NAME

efi_variables_supported, efi_del_variable, efi_get_variable, efi_get_variable_attributes, efi_get_variable_size, efi_set_variable − manipulate UEFI variables

SYNOPSIS

#include <efivar.h>

int efi_variables_supported(void);
int efi_del_variable(efi_guid_t *guid***, const char** **name***);**

int efi_get_variable(efi_guid_t *guid***, const char ***_name_**,**
void ****_data_, ssize_t ***_data_size_**,**
uint32_t ***_attributes_);**

int efi_get_variable_attributes(efi_guid_t *guid***, const char ***_name_**,**
uint32_t ***_attributes_);**

int efi_get_variable_exists(efi_guid_t *guid***, const char** **name***);**

int efi_get_variable_size(efi_guid_t *guid***, const char ***_name_**,**
size_t ***_size_);**

int efi_append_variable(efi_guid_t *guid***, const char ***_name_**,**
void ***_data_, size_t** *data_size***,**
uint32_t *attributes***);**

int efi_set_variable(efi_guid_t *guid***, const char ***_name_**,**
void ***_data_, size_t** *data_size***,**
uint32_t *attributes***, mode_t** *mode***);**

int efi_get_next_variable_name(efi_guid_t ****_guid_, char ****_name_**);**

int efi_str_to_guid(const char ***_s_, efi_guid_t ***_guid_**);**

int efi_guid_to_str(const efi_guid_t ***_guid_, char ****_sp_**);**

int efi_name_to_guid(const char ***_name_, efi_guid_t ***_guid_**);**

int efi_id_guid_to_guid(const char ***_id_guid_, efi_guid_t ***_guid_**);**

int efi_guid_to_name(efi_guid_t ***_guid_, char ****_name_**);**

int efi_guid_to_id_guid(efi_guid_t ***_guid_, char ****_id_guid_**);**

int efi_guid_to_symbol(efi_guid_t ***_guid_, char ****_symbol_**);**
int efi_symbol_to_guid(const char ***_symbol_, efi_guid_t ***_guid_**);**

DESCRIPTION

efi_variables_supported() tests if the UEFI variable facility is supported on the current machine.

efi_del_variable() deletes the variable specified by *guid* and *name*.

efi_get_variable() gets the variable specified by *guid* and *name*. The value is stored in *data*, its size in *data_size*, and its attributes are stored in *attributes*.

efi_get_variable_attributes() gets attributes for the variable specified by *guid* and *name*.

efi_get_variable_exists() gets if the variable specified by *guid* and *name* exists.

efi_get_variable_size() gets the size of the data for the variable specified by *guid* and *name*.

efi_append_variable() appends *data* of size *size* to the variable specified by *guid* and *name*.

efi_set_variable() sets the variable specified by *guid* and *name*, and sets the file mode to *mode*, subject to umask. Note that the mode will not persist across a reboot, and that the permissions only apply if on systems using efivarfs.

efi_get_next_variable_name() iterates across the currently extant variables, passing back a guid and name.

efi_str_to_guid() parses a UEFI GUID from string form to an efi_guid_t the caller provides

efi_guid_to_str() Creates a string representation of a UEFI GUID. If sp is NULL, it returns how big the string would be. If sp is not NULL but *sp is NULL, it allocates a string and returns it with. It is the caller's responsibility to free this string. If sp is not NULL and *sp is not NULL, **efi_guid_to_str**() assumes there is an allocation of suitable size and uses it.

efi_name_to_guid() translates from a well known name to an efi_guid_t the caller provides.

efi_guid_to_name() translates from an efi_guid_t to a well known name. If the supplied GUID does not have a well known name, this function is equivalent to **efi_guid_to_str**().

efi_guid_to_id_guid() translates from an efi_guid_t to an {ID GUID}. If the supplied GUID has a well known name, the {ID GUID} will be of the form "{name_here}". If not, it will be of the form "{66b2af1c-6211-4082-95cb-9f73a4795a7e}".

efi_id_guid_to_guid() translates from an {ID GUID} to an efi_guid_t the caller provides.

efi_guid_to_symbol() translates from an efi_guid_t to a unique (within libefivar) C-style symbol name. These symbol names are useful for printing as a unique, easily parsed identifier, and are also provide by the library and its header files.

efi_symbol_to_guid() translates from a libefivar efi_guid_$FOO symbol name to an efi_guid_t the caller provides.

RETURN VALUE

efi_variables_supported() returns true if variables are supported on the running hardware, and false if they are not.

efi_get_next_variable_name() returns 0 when iteration has completed, 1 when iteration has not completed, and -1 on error. In the event of an error, *errno*(3) is set appropriately.

efi_del_variable(), **efi_get_variable**(), **efi_get_variable_attributes**(), **efi_get_variable_exists**(), **efi_get_variable_size**(), **efi_append_variable**(), **efi_set_variable**(), **efi_str_to_guid**(), **efi_guid_to_str**(), **efi_name_to_guid**(), and **efi_guid_to_name**() return negative on error and zero on success.

AUTHORS

Peter Jones <pjones@redhat.com>