NAME

ddi_mapdev_intercept, ddi_mapdev_nointercept – control driver notification of user accesses

SYNOPSIS

#include <sys/sunddi.h>

int ddi_mapdev_intercept(ddi_mapdev_handle_t *handle*, **off_t** *offset*, **off_t** *len***);**

int ddi_mapdev_nointercept(ddi_mapdev_handle_t *handle*, **off_t** *offset*, **off_t** *len***);**

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

ARGUMENTS

handle An opaque pointer to a device mapping handle.

offset An offset in bytes within device memory.

len Length in bytes.

DESCRIPTION

ddi_mapdev_intercept( ) and **ddi_mapdev_nointercept( )** control whether or not user accesses to device mappings created by **ddi_mapdev**(9F) in the specified range will generate calls to the **mapdev_access**(9E) entry point. **ddi_mapdev_intercept( )** tells the system to intercept the user access and notify the driver to invalidate the mapping translations. **ddi_mapdev_nointercept( )** tells the system to not intercept the user access and allow it to proceed by validating the mapping translations.

For both routines, the range to be affected is defined by the *offset* and *len* arguments. Requests affect the entire page containing the *offset* and all pages up to and including the page containing the last byte as indicated by *offset* + *len*.

Supplying a value of **0** for the *len* argument affects all addresses from the *offset* to the end of the mapping. Supplying a value of **0** for the *offset* argument and a value of **0** for *len* argument affect all addresses in the mapping.

To manage a device context, a device driver would call **ddi_mapdev_intercept( )** on the context about to be switched out, switch contexts, and then call **ddi_mapdev_nointercept( )** on the context switched in.

RETURN VALUES

ddi_mapdev_intercept( ) and **ddi_mapdev_nointercept( )** return zero on success and non-zero on failure.

EXAMPLE

The following shows an example of managing a device context that is one page in length.

```
ddi_mapdev_handle_t cur_hdl;

static int
xxmapdev_access(ddi_mapdev_handle_t handle, void *devprivate,
   off_t offset)
{
        int err;

        /* enable access callbacks for the current mapping */
        if (cur_hdl != NULL) {
                if ((err = ddi_mapdev_intercept(cur_hdl, offset, 0)) != 0)
```

```
                return (err);
        }
        /* Switch device context - device dependent*/
        ...
        /* Make handle the new current mapping */
        cur_hdl = handle;
        /*
         * Disable callbacks and complete the access for the
         * mapping that generated this callback.
         */
        return (ddi_mapdev_nointercept(handle, offset, 0));
}
```

CONTEXT

These routines can be called from user or kernel context only.

SEE ALSO

mapdev_access(9E), **ddi_mapdev**(9F)

Writing Device Drivers