JFP Reference Manual 7 : Device and Network Interfaces

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

Part No: 817–0649–10
December 2002

021210@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 7 : Device and Network Interfaces

NAME

Intro_jfp, intro_jfp – introduction to JFP special files

DESCRIPTION

This section describes various device and network interfaces available on the system. The types of interfaces described include character and block devices, STREAMS modules, network protocols, file systems, and ioctl requests for driver subsystems and classes.

This section contains the following major collections:

(7I) This section describes ioctl requests which apply to a class of drivers or subsystems. Ioctl requests relevant to only a specific device are described on the man page for that device. The page for the device in question should still be examined for exceptions to the ioctls listed in section 7I.

(7M) This section describes STREAMS modules. Note that STREAMS drivers are discussed in section 7D. `streamio`(7I) contains a list of ioctl requests used to manipulate STREAMS modules and interface with the STREAMS framework. Ioctl requests specific to a STREAMS module will be discussed on the man page for that module.

SEE ALSO

`add_drv`(1M), `rem_drv`(1M), `intro`(2), `ioctl`(2), `socket`(3SOCKET), `driver.conf`(4), `intro`(7), `arp`(7P), `icmp`(7P), `inet`(7P), `ip`(7P), `mtio`(7I), `st`(7D), `streamio`(7I), `tcp`(7P), `udp`(7P)

System Administration Guide: IP Services

STREAMS Programming Guide

Writing Device Drivers

LIST

Name	Description
`Intro_jfp`(7)	introduction to JFP special files
`jaio`(7I)	interface to Japanese I/O modules
`jconv7`(7M)	code conversion STREAMS module (7-bit JIS/Japanese EUC)
`jconv8`(7M)	code conversion STREAMS module (8-bit JIS/Japanese EUC)
`jconvrs`(7M)	code conversion STREAMS module (Japanese EUC/PC kanji)
`jconvru`(7M)	code conversion STREAMS module (Japanese EUC/UTF—8)
`jconvs`(7M)	code conversion STREAMS module (PC kanji/Japanese EUC)
`jconvu`(7M)	code conversion STREAMS module (UTF-8/Japanese EUC)

NAME

jaio – interface to Japanese I/O modules

SYNOPSIS

```
#include <sys/jaio.h>
```

DESCRIPTION

This interface is implemented in pushable STREAMS modules that handle Japanese I/O. These are used to query or change the third character of the three character ISO escape sequence for announcing data.

These calls take an argument, which is expected to be a pointer to a "kioc" structure, defined in the header file `sys/jaio.h`, as follows:

```
struct kioc {
    char ki;
    char ko;
}:
```

IOCTLS

`JA_SKIOC` This call changes the 3rd characters of JIS Kanji-In/Out ISO announcement sequences.

`JA_GKIOC` This call returns the 3rd characters of JIS Kanji-In/Out ISO announcement sequences in the "kioc" structure.

EXAMPLES

EXAMPLE 1 The following is an example of using these ioctls:

```
#include <sys/jaio.h>
struct kioc ja_kio;
struct strioctl cmd;
     .
     .
ja_kio.ki = '[input escape char]';
ja_kio.ko = '[output escape char]';
cmd.ic_cmd = JA_SKIOC;
cmd.ic_timout = 0;
cmd.ic_len = sizeof(kioc_t);
cmd.ic_dp = (char *)&ja_kio;
ioctl(0, I_STR, &cmd);
     .
     .
```

FILES

`/sys/jaio.h`

SEE ALSO

`jtty`(1), `jconv7`(7M), `jconv8`(7M)

NAME

jconv7 – code conversion STREAMS module (7-bit JIS/Japanese EUC)

SYNOPSIS

```
#include <sys/types.h>

#include <sys/stropt.h>

#include <sys/conf.h>

ioctl(fd, I_PUSH, "jconv7");
```

DESCRIPTION

`jconv7` is a STREAMS module that is available to be pushed onto a stream. Usually, this module has to be pushed onto a stream between a raw device such as `ptem`(7M) and terminal line discipline module such as `ldterm`(7M).

`jconv7` has to be pushed when you set 7–bit JIS terminal and control Japanese EUC data. It converts up stream for 7–bit JIS code into Japanese EUC and passes high module. It also converts down stream for Japanese EUC into 7–bit JIS code and passes low module.

IOCTLS

`jconv7` processes the following ioctls. `JA_SKIOC` and `JA_GKIOC` are specified pointer to the next structure as argument:

```
struct kioc {
        char ki;
        char ko;
};
```

`JA_SKIOC`	Change the 3rd character of JIS kanji and ASCII indication escape sequence.
`JA_GKIOC`	Return the 3rd character of current JIS kanji and ASCII indication escape sequence.
`EUC_OXLON`	Start performing code conversion between 7–bit JIS and Japanese EUC for I/O stream.
`EUC_OXLOFF`	Stop performing code conversion between 7–bit JIS and Japanese EUC for I/O stream.

SEE ALSO

`jtty`(1), `setterm`(1), `stty`(1), `streamio`(7I), `jconv8`(7M), `jconvrs`(7M), `jconvru`(7M),`jconvs`(7M), `jconvu`(7M), `ldterm`(7M), `ptem`(7M)

NOTES

When you use `jconv7` with `jconvrs`(7M) or `jconvru`(7M) at once and ‘raw’ is specified by `stty`(1), code convert function automatically become off without specification by `EUC_OXLON` / `EUC_OXLOFF`. By default, character set indication escape sequences as follows. The end character may be changed by `JA_SKIOC`.

- ASCII indication: ESC 2/8 4/10
- Character indication: ESC 2/4 4/2
- Convert with JIS X 0201 Kana: SI/SO

`jconv7` does not support Kanji code for Information Interchange (secondary kanji set) provided by JIS X 0212–1990.

When `jconv7` is in use, `csh`(1) does not work properly for filename additional function.

NAME

jconv8 – code conversion STREAMS module (8-bit JIS/Japanese EUC)

SYNOPSIS

```
#include <sys/types.h>

#include <sys/stropt.h>

#include <sys/conf.h>

ioctl(fd, I_PUSH, "jconv8");
```

DESCRIPTION

`jconv8` is a STREAMS module that is available to be pushed onto a stream. Usually, this module has to be pushed onto a stream between a raw device such as `ptem`(7M) and terminal line discipline module such as `ldterm`(7M).

`jconv8` has to be pushed when you set 8–bit JIS terminal and control Japanese EUC data. It converts up stream for 8–bit JIS code into Japanese EUC and passes high module. It also converts down stream for Japanese EUC into 8–bit JIS code and passes low module.

IOCTLS

`jconv8` processes the following ioctls. `JA_SKIOC` and `JA_GKIOC` are specified pointer to the next structure as argument:

```
struct kioc {
        char ki;
        char ko;
};
```

`JA_SKIOC`	Change the 3rd character of JIS kanji and ASCII indication escape sequence.
`JA_GKIOC`	Return the 3rd character of current JIS kanji and ASCII indication escape sequence.
`EUC_OXLON`	Start performing code conversion between 8–bit JIS and Japanese EUC for I/O stream.
`EUC_OXLOFF`	Stop performing code conversion between 8–bit JIS and Japanese EUC for I/O stream.

SEE ALSO

`jtty`(1), `setterm`(1), `stty`(1), `streamio`(7I), `jconv7`(7M), `jconvrs`(7M), `jconvru`(7M), `jconvs`(7M), `jconvu`(7M), `ldterm`(7M), `ptem`(7M)

NOTES

When you use `jconv8` with `jconvrs`(7M) or `jconvru`(7M) at once and ‘raw’ is specified by `stty`(1), code convert function automatically become off without specification by `EUC_OXLON` / `EUC_OXLOFF`. By default, character set indication escape sequences as follows. The end character may be changed by `JA_SKIOC`.

- ASCII indication: ESC 2/8 4/10
- Character indication: ESC 2/4 4/2
- Convert with JIS X 0201 Kana: LS1/LS0

`jconv8` does not support Kanji code for Information Interchange (secondary kanji set) provided by JIS X 0212–1990.

When `jconv8` is in use, `csh`(1) does not work properly for filename additional function.

NAME

jconvrs – code conversion STREAMS module (Japanese EUC/PC kanji)

SYNOPSIS

```
#include <sys/types.h>

#include <sys/stropt.h>

#include <sys/conf.h>

ioctl(fd, I_PUSH, "jconvrs");
```

DESCRIPTION

`jconvrs` is a STREAMS module that is available to be pushed onto a stream. Usually, this module has to be pushed onto a stream between terminal circuit module such as `ldterm`(7M) and STREAMS compatible module such as `ttcompat`(7M).

`jconvrs` has to be pushed when you operate PC kanji data. It converts up stream for Japanese EUC into PC kanji code and passes high module. It also converts down stream for PC kanji code into Japanese EUC and passes low module. `jconvrs` controls terminal circuit under PCK environment by using with `jconvs`(7M).

IOCTLS

`jconvrs` processes the following ioctls:

`EUC_OXLON` Start performing code conversion between Japanese EUC and PC kanji for I/O stream.

`EUC_OXLOFF` Stop performing code conversion between Japanese EUC and PC kanji for I/O stream.

SEE ALSO

`jtty`(1), `setterm`(1), `stty`(1), `PCK`(5), `streamio`(7I), `jconv7`(7M), `jconv8`(7M), `jconvru`(7M), `jconvs`(7M), `jconvu`(7M), `ldterm`(7M), `ptem`(7M)

NOTES

When you use `jconvrs` with `jconvs`(7M), `jconvu`(7M), `jconv7`(7M), or `jconv8`(7M) at once and 'raw' is specified by `stty`(1), code convert function automatically become off without specification by `EUC_OXLON` / `EUC_OXLOFF`.

`jconvrs` supports PC kanji/Japanese EUC code conversion under *TOG Japanese Vendors Council (TOG/JVC) Recommended Code Set Conversion Specification between Japanese EUC and Shift–JIS*.

NAME

jconvru – code conversion STREAMS module (Japanese EUC/UTF—8)

SYNOPSIS

```
#include <sys/types.h>

#include <sys/stropt.h>

#include <sys/conf.h>

ioctl(fd, I_PUSH, "jconvru");
```

DESCRIPTION

`jconvru` is a STREAMS module that is available to be pushed onto a stream. Usually, this module has to be pushed onto a stream between terminal circuit module such as `ldterm`(7M) and STREAMS compatible module such as `ttcompat`(7M).

`jconvru` has to be pushed when you operate UTF-8 data. It converts up stream for Japanese EUC into UTF-8 and passes high module. It also converts down stream for UTF-8 into Japanese EUC and passes low module.

IOCTLS

`jconvru` processes the following ioctls:

`EUC_OXLON` Start performing code conversion between Japanese EUC and UTF-8 for I/O stream.

`EUC_OXLOFF` Stop performing code conversion between Japanese EUC and UTF-8 for I/O stream.

SEE ALSO

`jtty`(1), `setterm`(1), `stty`(1), `streamio`(7I), `jconv7`(7M), `jconv8`(7M), `jconvrs`(7M), `jconvs`(7M), `jconvu`(7M), `ldterm`(7M), `ptem`(7M)

NOTES

When you use `jconvru` with `jconvu`(7M), `jconvs`(7M), `jconv7`(7M), or `jconv8`(7M) at once and 'raw' is specified by `stty`(1), code convert function automatically become off without specification by `EUC_OXLON` / `EUC_OXLOFF`.

NAME

jconvs – code conversion STREAMS module (PC kanji/Japanese EUC)

SYNOPSIS

```
#include <sys/types.h>

#include <sys/stropt.h>

#include <sys/conf.h>

ioctl(fd, I_PUSH, "jconvs");
```

DESCRIPTION

`jconvs` is a STREAMS module that is available to be pushed onto a stream. Usually, this module has to be pushed onto a stream between a raw device such as `ptem`(7M) and terminal line discipline module such as `ldterm`(7M).

`jconvs` has to be pushed when you set PC kanji terminal. It converts up stream for PC kanji code into Japanese EUC and passes high module. It also converts down stream for Japanese EUC into PC kanji code and passes low module. `jconvs` controls terminal line discipline under PCK environment by using with `jconvrs`(7M).

IOCTLS

`jconvs` processes the following ioctls:

`EUC_OXLON`	Start performing code conversion between PC kanji and Japanese EUC for I/O stream.
`EUC_OXLOFF`	Stop performing code conversion between PC kanji and Japanese EUC for I/O stream.

SEE ALSO

`jtty`(1), `setterm`(1), `stty`(1), `PCK`(5), `streamio`(7I), `jconv7`(7M), `jconv8`(7M), `jconvrs`(7M), `jconvru`(7M), `jconvu`(7M), `ldterm`(7M), `ptem`(7M)

NOTES

When you use `jconvs` with `jconvrs`(7M) or `jconvru`(7M) at once and 'raw' is specified by `stty`(1), code convert function automatically become off without specification by `EUC_OXLON` / `EUC_OXLOFF`.

`jconvs` supports PC kanji/Japanese EUC code conversion under *TOG Japanese Vendors Council (TOG/JVC) Recommended Code Set Conversion Specification between Japanese EUC and Shift–JIS*.

NAME

jconvu – code conversion STREAMS module (UTF-8/Japanese EUC)

SYNOPSIS

```
#include <sys/types.h>

#include <sys/stropt.h>

#include <sys/conf.h>

ioctl(fd, I_PUSH, "jconvu");
```

DESCRIPTION

`jconvu` is a STREAMS module that is available to be pushed onto a stream. Usually, this module has to be pushed onto a stream between a raw device such as `ptem`(7M) and terminal line discipline module such as `ldterm`(7M).

`jconvu` has to be pushed when you set UTF-8 terminal. It converts up stream for UTF-8 into Japanese EUC and passes high module. It also converts down stream for Japanese EUC into UTF-8 and passes low module.

IOCTLS

`jconvu` processes the following ioctls:

`EUC_OXLON` Start performing code conversion between UTF-8 and Japanese EUC for I/O stream.

`EUC_OXLOFF` Stop performing code conversion between UTF-8 and Japanese EUC for I/O stream.

SEE ALSO

`jtty`(1), `setterm`(1), `stty`(1), `streamio`(7I), `jconv7`(7M), `jconv8`(7M), `jconvrs`(7M), `jconvru`(7M), `jconvs`(7M), `ldterm`(7M), `ptem`(7M)

NOTES

When you use `jconvu` with `jconvrs`(7M) or `jconvru`(7M) at once and 'raw' is specified by `stty`(1), code convert function automatically become off without specification by `EUC_OXLON` / `EUC_OXLOFF`.