NAME

mac_prepare, **mac_prepare_type**, **mac_prepare_file_label**, **mac_prepare_ifnet_label**, **mac_prepare_process_label** - allocate appropriate storage for *mac_t*

SYNOPSIS

#include <sys/mac.h>

int
mac_prepare(*mac_t *mac, const char *elements*);

int
mac_prepare_type(*mac_t *mac, const char *name*);

int
mac_prepare_file_label(*mac_t *mac*);

int
mac_prepare_ifnet_label(*mac_t *mac*);

int
mac_prepare_process_label(*mac_t *mac*);

DESCRIPTION

The **mac_prepare** family of functions allocates the appropriate amount of storage and initializes **mac* for use by mac_get(3). When the resulting label is passed into the mac_get(3) functions, the kernel will attempt to fill in the label elements specified when the label was prepared. Elements are specified in a nul-terminated string, using commas to delimit fields. Element names may be prefixed with the '?' character to indicate that a failure by the kernel to retrieve that element should not be considered fatal.

The **mac_prepare**() function accepts a list of policy names as a parameter, and allocates the storage to fit those label elements accordingly. The remaining functions in the family make use of system defaults defined in mac.conf(5) instead of an explicit *elements* argument, deriving the default from the specified object type.

mac_prepare_type() allocates the storage to fit an object label of the type specified by the *name* argument. The **mac_prepare_file_label**(), **mac_prepare_ifnet_label**(), and **mac_prepare_process_label**() functions are equivalent to invocations of **mac_prepare_type**() with arguments of "file", "ifnet", and "process" respectively.

RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

SEE ALSO

mac(3), mac_free(3), mac_get(3), mac_is_present(3), mac_set(3), mac(4), mac.conf(5), maclabel(7)

STANDARDS

POSIX.1e is described in IEEE POSIX.1e draft 17. Discussion of the draft continues on the cross-platform POSIX.1e implementation mailing list. To join this list, see the FreeBSD POSIX.1e implementation page for more information.

HISTORY

Support for Mandatory Access Control was introduced in FreeBSD 5.0 as part of the TrustedBSD Project. Support for generic object types first appeared in FreeBSD 5.2.