NAME

ddi_ufm, **ddi_ufm_init**, **ddi_ufm_is_ioctl**, **ddi_ufm_ioctl**, **ddi_ufm_update**, **ddi_ufm_fini** - DDI upgradable firmware module interfaces

SYNOPSIS

#include <sys/ddi_ufm.h>

int
ddi_ufm_init(*dev_info_t *dip, int version, ddi_ufm_ops_t *ops, ddi_ufm_handle_t **ufmpp, void *drv_arg*);

boolean_t
ddi_ufm_is_ioctl(*ddi_ufm_handle_t *ufmp, int cmd*);

int
ddi_ufm_ioctl(*ddi_ufm_handle_t *ufmp, dev_t dev, int cmd, intptr_t arg, int mode, cred_t *credp, int *rvalp*);

void
ddi_ufm_update(*ddi_ufm_handle_t *ufmp*);

void
ddi_ufm_fini(*ddi_ufm_handle_t *ufmp*);

INTERFACE LEVEL

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

PARAMETERS

dip Pointer to the devices *dev_info* structure for the specific instance.

version A value which indicates the current revision of the interface that the device supports. Should generally be set to DDI_UFM_CURRENT_VERSION.

ops A pointer to a UFM operations structure. See ddi_ufm(9E) for more information.

ufmp A pinter to the opaque handle returned from **ddi_ufm_init**().

ufmpp A pointer to store the opaque handle from **ddi_ufm_init**().

drv_arg A driver specific argument that will be passed to various operations.

dev Device number.

cmd Command argument that indicates the operation to perform. See ioctl(9E) for more information.

arg Argument used to pass data between a user program and the kernel. See ioctl(9E) for more information.

mode A bit field that describes the caller. See ioctl(9E) for more information.

credp A pointer to the user credential structure. See ioctl(9E) for more information.

rvalp A pointer to a return value. See ioctl(9E) for more information.

DESCRIPTION

These functions provide support for initializing and performing various upgradeable firmware module (UFM) operations. For more information, please see ddi_ufm(9E).

The **ddi_ufm_init**() function is used to initialize support for the UFM subsystem for a given device. The *dip* argument should be the *dev_info* structure of the specific device. The *version* argument represents the current revision of the UFM interface that the driver supports. Drivers inside of illumos should always use DDI_UFM_CURRENT_VERSION. Device drivers which need to bind to a specific revision, should instead pass the latest version: DDI_UFM_VERSION_ONE. The operations structure, *ops*, should be filled according to the rules in ddi_ufm(9E). These will be the entry points that device drivers call. The value of *drv_arg* will be passed to all of the driver's entry points. When the function returns, *ufmpp* will be filled in with a handle that the driver should reference when needing to perform subsequent UFM operations. No UFM entry points will be called until after the driver calls the **ddi_ufm_update**() function.

When the device driver is detaching or needs to unregistger from the UFM subsystem, then the device driver should call the **ddi_ufm_fini**() function with the handle that they obtained during the call to initialize. Note, this function will block and ensure that any outstanding UFM operations are terminated. The driver must not hold any locks that are required in its callbacks across the call to **ddi_ufm_fini**().

The **ddi_ufm_is_ioctl**() function is used during a device driver's ioctl(9E) entry point to determine if the ioctl command *cmd* is a known UFM command. The device driver should not hold any locks that may be used by its UFM entry points while calling this function.

The **ddi_ufm_ioctl**() function should be called by a device driver when the **ddi_ufm_ioctl**() function returns successfully. It will be used to handle a UFM ioctl on behalf of the device driver. The device

driver should simply return the value from this and not perform any other handling of an ioctl. When the device driver calls this, it should ensure that it holds no locks that might be used during the various UFM entry points.

The **ddi_ufm_update**() function should be used in two different circumstances. It should be used at the end of a driver's attach(9E) endpoint to indicate that it is ready to receive UFM requests. It should also be called whenever the driver believes that the UFM might have changed. This may happen after a device reset or firmware change. Unlike the other functions, this can be called from any context with any locks held, excepting high-level interrupt context which normal device drivers will not have interrupts for.

RETURN VALUES

Upon successful completion, the **ddi_ufm_init**() function returns zero, indicating that it has successfully registered with the UFM subsystem. *ufmpp* will be filled in with a pointer to the UFM handle.

The **ddi_ufm_is_ioctl**() function will return B_TRUE if the specified command is a UFM ioctl, and B_FALSE otherwise.

The **ddi_ufm_ioctl**() function's return value should not be inspected or interpreted by the device driver. Instead, it should be returned umodified to the ioctl(9E) function.

CONTEXT

The **ddi_ufm_init**() and **ddi_ufm_fini**() functions are generally called from a device's attach(9E) and fini(9E) routines, though they may be called from **user** or **kernel** context.

The **ddi_ufm_is_ioctl**() and **ddi_ufm_ioctl**() functions should be called from user context from the same thread that received the ioctl(9E) entry point.

The **ddi_ufm_update**() function may be called from any context except a high-level interrupt handler above lock level.

SEE ALSO

attach(9E), ddi_ufm(9E), fini(9E), ioctl(9E)