NAME

efi_variable_import, efi_variable_export, efi_variable_alloc, efi_variable_free, efi_variable_set_name, efi_variable_get_name, efi_variable_set_guid, efi_variable_get_guid, efi_variable_set_data, efi_variable_get_data, efi_variable_set_attributes, efi_variable_get_attributes, efi_variable_realize − utility functions to import and export UEFI variables to files.

SYNOPSIS

#include <efivar.h>

typedef struct efi_variable **efi_variable_t***;*

ssize_t **efi_variable_import**(*uint8_t* ***data**, *size_t* **size**, *efi_variable_t* ****var**);
ssize_t **efi_variable_export**(*efi_variable_t* ***var**, *uint8_t* ****data**, *size_t* ***size**);

efi_variable_t ***efi_variable_alloc**(*void*);
void **efi_variable_free**(*efi_variable_t* ***var**, *int* **free_data**);

int **efi_variable_set_name**(*efi_variable_t* ***var**, *char* ***name**);
char ***efi_variable_get_name**(*efi_variable_t* ***var**);

int **efi_variable_set_guid**(*efi_variable_t* ***var**, *efi_guid_t* ***guid**);
int **efi_variable_get_guid**(*efi_variable_t* ***var**, *efi_guid_t* ****guid**);

int **efi_variable_set_data**(*efi_variable_t* ***var**, *uint8_t* ***data**, *size_t* **size**);
int **efi_variable_get_data**(*efi_variable_t* ***var**, *uint8_t* ****data**, *size_t* ***size**);

#define **EFI_VARIABLE_NON_VOLATILE** *0x0000000000000001*
#define **EFI_VARIABLE_BOOTSERVICE_ACCESS** *0x0000000000000002*
#define **EFI_VARIABLE_RUNTIME_ACCESS** *0x0000000000000004*
#define **EFI_VARIABLE_HARDWARE_ERROR_RECORD** *0x0000000000000008*
#define **EFI_VARIABLE_AUTHENTICATED_WRITE_ACCESS** *0x0000000000000010*
#define **EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS** *0x0000000000000020*
#define **EFI_VARIABLE_APPEND_WRITE** *0x0000000000000040*
#define **EFI_VARIABLE_HAS_AUTH_HEADER** *0x0000000100000000*
#define **EFI_VARIABLE_HAS_SIGNATURE** *0x0000000200000000*

int **efi_variable_set_attributes**(*efi_variable_t* ***var**, *uint64_t* **attrs**);
int **efi_variable_get_attributes**(*efi_variable_t* ***var**, *uint64_t* ***attrs**);

int **efi_variable_realize**(*efi_variable_t* ***var**);

DESCRIPTION

efi_variable_t is an opaque data type used to store variables in-memory for use with this API.

efi_variable_import() is used to import raw data read from a file. This function returns the amount of data consumed with this variable, and may be used successively, using its return code as an offset, to parse a list of variables. Note that the internal guid, name, and data values are allocated separately, and must be freed either individually or using the **free_data** parameter of **efi_variable_free**(). **_get**() accessors for those values return data suitable for freeing individually, except in such cases where a **_set**() accessor has been passed an object already unsuitable for that.

efi_variable_export() is used to marshall **efi_variable_t** objects into linear data which can be written to a file. If**data** or **size** parameters are not pro vided, this function will return how much storage a caller must allocate. Otherwise,**efi_v ariable_export**() will use the storage referred to as its buffer; if **size** is smaller than the amount of needed storage , the buffer will not be modified, and the difference between the needed space and **size** will be returned.

efi_variable_alloc() is used to allocate an unpopulated **efi_variable_t** object suitable to be used throughout this API. **efi_v ariable_free**() is used to free an **efi_variable_t** object, and if **free_data** is nonzero, to free its constituent data.

Each pair of **_set**() and **_get**() accessors have essentially the same semantics. Neither operation performs any memory management, including freeing of previously set values or values set by **efi_variable_import**(), and so in some cases it may be necessary to use a **_get**() accessor to retrieve an object to be freed. In cases where no value has been set, **_get**() accessors will set **errno** to **ENOENT** and return a negative value or NULL.

efi_variable_set_name() and **efi_variable_get_name**() are used to set and retrieve the name of the variable referred to by the **efi_variable_t** object.

efi_variable_set_guid() and **efi_variable_get_guid**() are used to set and retrieve the Vendor GUID value of the variable referred to by the **efi_variable_t** object.

efi_variable_set_data() and **efi_variable_get_data**() are used to set and retrieve an **efi_variable_t** object's variable data.

efi_variable_set_attributes() and **efi_variable_get_attributes** are used to set and retrieve an **efi_variable_t** object's attributes. All bits except **EFI_VARIABLE_HAS_AUTH_HEADER** and **EFI_VARIABLE_HAS_SIGNATURE** are defined in the UEFI specification and should be used accordingly. **EFI_VARIABLE_HAS_AUTH_HEADER** should be used by applications to track whether the variable data contents include an authentication header. **EFI_VARIABLE_HAS_SIGNATURE** should be used by applications to track if the variable's data contents include a signature, and should not be set unless **EFI_VARIABLE_HAS_AUTH_HEADER** is also set. These attributes are used to track if an exported variable is in a state of partial construction, for example if an authenticated variable has been created but is intended to be signed at a later date.

efi_variable_realize() is a convenience function to set or append a UEFI variable on the running system from an **efi_variable_t** object. its return codes are the same as **efi_append_variable**(3) if EFI_VARIABLE_APPEND_WRITE is set, and **efi_set_variable**() if that bit is not set. Additionally, in the case that any of the authentication bits are set, **efi_variable_realize**() will return error and set **errno** to **EPERM** unless both **EFI_VARIABLE_HAS_AUTH_HEADER** and **EFI_VARIABLE_HAS_SIGNATURE** attribute bits are been set.

RETURN VALUE

efi_variable_import() returns 0 on success, and -1 on failure. In cases where it cannot parse the data,**errno** will be set to **EINVAL**. In cases where memory has been exhausted, **errno** will be set to **ENOMEM**.

efi_variable_export() returns the size of the buffer data on success, or a negative value in the case of an error. If**data** or **size** parameters are not pro vided, this function will return how much storage a caller must allocate. Otherwise, this function will use the storage provided in **data**; if **size** is less than the needed space, the buffer will not be modified, and the return value will be the difficiency in size.

efi_variable_alloc() returns a newly allocated **efi_variable_t** object, but does not peform any allocation for that object's **name**, **guid**, or **data**. In the case that memory is exhausted, **NULL** will be returned, and **errno** will be set to **ENOMEM**.

efi_variable_get_name() returns a pointer the NUL-terminated string containing the **efi_variable_t** object's name information.

efi_variable_set_name(), **efi_variable_set_guid**(), **efi_variable_get_guid**(), **efi_variable_set_data**(), **efi_variable_get_data**(), **efi_variable_set_attributes**(), **efi_variable_get_attributes**(), and **efi_variable_realize**() return 0 on success and -1 on error.

AUTHORS

Peter Jones <pjones@redhat.com>