for Kana–Kanji conversion dictionary and conversion

`uumkey` Setting for key assignment

`cvt_xim_tbl` X key code conversion table

`mode` Setting file for Roman characters–Kana conversion

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjwncx

SEE ALSO

`htt`(1), `wnn6setup`(1) `wnn_ximrc`(4), `uumrc`(4), `wnnenvrc`(4), `uumkey`(4), `wnn_cvt_xim_tbl`(4), `wnn_mode`(4), `wnn_automaton`(4)