NAME

dnvlist_get, **dnvlist_take** **-** API for getting name/value pairs. Nonexistent pairs do not raise an error.

LIBRARY

Name/value pairs library (libnv, -lnv)

SYNOPSIS

#include <sys/dnv.h>

bool
dnvlist_get_bool(*const nvlist_t *nvl*, *const char *name*, *bool defval*);

uint64_t
dnvlist_get_number(*const nvlist_t *nvl*, *const char *name*, *uint64_t defval*);

*char **
dnvlist_get_string(*const nvlist_t *nvl*, *const char *name*, *const char *defval*);

*nvlist_t **
dnvlist_get_nvlist(*const nvlist_t *nvl*, *const char *name*, *nvlist_t *defval*);

int
dnvlist_get_descriptor(*const nvlist_t *nvl*, *const char *name*, *int defval*);

*void **
dnvlist_get_binary(*const nvlist_t *nvl*, *const char *name*, *size_t *sizep*, *void *defval*, *size_t defsize*);

bool
dnvlist_take_bool(*const nvlist_t *nvl*, *const char *name*, *bool defval*);

uint64_t
dnvlist_take_number(*const nvlist_t *nvl*, *const char *name*, *uint64_t defval*);

*char **
dnvlist_take_string(*const nvlist_t *nvl*, *const char *name*, *const char *defval*);

*nvlist_t **
dnvlist_take_nvlist(*const nvlist_t *nvl*, *const char *name*, *nvlist_t *defval*);

int

dnvlist_take_descriptor(*const nvlist_t *nvl*, *const char *name*, *int defval*);

*void **
dnvlist_take_binary(*const nvlist_t *nvl*, *const char *name*, *size_t *sizep*, *void *defval*, *size_t defsize*);

DESCRIPTION

The **libnv** library permits easy management of name/value pairs and can send and receive them over sockets. For more information, also see nv(9).

The **dnvlist_get** family of functions returns the value associated with the specified name. If an element of the specified name does not exist, the function returns the value provided in *defval*. Returned strings, nvlists, descriptors, binaries, or arrays must not be modified by the user. They still belong to the nvlist. If the nvlist is in an error state, attempts to use any of these functions will cause the program to abort.

The **dnvlist_take** family of functions returns the value associated with the specified name and removes the element from the nvlist. If an element of the supplied name does not exist, the value provided in **defval** is returned. When the value is a string, binary, or array value, the caller is responsible for freeing returned memory with **free**(*3*). When the value is an nvlist, the caller is responsible for destroying the returned nvlist with **nvlist_destroy**(). When the value is a descriptor, the caller is responsible for closing the returned descriptor with **close**(*2*).

SEE ALSO

close(2), free(3), nv(9)

HISTORY

The **dnv** API appeared in FreeBSD 11.0.

AUTHORS

The **dnv** API was implemented by Pawel Jakub Dawidek <*pawel@dawidek.net*> under sponsorship from the FreeBSD Foundation. This manual page was written by Adam Starak <*starak.adam@gmail.com*>