NAME

ddi_prop_lookup, ddi_prop_lookup_int_array, ddi_prop_lookup_string_array, ddi_prop_lookup_string, ddi_prop_lookup_byte_array, ddi_prop_free – look up property information

SYNOPSIS

```
#include <sys/ddi.h>
#include <sys/sunddi.h>

int ddi_prop_lookup_int_array(dev_t match_dev, dev_info_t *dip,
    u_int flags, char *name, int **datap, u_int *nelementsp);

int ddi_prop_lookup_string_array(dev_t match_dev, dev_info_t *dip,
    u_int flags, char *name, char ***datap, u_int *nelementsp);

int ddi_prop_lookup_string(dev_t match_dev, dev_info_t *dip,
    u_int flags, char *name, char **datap);

int ddi_prop_lookup_byte_array(dev_t match_dev, dev_info_t *dip,
    u_int flags, char *name, u_char **datap, u_int *nelementsp);

void ddi_prop_free(void *data);
```

ARGUMENTS

match_dev Device number associated with property or **DDI_DEV_T_ANY.**

dip Pointer to the device info node of device whose property list should be searched.

flags Possible flag values are some combination of:

DDI_PROP_DONTPASS
Do not pass request to parent device information node if the property is not found.

DDI_PROP_NOTPROM
Do not look at PROM properties (ignored on platforms that do not support PROM properties).

name String containing the name of the property.

nelementsp The address of an unsigned integer which, upon successful return, will contain the number of elements accounted for in the memory pointed at by *datap*. The elements are either integers, strings or bytes depending on the interface used.

ddi_prop_lookup_int_array()

datap The address of a pointer to an array of integers which, upon successful return, will point to memory containing the integer array property value.

ddi_prop_lookup_string_array()

datap The address of a pointer to an array of strings which, upon successful return, will point to memory containing the array of strings. The array of strings is formatted as an array of pointers to NULL terminated strings, much like the *argv* argument to **execve**(2).

ddi_prop_lookup_string( )

datap — The address of a pointer to a string which, upon successful return, will point to memory containing the NULL terminated string value of the property.

ddi_prop_lookup_byte_array( )

datap — The address of pointer to an array of bytes which, upon successful return, will point to memory containing the byte array value of the property.

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

DESCRIPTION

The property look up routines search for and, if found, return the value of a given property. Properties are searched for based on the *dip*, *name*, *match_dev*, and the type of the data (integer, string or byte). The property search order is as follows:

1. Search software properties created by the driver.
2. Search the software properties created by the system (or nexus nodes in the device info tree).
3. Search the driver global properties list.
4. If **DDI_PROP_NOTPROM** is not set, search the PROM properties (if they exist).
5. If **DDI_PROP_DONTPASS** is not set, pass this request to the parent device information node.
6. Return **DDI_PROP_NOT_FOUND.**

Usually, the *match_dev* argument should be set to the actual device number that this property is associated with. However, if the *match_dev* argument is **DDI_DEV_T_ANY,** the property look up routines will match the request regardless of the actual *match_dev* the property was created with. If a property was created with *match_dev* set to **DDI_DEV_T_NONE** then the only way to look up this property is with a *match_dev* set to **DDI_DEV_T_ANY.** PROM properties are always created with *match_dev* set to **DDI_DEV_T_NONE.**

name must always be set to the name of the property being looked up.

For the routines **ddi_prop_lookup_int_array**( ), **ddi_prop_lookup_string_array**( ), **ddi_prop_lookup_string**( ), and **ddi_prop_lookup_byte_array**( ), *datap* is the address of a pointer which, upon successful return, will point to memory containing the value of the property. In each case **datap* points to a different type of property value. See the individual descriptions of the routines below for details on the different return values. *nelementsp* is the address of an unsigned integer which, upon successful return, will contain the number of integer, string or byte elements accounted for in the memory pointed at by **datap*.

All of the property look up routines may block to allocate memory needed to hold the value of the property.

When a driver has obtained a property with any look up routine and is finished with that property, it must be freed by calling **ddi_prop_free**( ). **ddi_prop_free**( ) must be called with the address of the allocated property. For instance if one called

ddi_prop_lookup_int_array( ) with *datap* set to the address of a pointer to an integer, *&my_int_ptr*, then the companion free call would be **ddi_prop_free(***my_int_ptr***).**

ddi_prop_lookup_int_array( )

This routine searches for and returns an array of integer property values. An array of integers is defined to **nelementsp* number of 4 byte long integer elements. *datap* should be set to the address of a pointer to an array of integers which, upon successful return, will point to memory containing the integer array value of the property.

ddi_prop_lookup_string_array( )

This routine searches for and returns a property that is an array of strings. *datap* should be set to address of a pointer to an array of strings which, upon successful return, will point to memory containing the array of strings. The array of strings is formatted as an array of pointers to NULL terminated strings, much like the *argv* argument to **execve**(2).

ddi_prop_lookup_string( )

This routine searches for and returns a property that is a NULL terminated string. *datap* should be set to the address of a pointer to string which, upon successful return, will point to memory containing the string value of the property.

ddi_prop_lookup_byte_array( )

This routine searches for and returns a property that is an array of bytes. *datap* should be set to the address of a pointer to an array of bytes which, upon successful return, will point to memory containing the byte array value of the property.

ddi_prop_free( )

Frees the resources associated with a property previously allocated using **ddi_prop_lookup_int_array**( ), **ddi_prop_lookup_string_array**( ), **ddi_prop_lookup_string**( ), or **ddi_prop_lookup_byte_array**( ).

RETURN VALUES

ddi_prop_lookup_int_array( )
ddi_prop_lookup_string_array( )
ddi_prop_lookup_string( )
ddi_prop_lookup_byte_array( )

DDI_PROP_SUCCESS	on success.
DDI_PROP_INVAL_ARG	if an attempt is made to look up a property with *match_dev* equal to **DDI_DEV_T_NONE**, *name* is NULL or *name* is the null string.
DDI_PROP_NOT_FOUND	Property not found.
DDI_PROP_UNDEFINED	Property explicitly not defined (see **ddi_prop_undefine**(9F)).
DDI_PROP_CANNOT_DECODE	The value of the property cannot be decoded.

CONTEXT

These functions can be called from user or kernel context.

EXAMPLES

The following example demonstrates the use of **ddi_prop_lookup( )**.

```
int    *options;
int    noptions;

/*
```

```
 * Get the data associated with the integer "options" property
 * array, along with the number of option integers
 */
if (ddi_prop_lookup_int_array(DDI_DEV_T_ANY, xx_dip, 0,
        "options", &options, &noptions) == DDI_PROP_SUCCESS) {
        /*
         * Do "our thing" with the options data from the property
         */
        xx_process_options(options, noptions);

        /*
         * Free the memory allocated for the property data
         */
        ddi_prop_free(options);
}
```

SEE ALSO **ddi_prop_exists**(9F), **ddi_prop_get_int**(9F), **ddi_prop_update**(9F), **ddi_prop_remove**(9F), **ddi_prop_undefine**(9F)

Writing Device Drivers