NAME

cap_init, **cap_wrap**, **cap_unwrap**, **cap_sock**, **cap_clone**, **cap_close**, **cap_limit_get**, **cap_limit_set**, **cap_send_nvlist**, **cap_recv_nvlist**, **cap_xfer_nvlist**, **cap_service_open** **-** library for handling application capabilities

LIBRARY

Casper Library (libcasper, -lcasper)

SYNOPSIS

#define WITH_CASPER

#include <sys/nv.h>
#include <libcasper.h>

*cap_channel_t **
cap_init(*void*);

*cap_channel_t **
cap_wrap(*int sock*, *int flags*);

int
cap_unwrap(*cap_channel_t *chan*, *int *flags*);

int
cap_sock(*const cap_channel_t *chan*);

*cap_channel_t **
cap_clone(*const cap_channel_t *chan*);

void
cap_close(*cap_channel_t *chan*);

int
cap_limit_get(*const cap_channel_t *chan*, *nvlist_t **limitsp*);

int
cap_limit_set(*const cap_channel_t *chan*, *nvlist_t *limits*);

int
cap_send_nvlist(*const cap_channel_t *chan*, *const nvlist_t *nvl*);

*nvlist_t **
cap_recv_nvlist(*const cap_channel_t *chan*);

*nvlist_t **
cap_xfer_nvlist(*const cap_channel_t *chan*, *nvlist_t *nvl*);

*cap_channel_t **
cap_service_open(*const cap_channel_t *chan*, *const char *name*);

DESCRIPTION

The **libcasper** library provides for the control of application capabilities through the casper process.

An application capability, represented by the *cap_channel_t* type, is a communication channel between the caller and the casper daemon or an instance of one of the daemon's services. A capability to the casper process, obtained with the **cap_init**() function, allows a program to create capabilities to access the casper daemon's services via the **cap_service_open**() function.

The **cap_init**() function instantiates a capability to allow a program to access the casper daemon. It must be called from a single-threaded context.

The **cap_wrap**() function creates a *cap_channel_t* based on the socket supplied in the call. The function is used when a capability is inherited through the execve(2) system call, or sent over a unix(4) domain socket as a file descriptor, and has to be converted into a *cap_channel_t*. The *flags* argument defines the channel behavior. The supported flags are:

CASPER_NO_UNIQ
The communication between the process and the casper daemon uses no unique version of nvlist.

The **cap_unwrap**() function returns the unix(4) domain socket used by the daemon service, and frees the *cap_channel_t* structure.

The **cap_clone**() function returns a clone of the capability passed as its only argument.

The **cap_close**() function closes, and frees, the given capability.

The **cap_sock**() function returns the unix(4) domain socket descriptor associated with the given capability for use with system calls such as: kevent(2), poll(2), and select(2).

The **cap_limit_get**() function stores the current limits of the given capability in the *limitsp* argument. If the function returns *0* and NULL is stored in the *limitsp* argument, there are no limits set.

The **cap_limit_set**() function sets limits for the given capability. The limits are provided as an nvlist(9). The exact format of the limits depends on the service that the capability represents. **cap_limit_set**() frees the limits passed to the call, whether or not the operation succeeds or fails.

The **cap_send_nvlist**() function sends the given nvlist(9) over the given capability. This is a low level interface to communicate with casper services. It is expected that most services will provide a higher level API.

The **cap_recv_nvlist**() function receives the given nvlist(9) over the given capability.

The **cap_xfer_nvlist**() function sends the given nvlist(9), destroys it, and receives a new nvlist(9) in response over the given capability. It does not matter if the function succeeds or fails, the nvlist(9) given for sending will always be destroyed before the function returns.

The **cap_service_open**() function opens the casper service named in the call using the casper capability obtained via the **cap_init**() function. The **cap_service_open**() function returns a capability that provides access to the opened service. Casper supports the following services in the base system:

system.dns	provides libc compatible DNS API
system.fileargs	provides an API for opening files specified on a command line
system.grp	provides a getgrent(3) compatible API
system.net	provides a libc compatible network API
system.netdb	provides libc compatible network proto API
system.pwd	provides a getpwent(3) compatible API
system.sysctl	provides a sysctlbyname(3) compatible API
system.syslog	provides a syslog(3) compatible API

RETURN VALUES

The **cap_clone**(), **cap_init**(), **cap_recv_nvlist**(), **cap_service_open**(), **cap_wrap**() and **cap_xfer_nvlist**() functions return NULL and set the *errno* variable on failure.

The **cap_limit_get**(), **cap_limit_set**() and **cap_send_nvlist**() functions return -1 and set the *errno* variable on failure.

The **cap_close**(), **cap_sock**() and **cap_unwrap**() functions always succeed.

SEE ALSO

errno(2), execve(2), kevent(2), poll(2), select(2), cap_dns(3), cap_fileargs(3), cap_grp(3), cap_net(3), cap_netdb(3), cap_pwd(3), cap_sysctl(3), cap_syslog(3), libcasper_service(3), capsicum(4), unix(4), nv(9)

HISTORY

The **libcasper** library first appeared in FreeBSD 10.3.

AUTHORS

The **libcasper** library was implemented by Pawel Jakub Dawidek <*pawel@dawidek.net*> under sponsorship from the FreeBSD Foundation. The **libcasper** new architecture was implemented by Mariusz Zaborski <*oshogbo@FreeBSD.org*>