UF_TEXT_ask_text_mode (view source)

Defined in: uf_text.h

Overview

This routine returns the current text mode for a user function program.
This routine is provided to let the user function programmer know what the mode is so that it can be changed temporarily and then reset as needed.

Environment

Internal and External

See Also

UF_TEXT_set_text_mode

Required License(s)

gateway

UF_TEXT_mode_t UF_TEXT_ask_text_mode
(

)

UF_TEXT_copy_nchars (view source)

Defined in: uf_text.h

Overview

Copy a string to a given byte count and a number of characters based on the internal NX encoding for strings. This will ensure the string is appropriately sized for calls to other UF function calls.

This is designed to work with multibyte characters to support all languages. If passed a NULL it will return an error. If the internal NX string representation exceeds either the number of bytes or the number of characters, the string will be terminated after the last full character that ensures the string is both less than the number of bytes and less than the number of characters. The value returned will indicate whether the copy was successful or not.

This routine is designed to work with the new UF definitions that now specify a character and byte count limitation.

Environment

Internal and External

History

Released in NX 10.0

Required License(s)

gateway

```
int UF_TEXT_copy_nchars
(
    const char * input_buffer,
    unsigned int output_buffer_length,
    unsigned int nchars,
    char * output_buffer
)
```

const char *	**input_buffer**	Input	The string to look at
unsigned int	**output_buffer_length**	Input	The maximum number of bytes allowed in the output string including the trailing NULL.
unsigned int	**nchars**	Input	The maximum number of characters allowed in the output string
char *	**output_buffer**	Output	The copied string

UF_TEXT_count_characters (view source)

Defined in: uf_text.h

Overview

Count the number of characters in a string. This is designed to work with multibyte characters to support all languages. If passed a NULL string the count will be returned as zero, same as if the string was empty.

Environment

Internal and External

History

Released in NX 9.0

Required License(s)

gateway

```
int UF_TEXT_count_characters
(
    const char * string_to_check,
    int * num_characters
)
```

const char *	**string_to_check**	Input	The string to look at
int *	**num_characters**	Output	The count of characters (not bytes) in the string

---

UF_TEXT_init_native_lang_support (view source)

Defined in: uf_text.h

Overview

Initializes the system for native language support.

Environment
Internal and External

History
Originally released in NX 602

Required License(s)
gateway

```
int UF_TEXT_init_native_lang_support
(
    void
)
```

UF_TEXT_load_translation_file (view source)

Defined in: uf_text.h

Overview
This routine allows the programmer to load a translation file that will be used to translate strings from English into the users native language. This file will be used in addition to the standard ugii.lng that contains the translations for NX messages and dialogs.

The Open API programmer is responsible for creating this file using the NLDMGR tool as described in the NX Internationalization and Localization online help.

Environment
Internal and External

History
Originally released in V18.0

Required License(s)
gateway

```
int UF_TEXT_load_translation_file
(
    const char * file
)
```

const char *	**file**	Input	This is the complete path to the file to be loaded. When NX displays the user interface, it will use the contents of this file to try and translate strings into the users language. Translation strings that exist in the NX language file found in $UGII_LANGUAGE/ugii.lng will always be used first. If a translation is not found in that file, then this file will be searched.

UF_TEXT_set_text_mode (view source)

Defined in: uf_text.h

Overview
This routine sets the text mode for a user function program. UF_initialize defaults the text mode to UF_TEXT_LOCALE. This routine is provided to let the user function programmer change the default behavior. When text strings are returned in the locale, International characters may be lost, if they exist in the part.

Environment
Internal and External

See Also
UF_TEXT_ask_text_mode

Required License(s)
gateway

```
int UF_TEXT_set_text_mode
(
    UF_TEXT_mode_t mode
)
```

UF_TEXT_mode_t	**mode**	Input	One of the following enumerated text mode constants: UF_TEXT_LOCALE UF_TEXT_UTF8 UF_TEXT_ALL_UTF8

UF_TEXT_translate_string (view source)

Defined in: uf_text.h

Overview
This routine allows the programmer to translate a C string from English into the users native language. The strings to translate should be contained in the file loaded using UF_TEXT_load_translation_file. If no translation is found the original string is used.

Environment
Internal and External

History
Originally released in NX 2

Required License(s)
gateway

```
int UF_TEXT_translate_string
(
    const char* source,
    int size,
```

```
    char* const xstring
)
```

const char*	**source**	Input	This is the English source string to be translated.
int	**size**	Input	size in bytes of the xstring buffer
char* const	**xstring**	Output	This is the buffer to return the string translated into the users native language. If the source string is not translated xstring receives the source string unchanged. This buffer must be allocated by the caller of UF_TEXT_translate_string.

UF_TEXT_translate_string_2 (view source)

Defined in: uf_text.h

Overview

This routine allows the programmer to translate a C string from English into the users native language. The strings to translate should be contained in the file loaded using UF_TEXT_load_translation_file. If no translation is found the original string is used. You must use UF_free to deallocate space used by xstring.

Environment

Internal and External

History

Originally released in NX 602

Required License(s)

gateway

```
int UF_TEXT_translate_string_2
(
    const char* source,
    char* * xstring
)
```

const char*	**source**	Input	This is the English source string to be translated.
char* *	**xstring**	Output to UF_*free*	This returns the string translated into the users native language. If the source string is not translated xstring receives the source string unchanged. Use UF_free to deallocate memory when done.

UF_TEXT_truncate (view source)

Defined in: uf_text.h

Overview

Truncate a string to a given byte count and a number of characters based on the internal NX encoding for strings. This will ensure the string is appropriately sized for calls to other UF function calls.

This is designed to work with multibyte characters to support all languages. If passed a NULL it will return success. If the string is shorter than the number of characters passed and the number of bytes passed then no action will be taken and success will be returned. If the internal NX string representation exceeds either the number of bytes or the number of characters, the string will be terminated after the last full character that ensures the string is both less than the number of bytes and less than the number of characters. The logical returned will indicate whether the input string was changed or not.

This routine is designed to work with the new UF definitions that now specify a character and byte count limitation, e.g.

error = UF_TEXT_truncate(filespec, MAX_FSPEC_BUFSIZE, MAX_FSPEC_NCHARS, &trunc);
will make sure a filespec is appropriately sized for UF_CFI calls.

Environment

Internal and External

History

Released in NX 9.0

Required License(s)

gateway

```
int UF_TEXT_truncate
(
    char * string_to_truncate,
    int num_bytes,
    int num_characters,
    logical * truncated
)
```

char *	**string_to_truncate**	Input / Output	The string to look at
int	**num_bytes**	Input	The maximum number of bytes allowed in the output string.
int	**num_characters**	Input	The maximum number of characters allowed in the output string
logical *	**truncated**	Output	True if the string was truncated