NAME

MAC_CAPAB_LED, **mac_capab_led**, **mac_capab_led_t**, **mcl_set -** MAC LED capability

SYNOPSIS

#include <sys/mac_provider.h>

typedef struct mac_capab_led mac_capab_led_t;
typedef enum mac_led_mode mac_led_mode_t;

int
mcl_set(*void *driver*, *mac_led_mode_t mode*, *uint_t flags*);

INTERFACE LEVEL

Evolving - This interface is evolving still in illumos. API and ABI stability is not guaranteed.

PARAMETERS

driver	A pointer to the driver's private data that was passed in via the **m_pdata** member of the mac_register(9S) structure to the mac_register(9F) function.
mode	A value that indicates how the driver should drive the LEDs.
flags	Reserved for future use. *LED Modes*.

DESCRIPTION

The **MAC_CAPAB_LED** provides a means for GLDv3 device drivers to change the behavior of the LEDs, allowing an administrator to change the behavior for the purpose of better finding and identifying a physical device.

Implementing this capability is optional. For more information on how to handle capabilities and how to indicate that a capability is not supported, see mc_getcapab(9E).

This capability should be implemented if the device in question provides a way to manipulate its LEDs. Generally the LEDs on a device default to indicating link status and activity. However, they can often be turned off or set to a specific pattern to indicate identification purposes.

LED MODES

The system has a notion of different LED modes. Each LED mode suggests a different way that a device driver should drive the indicator LEDs on the device. While we generally want all such LED modes to be as uniform as possible, there is a limit to such similarities due to the capabilities of NICs. Each mode is a member of the *mac_led_mode_t* enumeration. The currently defined modes are:

MAC_LED_DEFAULT

This mode indicates that the device's default behavior should be used. This is usually some form of link status and activity. It is device specific and usually is the default behavior after a device is powered on.

MAC_LED_OFF

This mode indicates that the device's LEDs should be turned off and not emit any light.

MAC_LED_IDENT

This mode indicates that the driver should emit some form of identication pattern. We suggest that devices indicate some form of solid blinking light that is on and off at alternating units of time. If it is possible to use an alternate color from the normal link up and activity lighting, that is recommended.

MAC Capability Structure

When the device driver's mc_getcapab(9E) function entry point is called with the capability set to MAC_CAPAB_LED, then the value of the capability structure is the following structure:

```
typedef struct mac_capab_led {
        uint_t              mcl_flags;
        mac_led_mode_t    mcl_modes;
        int           (*mcl_set)(void *driver, mac_led_mode_t mode,
                          uint_t flags);
} mac_capab_led_t;
```

If the driver supports the MAC_CAPAB_LED capability, it should fill in this structure, based on the following rules:

mcl_flags

The *mcl_flags* member is used to negotiate extensions with the driver. MAC will set the value of *mcl_flags* to include all of the currently known extensions. The driver should intersect this list with the set that they actually support. At this time, no such features are defined and the driver should set the member to **0**.

mcl_modes

The *mcl_modes* member represents the support modes of the device driver. The device driver should set *mcl_modes* to the bitwise-inclusive-OR of the LED modes listed in *LED MODES*.

If the driver does not support anything other than the default behavior of

MAC_LED_DEFAULT, then the device driver should not indicate that it supports this capability.

mcl_set

The *mct_set* entry point will be called by the MAC framework when it needs the device driver to change how it is driving its LEDs. Each call will ask the driver to change the display mode to the specified mode. The driver does not have to multiplex requests for multiple modes or keep track of what has been requested, that is taken care of by the system itself.

The driver should first validate that *mode* is a mode that it supports. While the device reports the set of supported modes as a bitwise-inclusive-OR, the driver should only receive a single value in *mode*. The value of the *flags* argument is reserved for future use. Drivers must check that the value of flags is zero and if not, return EINVAL.

When this entry point is first called on a driver, it should snapshot its device registers such that it knows how to restore the default behavior. Because each method of programming the LEDs is different, it is up to the driver itself to take care of this, the broader framework cannot take care of it.

If for some reason the driver is asked to program the same mode as it is already driving, then it need not do anything and should simply return success.

Once the driver successfully changes the LED driving mode, it should return **0**. Otherwise, it should return the appropriate error number. For a full list of error numbers, see Intro(2). Common values are:

EINVAL	*mode* is unsupported. *flags* contains an unsupported or unknown value.
EIO	An I/O error occurred while trying to program the device's registers. This could be because a command timed out or an FM-aware driver encountered an error.

CONTEXT

The *mcl_set* entry point will only be called from **user** or **kernel** context. It will never be called from interrupt context.

SEE ALSO