UF_ATTR_count_user_attribute_titles (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_count_user_attribute_titles

Count the number of attributes that satisfy the given iterator

NOTE: This function counts array attributes as one title.
(in other words, an array with 'n' elements adds only one to the returned count.

Environment

Internal and External

History

NX10.0.0

Required License(s)

gateway

```
int UF_ATTR_count_user_attribute_titles
(
    tag_t object,
    const UF_ATTR_iterator_t * iter,
    int * count
)
```

tag_t	**object**	Input	The object holding the attributes
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attributes being counted. The iterator is automatically reset.
int *	**count**	Output	The count of attributes (and/or unset templates) found

---

UF_ATTR_count_user_attributes (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_count_user_attributes

Count the number of attributes that satisfy the given iterator

NOTE: This function counts individual array elements of arrays
(in other words, an array with 'n' elements adds 'n' to the returned count.

Replaces: UF_ATTR_count_attributes

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

int UF_ATTR_count_user_attributes
(
tag_t object,
const UF_ATTR_iterator_t * iter,
int * count
)

tag_t	**object**	Input	The object holding the attributes
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attributes being counted. The iterator is automatically reset.
int *	**count**	Output	The count of attributes (and/or unset templates) found

---

UF_ATTR_delete_user_attribute_with_title_and_type (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_delete_user_attribute_with_title_and_type

Delete the attribute with the given title and type

Replaces: UF_ATTR_delete

Replaces: UF_ATTR_set_locked

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

int UF_ATTR_delete_user_attribute_with_title_and_type
(
tag_t object,
const char * title,
int type,
int index,
logical update
)

tag_t	**object**	Input	The object holding the attributes
const char *	**title**	Input	The title of the attribute
int	**type**	Input	The attribute type Valid values:

			UF_ATTR_integer UF_ATTR_real UF_ATTR_time UF_ATTR_null UF_ATTR_string UF_ATTR_bool UF_ATTR_any NOTE: If UF_ATTR_any is used, the first attribute encountered that matches the given title (and index), is deleted
int	**index**	Input	The array index Set to UF_ATTR_NOT_ARRAY if the attribute is not an array. Set to UF_ATTR_LAST_ELEMENT if the attribute is an array.
logical	**update**	Input	Perform an update immediately

UF_ATTR_delete_user_attributes (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_delete_user_attributes

Delete all the attributes that satisfy the given iterator.

Replaces: UF_ATTR_delete_all

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_delete_user_attributes
(
    tag_t object,
    const UF_ATTR_iterator_t * iter,
    logical update
)
```

tag_t	**object**	Input	The object holding the attributes
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attributes to be deleted. The iterator is automatically reset.
logical	**update**	Input	Perform an update immediately

UF_ATTR_free_user_attribute_info_array (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_free_user_attribute_info_array

Frees the strings held by an array of UF_ATTR_info_t structs as well as the array itself.
(Use after call to UF_ATTR_get_user_attributes)

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_free_user_attribute_info_array
(
    int count,
    UF_ATTR_info_t info [ ]
)
```

int	**count**	Input	the count of attribute information structs in the array
UF_ATTR_info_t	**info [ ]**	Input	the array of attribute information structs to be freed

---

UF_ATTR_free_user_attribute_info_strings (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_free_user_attribute_info_strings

Frees the strings held by a UF_ATTR_info_t struct.

All strings held by the struct are freed.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_free_user_attribute_info_strings
(
    UF_ATTR_info_p_t info
)
```

UF_ATTR_info_p_t	**info**	Input / Output	the attribute information containing strings to be freed

UF_ATTR_free_user_attribute_iterator_strings (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_free_user_attribute_iterator_strings

Frees the strings held by a UF_ATTR_iterator_t struct.

All strings held by the struct are freed.
Note: The UF_ATTR API does not supply the strings held by this struct, this function for convenience only.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_free_user_attribute_iterator_strings
(
    UF_ATTR_iterator_p_t iterator
)
```

UF_ATTR_iterator_p_t	**iterator**	Input / Output	the attribute iterator containing strings to be freed

UF_ATTR_get_bool_user_attribute (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_get_bool_user_attribute

Gets value of a boolean type attribute, if it exists

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_get_bool_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    logical * value,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
logical *	**value**	Output	The value, if any
logical *	**has_attribute**	Output	An attribute has been found

---

UF_ATTR_get_computational_time_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_computational_time_user_attribute

Gets the value of a time type attribute, if it exists
The returned time is in the local time of the program that is running.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_get_computational_time_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    int value [ 2 ] ,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
int	**value [2]**	Output	Time/Date value as computational time
logical *	**has_attribute**	Output	An attribute has been found

UF_ATTR_get_integer_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_integer_user_attribute

Gets the value of a integer type attribute, if it exists

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_get_integer_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    int * value,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
int *	**value**	Output	The value, if any
logical *	**has_attribute**	Output	An attribute has been found

UF_ATTR_get_next_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_next_user_attribute

Gets the information from the next attribute that satisfies the given iterator

NOTE: If the iteration is allowed to completed, the iterator is automatically reset.
If not, it must be reset with 'UF_ATTR_release_user_attribute_iterator()', or the iterator will leak memory.
NOTE: The supplied 'info' struct must be initialized before use.
Between calls to this function, the 'info' struct's strings do not have to be freed, as this is done prior

to reading an attribute.
However after the last call, the returned 'info' struct's strings must be freed.
This can be done with UF_ATTR_free_user_attribute_info_strings().

Replaces: UF_ATTR_cycle

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

int UF_ATTR_get_next_user_attribute
(
tag_t object,
UF_ATTR_iterator_t * iter,
UF_ATTR_info_t * info,
logical * has_attribute
)

tag_t	**object**	Input	The object holding the attribute
UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attribute being queried. If iteration is allowed to complete, iterator is reset by the system. If not, reset with UF_ATTR_release_user_attribute_iterator() after use.
UF_ATTR_info_t *	**info**	Output to UF_*free*	The attribute information for the first attribute (or unset template) found, if any. After iteration ends, free embedded strings using UF_ATTR_free_user_attribute_info_strings().
logical *	**has_attribute**	Output	An attribute (or unset template) has been found. If 'false', this ends the iteration.

UF_ATTR_get_null_user_attribute (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_get_null_user_attribute

Queries the presence of a null type attribute

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_get_null_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
logical *	**has_attribute**	Output	An attribute has been found

---

UF_ATTR_get_real_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_real_user_attribute

Gets the value and units of a real type attribute, if it exists

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_get_real_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    double * value,
    tag_t * unit_type,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)

double *	**value**	Output	The value, if any. If attribute has units, this value is in the returned units.
tag_t *	**unit_type**	Output	The unit type tag (NULL_TAG may be returned if unit-less)
logical *	**has_attribute**	Output	An attribute has been found

---

UF_ATTR_get_reference_string_of_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_reference_string_of_user_attribute

Gets the reference string (if any) of a string type attribute, if it exists and it has a reference string

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_get_reference_string_of_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    char * * reference_string,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
char * *	**reference_string**	Output to UF_*free*	Reference string (Free using UF_free() after use)
logical *	**has_attribute**	Output	An attribute has been found AND it has a reference string

---

UF_ATTR_get_string_time_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_string_time_user_attribute

Gets the value of a time type attribute, if it exists
The returned time is in the local time of the program that is running.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_get_string_time_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    char * * time_string,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
char * *	**time_string**	Output to UF_*free*	Time/Date value formatted as a string (Free using UF_free() after use)
logical *	**has_attribute**	Output	An attribute has been found

---

UF_ATTR_get_string_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_string_user_attribute

Gets the value and reference string (if any) of a string type attribute, if it exists

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_get_string_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    char * * string_value,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
char * *	**string_value**	Output to UF_*free*	String attribute value (Free using UF_free() after use)
logical *	**has_attribute**	Output	An attribute has been found

---

UF_ATTR_get_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_user_attribute

Gets the information from the first attribute or template, if any, that satisfies the given iterator.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_get_user_attribute
(
    tag_t object,
    const UF_ATTR_iterator_t * iter,
    UF_ATTR_info_t * info,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attribute being queried. The iterator is automatically reset.

UF_ATTR_info_t *	**info**	Output to UF_*free*	The attribute information for the first attribute (or unset template) found, if any. This struct must be freed using UF_ATTR_free_user_attribute_info_strings() after use.
logical *	**has_attribute**	Output	An attribute or an unset template has been found

UF_ATTR_get_user_attribute_lock (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_user_attribute_lock

Gets the lock information from the first attribute, that satisfies the given iterator.

Replaces: UF_ATTR_ask_lock

Environment

Internal and External

History

NX11.0

Required License(s)

gateway

```
int UF_ATTR_get_user_attribute_lock
(
    tag_t object,
    const UF_ATTR_iterator_t * iter,
    logical * is_locked,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attribute being queried. The iterator is automatically reset.
logical *	**is_locked**	Output	The lock status of the set attribute or the template (if the attribute is unset)
logical *	**has_attribute**	Output	An attribute or an unset template has been found

UF_ATTR_get_user_attribute_lock_with_title_and_type (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_get_user_attribute_lock_with_title_and_type

Gets the information from the attribute with the given title and type

Replaces: UF_ATTR_ask_locked

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_get_user_attribute_lock_with_title_and_type
(
    tag_t object,
    const char * title,
    int type,
    int index,
    logical * is_locked,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**type**	Input	The attribute type Valid values: UF_ATTR_integer UF_ATTR_real UF_ATTR_time UF_ATTR_null UF_ATTR_string UF_ATTR_bool UF_ATTR_any NOTE: If UF_ATTR_any is used, the first attribute encountered that matches the given title (and index), is returned
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
logical *	**is_locked**	Output	The lock status of the attribute. Valid only if the attribute is set.
logical *	**has_attribute**	Output	A set attribute has been found

---

UF_ATTR_get_user_attribute_with_title_and_type (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_get_user_attribute_with_title_and_type

Gets the information from the set attribute with the given title and type

Replaces: UF_ATTR_read_value, UF_ATTR_read_reference_string, UF_ATTR_find_attribute

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_get_user_attribute_with_title_and_type
(
    tag_t object,
    const char * title,
    int type,
    int index,
    UF_ATTR_info_t * info,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attribute
const char *	**title**	Input	The attribute title
int	**type**	Input	The attribute type Valid values: UF_ATTR_integer UF_ATTR_real UF_ATTR_time UF_ATTR_null UF_ATTR_string UF_ATTR_bool UF_ATTR_any NOTE: If UF_ATTR_any is used, the first attribute encountered that matches the given title (and index), is returned
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
UF_ATTR_info_t *	**info**	Output to UF_*free*	The attribute information for the first set attribute found, if any. This struct must be freed using UF_ATTR_free_user_attribute_info_strings() after use.
logical *	**has_attribute**	Output	A set attribute has been found

---

UF_ATTR_get_user_attributes (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_get_user_attributes

Get information for all the attributes that satisfy the given iterator.

This function will return an array of UF_ATTR_info_t structs that must be
freed after use with UF_ATTR_free_user_attribute_info_array()

Replaces: UF_ATTR_ask_part_attrs
NOTE: No need to use UF_ATTR_ask_part_attribute() when reading part attributes. The part tag can be used.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

int UF_ATTR_get_user_attributes
(
tag_t object,
const UF_ATTR_iterator_t * iter,
int * num_attributes,
UF_ATTR_info_t * * info
)

tag_t	**object**	Input	The object holding the attributes
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attributes being queried. The iterator is automatically reset.
int *	**num_attributes**	Output	The number of attributes returned
UF_ATTR_info_t * *	**info**	Output to UF_*free*	The information for the attributes (and/or unset templates) found. Free using UF_ATTR_free_user_attribute_info_array() after use

---

UF_ATTR_get_user_attributes_in_file (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_get_user_attributes_in_file

Get information for all the part attributes in the given file that satisfy the given iterator.
This function will return an array of UF_ATTR_info_t structs that must be
freed after use with UF_ATTR_free_user_attribute_info_array()

The given part file is temporarily partly opened to read the part attributes.

WARNING:
All values that have units will be returned in the attributes' current units.
A string with the name of the units used is returned, but the unit tag is not.
Attribute expression references will not be returned.
All reference type attributes will be returned as plain string attributes.
The flags 'inherited', 'locked', 'required' are not returned by this function.

Replaces: UF_ATTR_ask_part_attrs_in_file

Environment
Internal and External

History
NX11.0.0

Required License(s)
gateway

```
int UF_ATTR_get_user_attributes_in_file
(
    const char * part_name,
    const UF_ATTR_iterator_t * iter,
    int * num_attributes,
    UF_ATTR_info_t * * info
)
```

const char *	**part_name**	Input	The name of the part to be examined
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attributes being queried. The iterator is automatically reset.
int *	**num_attributes**	Output	The number of attributes returned
UF_ATTR_info_t * *	**info**	Output to UF_*free*	The information for the attributes (and/or unset templates) found. Free using UF_ATTR_free_user_attribute_info_array() after use

---

UF_ATTR_has_user_attribute (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_has_user_attribute

Query the object for the existence of an attribute that satisfies the given iterator

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_has_user_attribute
(
    tag_t object,
    const UF_ATTR_iterator_t * iter,
    logical * has_attribute
```

)

tag_t	**object**	Input	The object holding the attributes
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attributes being queried. The iterator is automatically reset.
logical *	**has_attribute**	Output	An attribute (or unset template) has been found

---

UF_ATTR_has_user_attribute_with_title_and_type (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_has_user_attribute_with_title_and_type

Query the object for the existence of a set attribute that satisfies the given title and type.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_has_user_attribute_with_title_and_type
(
    tag_t object,
    const char * title,
    int type,
    int index,
    logical * has_attribute
)
```

tag_t	**object**	Input	The object holding the attributes
const char *	**title**	Input	The title of the attribute
int	**type**	Input	The attribute type Valid values: UF_ATTR_integer UF_ATTR_real UF_ATTR_time UF_ATTR_null UF_ATTR_string UF_ATTR_bool UF_ATTR_any NOTE: If UF_ATTR_any is used, the first attribute encountered that matches the given title (and index), is noted as being present
int	**index**	Input	The array index ((set to UF_ATTR_NOT_ARRAY if not an array)

logical *	**has_attribute**	Output	An attribute has been found

---

UF_ATTR_init_user_attribute_info (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_init_user_attribute_info

Initializes a UF_ATTR_info_t struct as follows:

All strings are set to NULL.
All numerical values are set to 0 or 0.0.
All logical values are set to false.
All tags are set to 0

NOTE: This function must be called before the struct can be used

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_init_user_attribute_info
(
    UF_ATTR_info_p_t info
)
```

UF_ATTR_info_p_t	**info**	Input / Output	the attribute information to be initialized

---

UF_ATTR_init_user_attribute_iterator (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_init_user_attribute_iterator

Initializes a UF_ATTR_iterator_t struct as follows:

All strings are set to empty.
All logical values are set to 'false'.
The 'type' is set to 'UF_ATTR_any'.
The 'array_element_index' is set to 'UF_ATTR_ANY_ATTRIBUTE'

NOTE: If the info has pointers to strings that it does not own, such strings must be set to NULL before this function is called.

Environment

Internal and External

History
NX8.5.3

Required License(s)
gateway

int UF_ATTR_init_user_attribute_iterator
(
UF_ATTR_iterator_p_t iter
)

UF_ATTR_iterator_p_t	**iter**	Input / Output	the attribute iterator to be initialized

---

UF_ATTR_release_user_attribute_iterator (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_release_user_attribute_iterator

Resets and releases for re-use a UF_ATTR_iterator_t struct.

This must be done in two situations:

a. Before the iterator is to be reused for another iteration.
b. When the iterator is no longer in use

This function does not free the strings supplied with the iterator or the iterator itself,
but it does free internal memory associated with the iterator.

If a release is not done after the iteration is completed, the iterator may
produce unpredictable results when reused.
If a release is not done after the iterator is retired, the iterator will leak internal memory.

NOTE: Releasing an iterator twice is OK.
Iterations that are always completed (such when used in UFT_ATTR_has_user_attribute())
will automatically release the iterator. Please see the individual functions for details.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

int UF_ATTR_release_user_attribute_iterator
(
UF_ATTR_iterator_p_t iter
)

UF_ATTR_iterator_p_t	**iter**	Input / Output	the attribute iterator to be released

---

UF_ATTR_set_bool_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_set_boolean_user_attribute

Creates a boolean type attribute with the option to update or not.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_set_bool_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    logical value,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
logical	**value**	Input	The value
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_computational_time_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_set_computational_time_user_attribute

Creates a time type attribute with the option to update or not.

The time value must be entered in the current time zone of the machine running the program.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_set_computational_time_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    int value [ 2 ] ,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
int	**value [2]**	Input	The time value { day,min }
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_integer_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_set_integer_user_attribute

Creates a integer type attribute with the option to update or not.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_set_integer_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    int value,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
int	**value**	Input	The value
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_null_user_attribute (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_set_null_user_attribute

Creates a null type attribute with the option to update or not.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_set_null_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_real_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_set_real_user_attribute

Creates a real type attribute with the option to update or not.

NOTE: The supplied 'unit_type' must be consistent with the unit specification of an existing attribute or template with same title.
In other words, the attribute's unit measure cannot change, but its display units can.
If 'unit_type' are not supplied (set to NULL_TAG), then:
If an existing attribute or template is found, then the units of that attribute or template are used.
If no existing attribute or template is found, then the attribute is set to be unit-less.
If 'unit_type' are supplied, then:
If an existing attribute is found, then that attribute's display unit is modified to use the input 'unit_type'.
If no existing attribute is found, then the attribute is created with the given 'unit_type' as its display units.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_set_real_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    double value,
    tag_t unit_type,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
double	**value**	Input	The value
tag_t	**unit_type**	Input	The unit type tag (may be NULL_TAG)
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_reference_string_user_attribute (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_set_reference_string_user_attribute

Creates a string type attribute with the option to update or not.

This string attribute will be created with the given reference string.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_set_reference_string_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    const char * reference_string,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
const char *	**reference_string**	Input	The reference string
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_string_time_user_attribute (view source)

Defined in: uf_attr.h

Overview
UF_ATTR_set_string_time_user_attribute

Creates a time type attribute with the option to update or not.

The time value in either case is in the current time zone of the machine running the program.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

```
int UF_ATTR_set_string_time_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    const char * value,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
const char *	**value**	Input	Time/Date value as a formatted string
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_string_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_set_string_user_attribute

Creates a string type attribute with the option to update or not.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_set_string_user_attribute
(
    tag_t object,
    const char * title,
    int index,
    const char * value,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const char *	**title**	Input	The attribute title
int	**index**	Input	The array index (set to UF_ATTR_NOT_ARRAY if not an array)
const char *	**value**	Input	The value
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_time_string_format (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_set_time_string_format

Sets the format used to convert a time attribute value to a string (and vice versa)
This function will override (replaces) any format set via the environment variable
'UGII_DEFAULT_DATE_ATTRIBUTE_DISPLAY_FORMAT'.
To allow setting a format temporarily, this function returns the existing format.
The given format is a string that has the following tokens. Interspersed text will be copied:
%b abbreviated month name
%B full month name
%d day of the month(0-31)
%H hour(24-hour clock)(00-23)
%I hour(12-hour clock)(01-12)
%m month(01-12)
%M minute(00-59)
%S second(00-59)
%y year without century(00-99)
%Y year with century(1970-2069)

Example:
%d-%b-%Y %H:%M:%S is the format that gives 01-Mar-2011 22:16:32

If no format is set (via the environment variable or via this function), then the example format is used by default.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_set_time_string_format
(
    const char * new_format,
    char * * old_format
)
```

const char *	**new_format**	Input	The new format to be used
char * *	**old_format**	Output to UF_*free*	The format previously in used (optional - may be NULL. If not NULL, the caller is responsible for freeing the returned old format string. Free using UF_free() after use.

UF_ATTR_set_user_attribute (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_set_user_attribute

Creates or modifies an attribute with the option to update or not.

NOTE: To set a date/time type attribute, either a formatted string or a computational value may be used.
If both are set (the string is not empty ("") and the computational value is not (0,0)), then the computational value will take precedence.
The time value in either case is in the current time zone of the machine running the program.

NOTE: To set a real type attribute, there is the option to include a units specification.
The supplied 'unit_type' must be consistent with the unit specification of an existing attribute or template with same title.
In other words, the attribute's unit measure cannot change, but its display units can.
If 'unit_type' is not supplied (the tag is NULL_TAG)
If an existing attribute or template is found, then the units of that attribute or template are used.
If no existing attribute or template is found, then the attribute is set to be unit-less.
If 'unit_type' is supplied:
If an existing attribute is found, then that attribute's display unit is modified to use the input 'unit_type'.
If no existing attribute is found, then the attribute is created with the given 'unit_type' as its display units.

Replaces: UF_ATTR_assign

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_ATTR_set_user_attribute
(
    tag_t object,
    const UF_ATTR_info_t * info,
    logical update
)
```

tag_t	**object**	Input	The object receiving the attribute
const UF_ATTR_info_t *	**info**	Input	The attribute information for the attribute (make sure to initialize it first)
logical	**update**	Input	Perform an update immediately

UF_ATTR_set_user_attribute_locks (view source)

Defined in: uf_attr.h

Overview

UF_ATTR_set_user_attribute_locks

Set the locks on all the attributes that satisfy the given iterator.

NOTE: This function cannot set locks on individual array elements

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

int UF_ATTR_set_user_attribute_locks
(
tag_t object,
const UF_ATTR_iterator_t * iter,
logical locked
)

tag_t	**object**	Input	The object holding the attributes
const UF_ATTR_iterator_t *	**iter**	Input	Iterator describing the attributes to be locked. The iterator is automatically reset.
logical	**locked**	Input	The status of the lock to be set ('true': locked or 'false': unlocked)

UF_UNIT_ask_measure_type_from_unit_tag (view source)

Defined in: uf_unit.h

Overview
UF_UNIT_ask_measure_type_from_unit_tag

Given a unit tag, return the measure type

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

int UF_UNIT_ask_measure_type_from_unit_tag
(
tag_t unit,
UF_UNIT_MEASURE_TYPE_p_t measure_type
)

tag_t	**unit**	Input	The tag of the unit type
UF_UNIT_MEASURE_TYPE_p_t	**measure_type**	Output	The measure type associated with the given units (see uf_unit_types.h)

UF_UNIT_ask_system_unit_tag_from_measure (view source)

Defined in: uf_unit.h

Overview

UF_UNIT_ask_system_unit_tag_from_measure

Given a unit type name and the associated measure type, return the tag of the system unit type. The system units is that used to store values in the part file (as opposed to display units).

To properly select the units, an object in the part as well as the measure type must be provided.

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_UNIT_ask_system_unit_tag_from_measure
(
    tag_t object,
    UF_UNIT_MEASURE_TYPE_t unit_measure_type,
    tag_t * unit
)
```

tag_t	**object**	Input	The units context (an object in the part where the units are used)
UF_UNIT_MEASURE_TYPE_t	**unit_measure_type**	Input	The measure type of the units (see uf_unit_types.h)
tag_t *	**unit**	Output	The tag of the unit type

UF_UNIT_ask_unit_name_from_unit_tag (view source)

Defined in: uf_unit.h

Overview

UF_UNIT_ask_unit_name_from_unit_tag

Given a unit tag, return the unit name of the units

For example, the name for millimeters is "MilliMeter", defined in uf_unit_types.h as
'UF_UNIT_LENGTH_mm
The returned string must be freed after use.

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

int UF_UNIT_ask_unit_name_from_unit_tag
(
tag_t unit,
char * * name
)

tag_t	**unit**	Input	The tag of the unit type
char * *	**name**	Output to UF_*free*	The unit name of the units (see uf_unit_types.h)

UF_UNIT_ask_unit_tag_from_unit_name (view source)

Defined in: uf_unit.h

Overview
UF_UNIT_ask_unit_tag_from_unit_name

Given a unit type name and the associated measure type, return the tag of the unit type

The given name is the name of the units as defined in uf_unit_types.h
To properly select the units, an object in the part as well as the measure type
must be provided.

For example, the name for millimeters is "MilliMeter", defined in uf_unit_types.h as
'UF_UNIT_LENGTH_mm

Environment
Internal and External

History
NX8.5.3

Required License(s)
gateway

int UF_UNIT_ask_unit_tag_from_unit_name
(
tag_t object,
UF_UNIT_MEASURE_TYPE_t unit_measure_type,
const char * name,
tag_t * unit
)

tag_t	**object**	Input	The units context (an object in the part where the units are used)
UF_UNIT_MEASURE_TYPE_t	**unit_measure_type**	Input	The measure type of the units (see uf_unit_types.h)
const char *	**name**	Input	The name of the unit type (see uf_unit_types.h)
tag_t *	**unit**	Output	The tag of the unit type

UF_UNIT_convert_value (view source)

Defined in: uf_unit.h

Overview

UF_UNIT_convert_value

Given a value with a given unit type, convert it to a value in a new unit type.

The measure type of the new units must be the same as that of the initial units

Environment

Internal and External

History

NX8.5.3

Required License(s)

gateway

```
int UF_UNIT_convert_value
(
    double initial_value,
    tag_t initial_units,
    tag_t new_units,
    double * converted_value
)
```

double	**initial_value**	Input	The value to be converted
tag_t	**initial_units**	Input	The tag of the unit type of the initial value
tag_t	**new_units**	Input	The tag of the unit type of the new units
double *	**converted_value**	Output	The value expressed in the new units