uc1600 (view source)

Defined in: uf_ui.h

Overview
Obtain the key in of a text string.

Environment
Internal

Return
1 = Back
2 = Cancel
3 = OK (Accept default)
5 = Data entered
8 = Disallowed state, unable to bring up dialog

Required License(s)
gateway

```
int uc1600
(
    const char * cue,
    char str [ 133 ] ,
    int * length
)
```

const char *	**cue**	Input	Menu title (displayed on the cue line, maximum of 80 characters.
char	**str [133]**	Input / Output	On input the default value of the string, on output the string typed in. This should be declared as char str[133] in the calling program.
int *	**length**	Output	The length of the returned string in ca2.

---

uc1601 (view source)

Defined in: uf_ui.h

Overview
Display a message box and wait for acknowledgement, if desired
A maximum of 132 characters can be displayed in cp1. An asterisk may be used to indicate where cp1 should be split to continue on another line when displaying to a message box. If you use an asterisk when displaying to the Status Line, the asterisk is replaced by a hyphen.

Please avoid using format escape sequences in the cp1 character string. For example, the newline escape sequence (\n) can cause an undesirable shift in the displayed text.

Environment
Internal

Required License(s)
gateway

```
void uc1601
(
    const char * message,
    int option
)
```

const char *	**message**	Input	The message to display.
int	**option**	Input	The display option: 0 = Display text to status line 1 = Display text to message box

---

uc1603 (view source)

Defined in: uf_ui.h

Overview

This routine displays a selection menu list box. You can have a maximum of 14 menu items in the list box. You can specify any one of the list items to be the default by setting the value of parameter ip2 to the desired list item. If you do not want a default list item, set ip2 = 0 and the OK button will be greyed out. If you want menu list item number 5 to be the default, then set ip2 = 5. Then menu item response values start at 5 and end at 18. If you choose menu item 1, then 5 is returned. If you select menu item 14, then 18 is returned.

Environment

Internal

Return

1 = Back
2 = Cancel Operation
5-18 = The given menu item was selected
19 = Disallowed state, unable to bring up dialog

Required License(s)

gateway

```
int uc1603
(
    const char * title,
    int default_item,
    const char items [ ] [ 38 ] ,
    int num_items
)
```

const char *	**title**	Input	Menu title (displayed on the cue line, 80 characters maximum
int	**default_item**	Input	Specify default menu item, if ip2 = 0 then no default, and the OK button will not be enabled. The range of ip2 is 0..ip4. If you have 14 menu items and set ip2 = 14, then menu item 14 will be highlighted as the default. NOTE: Changing the focused menu item with Tab or Arrow key

			does not change the default menu item. Pressing the OK button (or MB2) will select the Default item, which may not be the item that has focus.
const char	**items [ ] [38]**	Input	String array containing the menu items. If you use a hyphen(s) "-", for one of your options, this creates a nonselectable separator line. We do not recommend using the asterisk character "" in menu prompts or menu options.
int	**num_items**	Input	The number of items in the items array. This must be <= 14.

---

uc1605 (view source)

Defined in: uf_ui.h

Overview

Multiple selection menu
In internal mode, no defaults are set, and ia6 is initialized to zero.
Selecting numeric keypad #I sets ia6[I] to 1. Pressing Back exits the menu.

Environment

Internal

Return

1 = Back
2 = Cancel Operation
3 = OK
4 = Alternate Action
8 = Disallowed state, unable to bring up dialog

Required License(s)

gateway

```
int uc1605
(
    const char * cue,
    int ip2,
    const char items [ ] [ 38 ] ,
    int num_items,
    int selected [ ]
)
```

const char *	**cue**	Input	Menu title to display on the cue line. This is 80 characters maximum.
int	**ip2**	Input	Unused.
const char	**items [ ] [38]**	Input	String array containing the menu items.
int	**num_items**	Input	Number of strings in the cp3 array. This may not exceed 14 menu items.

int	**selected [ ]**	Output	Selection Array, which should be dimensioned to num_items. if ir6[n] == 0, then cp3[n][38] was not selected. if ir6[n] == 1, then cp3[n][38] was selected.

---

uc1607 (view source)

Defined in: uf_ui.h

Overview
Display keyboard entry menu and gets the integer response.

Environment
Internal

Return
1 = Back
2 = Cancel Operation
3 = OK - No user input
4 = OK with user input
8 = Disallowed state, unable to bring up dialog

Required License(s)
gateway

```
int uc1607
(
    const char * cp1,
    const char cp2 [ ] [ 16 ] ,
    int ip3,
    int ia4 [ ] ,
    int * ip5
)
```

const char *	**cp1**	Input	The message to display on the cue line, a maximum of 80 characters.
const char	**cp2 [ ] [16]**	Input	The menu list.
int	**ip3**	Input	The number of items in the menu list, cp2, this must be less than or equal to 14.
int	**ia4 [ ]**	Input / Output	On input the initial parameter values. On output the modified parameter values. This should be an array with ip3 entries. These will be the values the user can modify.
int *	**ip5**	Input	Unused.

---

uc1608 (view source)

Defined in: uf_ui.h

Overview

Display a keyboard entry menu and obtain integer or real data.

This routine displays an entry menu with integers and reals. The two arrays ia4[ip3] and ra5[ip3] are initialized with the default parameters that you specify. Both of these arrays as well as array ip6[ip3] and cp2[ip3][16] must be dimensioned to ip3. For example, if you decide to use the maximum of 14 parameter values you would dimension ia4[14], ra5[14], ip6[14], and cp2[14][16].

The array values of ip6[ip3] determine which menu entries are doubles and which are integers. For example if you set ip6[7] = 1, you are specifying the 8th menu entry to be double. This also means that the eighth value of ra5 (ra5[7]) is used as the default value. The index array value of ip6 maps to the index array values of ia4 and ra5 depending on whether you use an initial integer or double parameter value. For example, if you specify four integers and then one double you will get ra[4] as your first double, not ra[0].
The entry menu values always maintain the data type (double or int) of the initialized default values. Therefore, if you enter a double value into the entry menu where an integer was the default, the double is converted to an integer by truncating the decimal portion of the double. Similiarly, if you enter an integer where the default was a double, the integer is converted to a double.

Note: If a real number containing more than 12 decimal places is passed into this function, the underlying code will truncate it to 12 decimal places. This will then pass back a return a value of 4 (OK with user input) instead of the expected 3 (OK - No user input). By truncating your real parameter to 12 or less decimal places before calling this function, it will work as expected.

Environment

Internal

Return

1 = Back
2 = Cancel Operation
3 = OK - No user input
4 = OK with user input
8 = Disallowed state, unable to bring up dialog

Required License(s)

gateway

```
int uc1608
(
    const char * cp1,
    const char cp2 [ ] [ 16 ] ,
    int ip3,
    int ia4 [ ] ,
    double ra5 [ ] ,
    int ip6 [ ]
)
```

const char *	**cp1**	Input	Menu title to display on the cue line. This may be a maximum of 80 characters.

const char	**cp2 [ ] [16]**	Input	This is the array of menu items to display.
int	**ip3**	Input	This is the number of items in the menu array, cp2. This is also the size of the integer and double arrays.
int	**ia4 [ ]**	Input / Output	On input these are the initial parameter values. On output these are the values as modified by the user. This must be an array dimensioned to [ip3].
double	**ra5 [ ]**	Input / Output	On input these are the initial parameter values. On output these are the values as modified by the user. This must be an array dimensioned to [ip3].
int	**ip6 [ ]**	Input	This is the variable type to use for each menu item. if ip6[n] is 0, then the item is an integer. If ip6[n] is 1, then the item is a double. If an item is marked as an integer, its default value comes from ia4, and its return value is in ia4. If an item is marked as a double, its default value comes from ra5, and its return value is in ra5. This parameter must also be dimensioned as [ip3].

---

uc1609 (view source)

Defined in: uf_ui.h

Overview

Display keyboard entry menu and obtain a real response.

Note: If a real number containing more than 12 decimal places is passed into this function, the underlying code will truncate it to 12 decimal places. This will then pass back a return a value of 4 (OK with user input) instead of the expected 3 (OK - No user input). By truncating your real parameter to 12 or less decimal places before calling this function, it will work as expected.

Environment

Internal

Return

1 = Back
2 = Cancel Operation
3 = OK - No user input
4 = OK with user input
8 = Disallowed state, unable to bring up dialog

Required License(s)

gateway

```
int uc1609
(
    const char * cp1,
    const char cp2 [ ] [ 16 ] ,
    int ip3,
    double ra4 [ ] ,
    int * ip5
```

```
)
```

const char *	**cp1**	Input	Menu title displayed on the cue line. This may be a maximum of 80 characters.
const char	**cp2 [ ] [16]**	Input	This is the array of menu items to display.
int	**ip3**	Input	This is the number of items in the menu array, cp2. This is also the size of the double arrays.
double	**ra4 [ ]**	Input / Output	On input these are the initial parameter values. On output these are the values as modified by the user. This must be an array dimensioned to [ip3].
int *	**ip5**	Input	Unused

---

uc1613 (view source)

Defined in: uf_ui.h

Overview

Display keyboard entry menu and obtain an integer, real or string in response.

The displayed parameters can be any combination of integer, real, or string values. The type of variable used for displaying the parameter values are user input in array ip7.

Note: If a real number containing more than 12 decimal places is passed into this function, the underlying code will truncate it to 12 decimal places. This will then pass back a return a value of 4 (OK with user input) instead of the expected 3 (OK - No user input). By truncating your real parameter to 12 or less decimal places before calling this function, it will work as expected.

Environment

Internal

Return

1 = Back
2 = Cancel Operation
3 = OK - No user input
4 = OK with user input
8 = Disallowed state, unable to bring up dialog

Required License(s)

gateway

```
int uc1613
(
    const char * cp1,
    const char cp2 [ ] [ 16 ] ,
    int ip3,
    int ia4 [ ] ,
    double ra5 [ ] ,
```

```
    char ca6 [ ] [ 31 ] ,
    int ip7 [ ]
)
```

const char *	**cp1**	Input	Menu title displayed on the cue line. This may be a maximum of 80 characters.
const char	**cp2 [ ] [16]**	Input	This is the array of menu items to display.
int	**ip3**	Input	This is the number of items in the menu array, cp2. This is also the size of the integer, character and double arrays. Must be less than or equal to 14.
int	**ia4 [ ]**	Input / Output	On input these are the initial parameter values. On output these are the values as modified by the user. This must be an array dimensioned to [ip3].
double	**ra5 [ ]**	Input / Output	On input these are the initial parameter values. On output these are the values as modified by the user. This must be an array dimensioned to [ip3].
char	**ca6 [ ] [31]**	Input / Output	On input these are the initial parameter values. On output these are the values as modified by the user. This must be an array dimensioned to [ip3][31].
int	**ip7 [ ]**	Input	An array dimensioned to [ip3] which specifies the value type for each individual menu item. 100-199 - Indicates an integer value, which will be returned in ia4[] 200-299 - Indicates a double value which will be returned in ra5[] 300-399 - Indicates a character string value which will be returned in ca6[]

---

uc1615 (view source)

Defined in: uf_ui.h

Overview

Display the user message (cue) and returns a screen position point in parameter point (X,Y,Z) on the WCS plane in Absolute Coordinates of the Displayed Part. A typical user message might be "Pick a point."

Return

Response Returned
1 = Back
2 = Cancel
5 = Position Returned
7 = No Active Part
8 = Disallowed state, unable to bring up dialog

Environment

Internal

Required License(s)

gateway

```
int uc1615
(
    const char * cue,
    double point [ 3 ]
)
```

const char *	**cue**	Input	User Message (80 char max)
double	**point [3]**	Output	Point Picked

---

uc1616 (view source)

Defined in: uf_ui.h

Overview

Display a user message and return a point from the Point Subfunction in Absolute Coordinates of the current Work Part.

The returned coordinates are always relative to the current work view.

Return

Response Returned
1 = Back
2 = Cancel
5 = Position Returned
7 = No Active Part
8 = Disallowed state, unable to bring up dialog

Environment

Internal

History

The ia2[0] parameter was modified in V13.0 to return a value of 12 for a quadrant point.

Required License(s)

gateway

```
int uc1616
(
    const char * cue,
    int mode [ 2 ] ,
    int point_display_mode,
    double point [ 3 ]
)
```

const char *	**cue**	Input	User Message (40 char max)
int	**mode [2]**	Input / Output	On input the default selection type and offset mode. On output the selection type and offset mode used. [0] Selection Type 0 = Show Menu For User Selection (Inferred)

			1 = Cursor Location 2 = This value is ignored. 3 = This value is ignored. 4 = Existing Point 5 = End Point 6 = Control Point 7 = Intersection Point 8 = Arc/Ellipse/Sphere Center 9 = Pos On Arc/Ellipse 10 = This value is ignored. 11 = Intersection Point 12 = Quadrant Point 13 = Point on curve/Edge 14 = Point on Surface 15 = Point Between Points 16 = Cursor Location 17 = This value is ignored. [1] Offset Mode 0 = No Offset 1 = Rect Abs 2 = This value is ignored. 3 = Cylindrical 4 = Spherical 5 = 3D Vector 6 = 3D Vector
int	**point_display_mode**	Input	Temporary Point Display 0 = Display Temporary Points 1 = Do Not Display Temporary Points
double	**point [3]**	Output	Point Picked (x,y,z). This is only returned if the return code is 5.

---

uc1617 (view source)

Defined in: uf_ui.h

Overview

uc1617 object select with class selection menu
Use UF_UI_select_with_class_dialog instead.

Required License(s)

gateway

```
int uc1617
(
    const char * cp1,
    int ip2,
    tag_t * nr3,
    int * ir4,
    double * rr5
)
```

const char *	**cp1**	Input
int	**ip2**	

tag_t *	**nr3**	
int *	**ir4**	Input
double *	**rr5**	

---

uc1618 (view source)

Defined in: uf_ui.h

Overview
uc1618 simple object select. Use UF_UI_select_with_single_dialog instead.

Required License(s)
gateway

```
int uc1618
(
    const char * cp1,
    int ip2,
    tag_t * nr3,
    int * ir4,
    double * rr5
)
```

const char *	**cp1**	Input
int	**ip2**	
tag_t *	**nr3**	
int *	**ir4**	Input
double *	**rr5**	

---

uc1630 (view source)

Defined in: uf_ui.h

Overview
Interactively choose a CSYS. Though 40 characters are allowed in the title, it should be noted that the first option (ORIGIN,X-PT,Y-PT) appends the point number (PT1, PT2, and PT3) on the end of the message. Thus, only 36 characters are effectively allowed.

Return
Users Response
1 = Back
2 = Cancel
3 = OK
7 = No Active Part

8 = Disallowed state, unable to bring up dialog

Environment

Internal

Required License(s)

gateway

```
int uc1630
(
    const char * title,
    int * option,
    double csys_matrix [ 9 ] ,
    double origin [ 3 ]
)
```

const char *	**title**	Input	Menu Title (40 char max)
int *	**option**	Input / Output	On input the default CSYS Option. On output the CSYS option used. 0 = Origin, X-pt, Y-pt 1 = X-axis, Y-axis 2 = X-pt, Z-axis 3 = CSYS Of Arc/Conic 4 = WCS 5 = Offset CSYS 6 = Absolute CSYS 7 = Current View 8 = Drafting Object 9 = X-axis, Y-axis, Origin 10 = Point, Perpendicular Curve 11 = WCS 12 = Plane and Vector 13 = Three Planes 14 = Origin, X-pt, Y-pt 15 = Dynamic
double	**csys_matrix [9]**	Output	CSYS Matrix (9 element array) - relative to Absolute CSYS of Displayed Part
double	**origin [3]**	Output	Origin Of CSYS (3 element array) - relative to Absolute CSYS of Displayed Part

---

uc1652 (view source)

Defined in: uf_ui.h

Overview

Selects a view.
For modeling views only. If the current layout is a drafting layout,
the dialog will not come up and the return code will be 8.

Return

Return Code
0 = OK, View Selected

1 = Back
2 = Cancel
3 = OK, No View Selected
8 = Disallowed state, unable to bring up dialog

Environment
Internal

Required License(s)
gateway

```
int uc1652
(
    const char * title,
    char view_name [ UF_OBJ_NAME_BUFSIZE ]
)
```

const char *	**title**	Input	Menu Title (122 char max)
char	**view_name [UF_OBJ_NAME_BUFSIZE]**	Output	View Name (UF_OBJ_NAME_NCHARS char max)

---

uc1653 (view source)

Defined in: uf_ui.h

Overview
Returns the name of view in which the last screen pick occurred (either Interactive NX or Internal Open C API - uc1615, uc1616, uc1617, uc1618 or GRIP). If the view of the last position indication was removed from the current layout, the work view is returned. Using the matrix from the view (uc6433 - Read View Matrix) can help the determine which end of a curve was picked, etc.

Environment
Internal

Required License(s)
gateway

```
int uc1653
(
    char view_name [ UF_OBJ_NAME_BUFSIZE ]
)
```

char	**view_name [UF_OBJ_NAME_BUFSIZE]**	Output	View Name (UF_OBJ_NAME_NCHARS char max)

UF_UI_add_to_class_sel (view source)

Defined in: uf_ui_ugopen.h

Overview
Adds the friendly name associated with the class name of a User Defined Object (UDO) to the Type list of the class selection dialog. The class becomes selectable.

Environment
Internal and External

See Also
UF_UI_delete_from_class_sel

Required License(s)
gateway

int UF_UI_add_to_class_sel
(
unsigned int class_id
)

unsigned int	**class_id**	Input	The identifier of the class.

UF_UI_add_to_sel_list (view source)

Defined in: uf_ui.h

Overview
Add objects to the selection list.

Any objects already in the selection list are ignored.
You can use this function with UIStyler dialogs and with the UF_UI_select_with_class_dialog function. You can call UF_UI_add_to_sel_list from the constructor callback/selection initialization procedure to begin the dialog with objects already selected. The user can then review these objects, and if desired, deselect them.

The application selection callback can also call this function to add other objects to the selection list based on the objects(s) just selected. The selection filter procedure cannot call this function. You cannot use this function with UF_UI_select_with_single_dialog.
Based on the object(s) just selected, other objects may need to be selected. For example, all edges of the selected face or all faces tangent to the selected face.

Environment
Internal

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_UI_add_to_sel_list
(
    UF_UI_selection_p_t select_,
    int num,
    tag_t * objs,
    logical highlight_flag
)
```

UF_UI_selection_p_t	**select_**	Input	Selection pointer
int	**num**	Input	Number of objects to add to selection
tag_t *	**objs**	Input	Array of object tags.
logical	**highlight_flag**	Input	if true, highlight objects

UF_UI_allow_non_work_part_feature_selection (view source)

Defined in: uf_ui.h

Overview

Allows selection of features belonging to any fully loaded part. By default, only features belonging to the Work Part are selectable.

For UIStyler dialogs, this function should be called from the initialize_cb.

For use with UF_UI_select_with_class_dialog or UF_UI_select_with_single_dialog, this function should be called from the init_proc (UF_UI_sel_init_fn_t).

Note that features not belonging to the Work Part can be queried, but should not be edited in any way.

Environment

Internal

Required License(s)

gateway

```
int UF_UI_allow_non_work_part_feature_selection
(
    void* select_,
    logical allow
)
```

void*	**select_**	Input	selection pointer
logical	**allow**		

UF_UI_append_menubar_menu (view source)

Defined in: uf_ui.h

Overview

Appends a menu item to the NX menu bar. Once a menu is defined as an array of UF_UI_menubar_item_t, the custom application should call UF_UI_append_menubar_menu() from the user exit ufsta().

Once this routine is called, an Open C API license is captured and not released until NX is exited.

A program which uses UF_UI_append_menubar_menu should not use the option to unload an Open C API image. Additionally, if your code which defines UF_UI_append_menubar_menu itself loads a shared library, this code should not attempt to unload the library. NX always makes a strong attempt to prevent unloading a library which was loaded by using UF_UI_append_menubar_menu.

The USER_STARTUP (ufsta) user_exit was developed exclusively to work in conjunction with UF_UI_append_menubar_menu to define menus. If you use this routine it must be called early (i.e. prior to complete initialization of NX).

This requirement makes the use of the USER_STARTUP exit inappropriate for any purpose other than calling UF_UI_append_to_menubar. In particular, running a GRIP program, opening a part file, etc. from the shared library pointed to by the USER_STARTUP environment variable does not work because NX initialization has not yet completed at the time this library is loaded.

NOTE: This function is only available on Unix. Please look at menuscript for this functionality on Windows.

Environment

Internal

Required License(s)

gateway

```
int UF_UI_append_menubar_menu
(
    UF_UI_menubar_item_t * menu,
    UF_UI_change_state_fn_t change_state,
    char * application_name
)
```

UF_UI_menubar_item_t *	**menu**	Input	Array of UF_UI_menubar_item_t structures. This array defines the menu item, its pulldown and cascade pulldown to be appended to the NX menu bar.
UF_UI_change_state_fn_t	**change_state**	Input	Custom supplied change state function to be registered for the application. If change_state is NULL then a change state function can be registered later using UF_UI_register_change_state_fn.

char *	**application_name**	Input	Application name that pairs with the registered change state function.

UF_UI_ask_create_part_filename (view source)

Defined in: uf_ui.h

Overview
Displays the File-->New File Selection Dialog.

Environment
Internal

See Also
UF_UI_ugmgr_ask_create_part_file_name

Required License(s)
gateway

```
int UF_UI_ask_create_part_filename
(
    char file_name [ MAX_FSPEC_BUFSIZE ] ,
    int * units,
    int * response
)
```

char	**file_name [MAX_FSPEC_BUFSIZE]**	Input / Output	On input default file name to use for part creation dialog. On output the filename actually used.
int *	**units**	Input / Output	On input default units to use for part creation dialog. On output the units actually used. UF_PART_METRIC UF_PART_ENGLISH NONE = uses default
int *	**response**	Output	User response from dialog: UF_UI_OK UF_UI_CANCEL

UF_UI_ask_cursor_view (view source)

Defined in: uf_ui.h

Overview
Reads the current mask for selection within views. Object selection can be made in the work view on a drawing layout only or any view, and this routine returns the current value.

In drafting, the default is the work view so use UF_UI_set_cursor_view

if you want to select in any view.

Environment
Internal

See Also
UF_UI_set_cursor_view

Required License(s)
gateway

```
int UF_UI_ask_cursor_view
(
    int * cursor_view
)
```

int *	**cursor_view**	Output	The current cursor view: 0 = Any view 1 = Work view

---

UF_UI_ask_dialog_directory (view source)

Defined in: uf_ui.h

Overview
Reads the directory path of a given dialog.

Environment
Internal

See Also
UF_UI_dialog_dir_id_t

Required License(s)
gateway

```
int UF_UI_ask_dialog_directory
(
    UF_UI_dialog_dir_id_t dir_index,
    char * * dir_name
)
```

UF_UI_dialog_dir_id_t	**dir_index**	Input	Enumeration constant of the dialog.
char * *	**dir_name**	Output to UF_*free*	Directory path of the dialog. This must be freed by calling UF_free.

UF_UI_ask_dialog_filter (view source)

Defined in: uf_ui.h

Overview
Reads the filter extension of the current dialog.

Environment
Internal

See Also
UF_UI_dialog_dir_id_t

Required License(s)
gateway

```
int UF_UI_ask_dialog_filter
(
    UF_UI_dialog_filter_id_t dir_index,
    char * * fltr_name
)
```

UF_UI_dialog_filter_id_t	**dir_index**	Input	Enumeration constant of the dialog.
char * *	**fltr_name**	Output to UF_*free*	Filter extension of the dialog.

UF_UI_ask_global_sel_object_list (view source)

Defined in: uf_ui.h

Overview
This function will ask for the objects that are on the global selection list

Environment
Internal

History
Added in NX2

Required License(s)
gateway

```
int UF_UI_ask_global_sel_object_list
(
    int * num_objects,
    tag_t * * objects
)
```

int *	**num_objects**	Output	
tag_t * *	**objects**	Output to UF_*free*	Array of tags containing the selected objects. This array must be freed by calling UF_free.

UF_UI_ask_info_units (view source)

Defined in: uf_ui.h

Overview

Inquires what units the user would like information displayed in. These are the units used to display results in the Info functions. The units are only used for displaying results in Info and do not necessarily correspond to the part units or the units used in other Open C API functions.

Environment

Internal and External

See Also

Refer to the example

Required License(s)

gateway

```
int UF_UI_ask_info_units
(
    int * units
)
```

int *	**units**	Output	Current units: UF_UI_POUNDS_INCHES UF_UI_POUNDS_FEET UF_UI_GRAMS_MILLIMETERS UF_UI_GRAMS_CENTIMETERS UF_UI_KILOS_METERS UF_UI_KILOS_MILLIMETERS

UF_UI_ask_iw_decimal_places (view source)

Defined in: uf_ui.h

Overview

Inquires the information window (listing window) real number display preference and number of decimal places to display.

UF_UI_SYSTEM_DECIMAL_PLACES means the user has requested the system precision to be used for formatting real numbers for the information window. In this case, the number of decimal places varies based on each real value, such that the sum of the number of places before and after the decimal point equals the number of significant digits of double precision floating point accuracy that can be represented on the client machine. The decimal_places parameter should not be used when this mode is set.

UF_UI_USER_DECIMAL_PLACES means the user has asked for a specific number of decimal places. The decimal_places parameter is only valid with this mode.

Environment

Internal and External

See Also
Refer to the example

Required License(s)
gateway

int UF_UI_ask_iw_decimal_places
(
int * mode,
int * decimal_places
)

int *	**mode**	Output	Current preference: UF_UI_SYSTEM_DECIMAL_PLACES UF_UI_USER_DECIMAL_PLACES
int *	**decimal_places**	Output	Number of user defined decimal places.

UF_UI_ask_lock_status (view source)

Defined in: uf_ui.h

Overview
Query NX lock status.

This function is useful when dismissing a custom dialog and you want to determine whether or not a lock has been set. If a lock has been set then you know you need to call UF_UI_cancel_uf_dialog in order to cancel the currently displayed Open dialog or UIStyler dialog.

NOTE: If an NX's owned DA2 is currently displayed when this check is done then UF_UI_ask_lock_status returns UF_UI_LOCK. When your custom application attempts to cancel this dialog, UF_UI_cancel_uf_dialog returns an error. This failure does not cause any problems, and you should continue using this method in order to cancel any potential Open dialogs that are currently displayed.

Return
UF_UI_LOCK when NX is in lock status.
UF_UI_UNLOCK when NX is in unlock status.

Environment
Internal

See Also
Refer to the example
Refer to the example

Required License(s)
gateway

int UF_UI_ask_lock_status
(

```
    void
)
```

UF_UI_ask_minimal_graphics_window (view source)

Defined in: uf_ui.h

Overview
This function return information if NX graphics window special minimal mode is on or off

Environment
Privileged

History
Added in NX7.5.2

Required License(s)
gateway

```
int UF_UI_ask_minimal_graphics_window
(
    logical* is_set
)
```

logical*	**is_set**	Output

UF_UI_ask_open_part_filename (view source)

Defined in: uf_ui.h

Overview
Displays the File-->Open File Selection Dialog.

Environment
Internal

History
NX1 - "use_disp_file" parameter is obsolete; left in interface
for forward compatibility and changed name to "unused"

Required License(s)
gateway

```
int UF_UI_ask_open_part_filename
(
    char file_name [ MAX_FSPEC_BUFSIZE ] ,
    logical * unused,
    int * response
)
```

char	**file_name [MAX_FSPEC_BUFSIZE]**	Input / Output	On input default part name to use for part open dialog. On output the name of the part to open.
logical *	**unused**	Input	Parameter no longer used; left for forward compatibility
int *	**response**	Output	User response from dialog: UF_UI_OK UF_UI_CANCEL

UF_UI_ask_ribbon_vis (view source)

Defined in: uf_ui.h

Overview
This routine returns the current visibility of the given ribbon tab. This function only applies to ribbon tabs; for groups, galleries, cascades, and dropdowns the result is undefined.

Environment
Internal

See Also
UF_UI_set_ribbon_vis
UF_UI_create_ribbon

History
Originally released in NX9.0

Required License(s)
gateway

```
int UF_UI_ask_ribbon_vis
(
    UF_UI_ribbon_id_t ribbon_id,
    int* show
)
```

UF_UI_ribbon_id_t	**ribbon_id**	Input	Valid Ribbon id from a call to UF_UI_create_ribbon
int*	**show**	Output	1 = show; 0 = hide

UF_UI_ask_sel_cursor_pos (view source)

Defined in: uf_ui.h

Overview
Returns the view and absolute coordinates of the cursor position for the associated selection.

If an object was selected, the view returned is the view in which the object was selected. For single position, the view is the view of the cursor.

If the object was selected by name, view = NULL_TAG and the cursor position are undefined.

Environment
Internal

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_UI_ask_sel_cursor_pos
(
    UF_UI_selection_p_t select_,
    tag_t * view,
    double abs_cursor_pos [ 3 ]
)
```

UF_UI_selection_p_t	**select_**	Input	Selection pointer
tag_t *	**view**	Output	View
double	**abs_cursor_pos [3]**	Output	absolute coordinates of the cursor position

---

UF_UI_ask_sel_descriptor (view source)

Defined in: uf_ui.h

Overview
Returns a bitmask of information describing the selection that was just performed.

The valid bits which can be examined are defined in this header file.
UF_UI_SEL_DESC_SELECTION
UF_UI_SEL_DESC_DESELECTION
UF_UI_SEL_RESELECTION
UF_UI_SEL_DESC_SINGLE
UF_UI_SEL_DESC_MULTIPLE
UF_UI_SEL_DESC_SINGLE_POSITION
UF_UI_SEL_DESC_RECTANGLE_POSITION
UF_UI_SEL_DESC_NAME_SELECTION
UF_UI_SEL_DESC_RECTANGLE

Multiple bits may be set. For example, for a rectangle deselection, the DESELECTION, MULTIPLE, and RECTANGLE bits would be set.

For a name selection which selected one object, the SELECTION, SINGLE, and NAME_SELECTION bits would be set.

If a reselect was done (an object is selected and the previous object selected is deselected), only the RESELECTION bit is set.

The SINGLE and MULTIPLE bits are not set for position.

The RECTANGLE bit is set for rectangle selection, rectangle deselection, and rectangle position.

You use this function with UIStyler dialogs and UF_UI_select_by_class and can be called from either the selection filter procedure or the selection callback.

Environment
Internal

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_UI_ask_sel_descriptor
(
    UF_UI_selection_p_t select_,
    int * descriptor
)
```

UF_UI_selection_p_t	**select_**	Input	Selection pointer
int *	**descriptor**	Output	bit mask describing the selection

---

UF_UI_ask_sel_list_count (view source)

Defined in: uf_ui.h

Overview
Returns the number of objects currently selected.
You can use this function with UIStyler dialogs and with UF_UI_select_with_class_dialog.

Environment
Internal

See Also
UF_UI_remove_from_sel_list

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_UI_ask_sel_list_count
(
    UF_UI_selection_p_t select_,
    int * count
)
```

UF_UI_selection_p_t	**select_**	Input	Selection pointer
int *	**count**	Output	Count of objects selected

UF_UI_ask_sel_object_list (view source)

Defined in: uf_ui.h

Overview

Returns the number of objects selected and a pointer to an array of tags of the objects selected.

You can use this function with the UIStyler dialogs and with UF_UI_select_with_class_dialog.

Environment

Internal

See Also

UF_UI_remove_from_sel_list

History

Original release was in V13.0.

Required License(s)

gateway

```
int UF_UI_ask_sel_object_list
(
    UF_UI_selection_p_t select_,
    int * count,
    tag_p_t * objs
)
```

UF_UI_selection_p_t	**select_**	Input	Selection pointer
int *	**count**	Output	Count of objects selected
tag_p_t *	**objs**	Output to UF_*free*	Allocated array of the tags of the selected objects. Must be freed with UF_free after use.

UF_UI_ask_sel_rectangle_pos (view source)

Defined in: uf_ui.h

Overview

Returns the absolute coordinates of the rectangle cursor positions.

The view that returns is the view of the button down position.
button down position - is the position where you press and hold mouse button 1.
button up position - is the position where you release mouse button 1.

Pos1 and Pos2 are the absolute coordinates of the button down and button up positions respectively. Pos3 is the absolute coordinates of the rectangle corner which, as viewed on the screen, is horizontal with pos1. Pos4 is the absolute coordinates of the rectangle corner which, as viewed on the screen, is horizontal with pos2.

Returns an error if the last gesture was not rectangle.

Environment
Internal

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_UI_ask_sel_rectangle_pos
(
    UF_UI_selection_p_t select_,
    tag_t * view,
    double pos1 [ 3 ] ,
    double pos2 [ 3 ] ,
    double pos3 [ 3 ] ,
    double pos4 [ 3 ]
)
```

UF_UI_selection_p_t	**select_**	Input	Selection pointer
tag_t *	**view**	Output	View of button down position
double	**pos1 [3]**	Output	Absolute coordinates of the button down position
double	**pos2 [3]**	Output	Absolute coordinates of the button up position
double	**pos3 [3]**	Output	Absolute coordinates of corner of screen rectangle which is horizontal with button down position
double	**pos4 [3]**	Output	Absolute coordinates of corner of screen rectangle which is horizontal with button up position

---

UF_UI_ask_toolbar_vis (view source)

Defined in: uf_ui.h

Overview
This routine returns the current visibility of the given toolbar.

Environment
Internal

See Also
UF_UI_set_toolbar_vis
UF_UI_create_toolbar

History
Originally released in V16.0

Required License(s)
gateway

int UF_UI_ask_toolbar_vis
(
UF_UI_toolbar_id_t tool_id,
int* show
)

UF_UI_toolbar_id_t	**tool_id**	Input	Valid Toolbar id from a call to UF_UI_create_toolbar
int*	**show**	Output	1 = show; 0 = hide

UF_UI_cancel_uf_dialog (view source)

Defined in: uf_ui.h

Overview
Cancels the existing custom application dialog area 2 by sending a cancel message.

Return
UF_UI_SUCCESS when dialog area 2 is cancelled successfully or there is not DA2 displayed

UF_UI_FAILURE when dialog area 2 is not cancelled.

Dialog area 2 is not cancelled when the current dialog area 2 is not an interactive Open C API dialog.

The custom application can only bring up a new interactive Open C API dialog when UF_UI_cancel_uf_dialog returns a UF_UI_SUCCESS. The new interactive Open C API must be brought up with an XtAppAddTimeOut call. Failure to follow the above guideline can give unpredictable results.

Use UF_UI_cancel_uf_dialog for action buttons which are always available and that call interactive Open C API (i.e. the action button is not greyed out or ungreyed in the custom supplied state change function).

Environment
Internal

See Also
Refer to the example

Required License(s)
gateway

```
int UF_UI_cancel_uf_dialog
(
    int from_where
)
```

int	**from_where**	Input	Predefined constant used to support error checking. This should always be UF_UI_FROM_CUSTOM for custom applications.

UF_UI_close_listing_window (view source)

Defined in: uf_ui_ugopen.h

Overview
Closes the listing window. If in internal Open API, the window is closed. If in external Open API, then the device is set to closed.

Environment
Internal and External

See Also
UF_UI_exit_listing_window

History
For V15.0, this function was modified so that it closes the window but does not clear the windows contents. Prior to V15.0, this function closed and cleared the window in Internal Open API.

Required License(s)
gateway

```
int UF_UI_close_listing_window
(
    void
)
```

UF_UI_create_filebox (view source)

Defined in: uf_ui.h

Overview
Create an File Selection Box dialog.

Environment
Internal

Required License(s)
gateway

```
int UF_UI_create_filebox
(
    char * prompt_string,
    char * title_string,
    char filter_string [ MAX_FSPEC_BUFSIZE ] ,
    char * default_name,
    char filename [ MAX_FSPEC_BUFSIZE ] ,
    int * response
)
```

char *	**prompt_string**	Input	Text for standard prompt.
char *	**title_string**	Input	The dialog title.
char	**filter_string [MAX_FSPEC_BUFSIZE]**	Input / Output	The initial value of filter_string is used to initialize the Filter list and text field. If it contains only the pattern and not the path, then "/current_dir/pattern" is used for the filter. If it contains only the path and not the pattern, then "\path\" is used for the filter. If the path doesn't exist,"\current_dir\" is used for the filter. On Unix when OK is selected from the FSB dialog, the filter_string is the complete path+pattern of the filter. On NT, the FSB dialog doesn't have a filter edit box. Therefore, when OK is selected, the returned filter_string is always set to the folder + " + extension of the selected file. The user must allocate a buffer big enough to hold the maximum possible file filter_string, which is MAX_FSPEC_BUFSIZE bytes. This function does not support multiple patterns.
char *	**default_name**	Input	The default name is used to initialize the Selection text field. If it's an empty string or NULL string, the Selection text field is set to the current directory.
char	**filename [MAX_FSPEC_BUFSIZE]**	Output	When OK is selected from the FSB dialog, filename is whatever was typed into the Selection field. Usually, it is a complete path+name if the string is automatically set by clicking on one of the files in the filter list. Note: An error message displays if no filename is entered; even if there is a directory path in the selection field.

			It is caller's responsibility to allocate enough memory for the filename string.
int *	**response**	Output	UF_UI_OK: OK was selected UF_UI_CANCEL: CANCEL was selected

---

UF_UI_create_filebox_with_multiple_filters (view source)

Defined in: uf_ui.h

Overview
This will creates a File Selection Box dialog which also supports multiple file selection filters.

Return
0 = No error
!= 0 = Error code

Environment
Internal

Required License(s)
gateway

```
int UF_UI_create_filebox_with_multiple_filters
(
    char* prompt_string,
    char* title_string,
    char* * file_extensions,
    int num_extensions,
    char * default_name,
    char filename [ MAX_FSPEC_BUFSIZE ] ,
    int* response
)
```

char*	**prompt_string**	Input	Text for standard prompt.
char*	**title_string**	Input	The dialog title.
char* *	**file_extensions**	Input	Caller has to supply the list of file extension strings. The elements of the list will decide which type of files need to be displayed in the selection dialog box for selection. The file extensions should be in the format ".XXX". E.g ".prt", ".sim", ".dat" etc.
int	**num_extensions**	Input	The variable will provide the size of the file extensions list. It should not be more than 32.
char *	**default_name**	Input	The default name is used to initialize the Selection text field. If it's an empty string or NULL string, the Selection text field is set to the current directory.

char	**filename [MAX_FSPEC_BUFSIZE]**	Output	When OK is selected from the FSB dialog, filename is whatever was typed into the Selection field. Usually, it is a complete path+name if the string is automatically set by clicking on one of the files in the filter list. Note: An error message displays if no filename is entered; even if there is a directory path in the selection field. It is caller's responsibility to allocate enough memory for the filename string.
int*	**response**	Output	UF_UI_OK: OK was selected UF_UI_CANCEL: CANCEL was selected

---

UF_UI_create_part (view source)

Defined in: uf_ui.h

Overview

Creates a part and makes it the work part using the File-->New File Selection Dialog.

Note: This function will not work when using TCIN (NXManager). To create a new part for for that mode, call UF_UI_ugmgr_ask_create_part_name followed by UF_UGMGR_new_part_from_template.

Environment

Internal

Required License(s)

gateway

```
int UF_UI_create_part
(
    const UF_UI_err_p_t error_fn,
    char file_name [ MAX_FSPEC_BUFSIZE ] ,
    int * units,
    tag_t * part,
    int * response
)
```

const UF_UI_err_p_t	**error_fn**	Input	Pointer to data structure and user-defined error handling function.
char	**file_name [MAX_FSPEC_BUFSIZE]**	Input / Output	On input the default part name to use for the part open dialog. On output the name of the part to open
int *	**units**	Input / Output	On input the default units to use for part creation dialog, UF_PART_METRIC or UF_PART_ENGLISH. On output the units of the newly created part.

tag_t *	**part**	Output	Pointer to tag of newly created part (only valid if response = UF_UI_OK)
int *	**response**	Output	User response from dialog: UF_UI_OK UF_UI_CANCEL

UF_UI_create_ribbon (view source)

Defined in: uf_ui.h

Overview

Creates a ribbon from the given .rtb/.gly/.grb file. The file name should not have any hard coded path and should exist in one of the Open application directories. The show parameter is only used to show or hide the ribbon when it is loaded for the first time. On all subsequent loads, the show/hide value as recorded in the users registry is used.

In order to be successfully loaded, the .rtb/.gly/.grb file must be located in the application subdirectory of one of the directories listed in the file pointed to by UGII_CUSTOM_DIRECTORY_FILE, which defaults to $UGII_BASE_DIR/ugii/menus/custom_dirs.dat.

Example:
UF_UI_ribbon_id_t id = NULL;

error = UF_UI_create_ribbon("my.rtb", 1, &id);

Environment

Internal

See Also

UF_UI_remove_ribbon

History

Originally released in NX9.0

Required License(s)

gateway

```
int UF_UI_create_ribbon
(
    char* file_name,
    int show,
    UF_UI_ribbon_id_t * ribbon_id
)
```

char*	**file_name**	Input	The .rtb/.gly/.grb file name without any hard coded path
int	**show**	Input	Initial visibility of the ribbon (1 = show; 0 = hide)
UF_UI_ribbon_id_t *	**ribbon_id**	Output to UF_*free*	ribbon id if creation is successful

UF_UI_create_toolbar (view source)

Defined in: uf_ui.h

Overview

Creates a toolbar from the given .tbr file. The file name should not have any hard coded path and should exist in one of the Open application directories. The show parameter is only used to show or hide the toolbar when it is loaded for the first time. On all subsequent loads, the show/hide value as recorded in the users registry is used.

In order to be successfully loaded, the .tbr file must be located in the application subdirectory of one of the directories listed in the file pointed to by UGII_CUSTOM_DIRECTORY_FILE, which defaults to $UGII_BASE_DIR/ugii/menus/custom_dirs.dat.

Example:
UF_UI_toolbar_id_t id = NULL;

error = UF_UI_create_toolbar("my.tbr", 1, &id);

Environment

Internal

See Also

UF_UI_remove_toolbar

History

Originally released in V16.0

Required License(s)

gateway

```
int UF_UI_create_toolbar
(
    char* file_name,
    int show,
    UF_UI_toolbar_id_t * tool_id
)
```

char*	**file_name**	Input	The .tbr file name without any hard coded path
int	**show**	Input	Initial visibility of the toolbar (1 = show; 0 = hide)
UF_UI_toolbar_id_t *	**tool_id**	Output to UF_*free*	Toolbar id if creation is successful

UF_UI_create_usertool (view source)

Defined in: uf_ui.h

Overview
Loads a User Tool dialog. This function replaces previously loaded definitions. Information on user tools can be found in the NX documentation under System administration -> Customizing the NX installation.

Environment
Internal

See Also
Refer to the example

Required License(s)
gateway

int UF_UI_create_usertool
(
int tool_num,
char * filename,
logical map_flag,
logical * read_flag
)

int	**tool_num**	Input	Number of tool to load
char *	**filename**	Input	Pointer to tool definition file specification name (132 char max)
logical	**map_flag**	Input	String defined flag: UF_UI_SHOW = Display tool UF_UI_HIDE = Do not display tool
logical *	**read_flag**	Output	Syntax error flag: TRUE = Syntax error FALSE = No syntax error for file read

---

UF_UI_delete_from_class_sel (view source)

Defined in: uf_ui.h

Overview
Removes the friendly name associated with the class name of a User Defined Object (UDO) from the Type list of the class selection dialog. The class becomes nonselectable.

Environment
Internal

See Also
UF_UI_add_to_class_sel

Required License(s)
gateway

int UF_UI_delete_from_class_sel
(

```
    unsigned int class_id
)
```

unsigned int	**class_id**	Input	The identifier of the class.

UF_UI_disable_quick_access (view source)

Defined in: uf_ui.h

Overview

Quick Access Menus, provides for enabling global functions and special functions in many instances where they were previously greyed out. This is done by
providing an automatic cancel feature when these menu items are chosen. The two most common cases are:
- with a special function dialog open, Quick Access Menus will allow a different special function to be chosen. The original special function is canceled and replace by the new one.
- With a tool palette open in DA1 and one of its dialogs in DA2, you can now choose another tool palette or global function from the menubar. The DA2 will be canceled, and the DA1 wll be replaced by the the new DA1.

The Quick Access Menus feature may not work correctly with some applications written that directly use Motif dialogs, use certain User Function dialogs or which provide a confirmation step when a dialog is canceled.

To disable Quick Access Menus in your application, call UF_UI_disable_quick_access in your application's enter routine and UF_UI_enable_quick_access in your application's exit routine.

Environment

Internal

History

V12

Required License(s)

gateway

```
int UF_UI_disable_quick_access
(
    void
)
```

UF_UI_dismiss_dialog_area_2 (view source)

Defined in: uf_ui.h

Overview

Dismisses the interactive Open C API dialog after you select cancel.

You use this routine after a call to any Open C dialog if it is not wrappered with UF_UI_lock_ug_access and UF_UI_unlock_ug_access. This call guarantees that the Open dialog is dismissed when the user select's one of its navigation buttons.

Calling this function after an Open dialog has been displayed when it is not needed should not cause any negative side affects.

This function is not needed after a UIStyler dialog has been displayed. Below is an example of UF_UI_dismiss_dialog_area_2. This creates a custom dialog via UF_UI_run_dialog that only has a pulldown item in it called File. File has two pulldown items: Selection and Quit. The Selection item launches UF_UI_select_by_class. The lock wrappers are used to wrapper the start of the custom dialog and the end. Because of this location of the lock wrappers it is necessary to call UF_UI_dismiss_dialog_area_2.

Return
This function

Returns
UF_UI_SUCCESS when dialog area 2 is dismissed successfully. UF_UI_FAILURE when dialog area 2 is not dismissed. Dialog area 2 is not dismissed when the current dialog area 2 is not an interactive Open C API dialog or there is no DA2 displayed.

Environment
Internal

See Also
Refer to the example

Required License(s)
gateway

```
int UF_UI_dismiss_dialog_area_2
(
    void
)
```

UF_UI_display_nonmodal_msg (view source)

Defined in: uf_ui.h

Overview
Displays a nonmodal Message Dialog with an OK button. A message box displays with the character string inputted through the message argument. The character string message is written to the syslog. You can call this routine multiple times to display more than one message dialog. While the message dialog displays, your Open C API program continues to execute. You can dismiss the dialog box by clicking the OK button only after your program finishes its execution.

Environment

Internal

See Also
Refer to the example

Required License(s)
gateway

int UF_UI_display_nonmodal_msg
(
char * title_string,
char * message,
int pos_method
)

char *	**title_string**	Input	Dialog title string
char *	**message**	Input	The message string to be displayed in the Message Dialog. Use "\n" within the string if multiple lines are desired.
int	**pos_method**	Input	Position method: UF_UI_MSG_POS_CURSOR = the dialog's OK button will be under the mouse cursor. UF_UI_MSG_POS_CASCADE = the dialog will be positioned diagonally accross the screen relative to other non-modal message dialogs. UF_UI_MSG_POS_BOTTOM_RIGHT = the dialog will be positioned at the bottom right corner of the graphics window. UF_UI_MSG_POS_TOP_LEFT = the dialog will be positioned at the top left corner of the screen.

UF_UI_display_url (view source)

Defined in: uf_ui.h

Overview
Displays the specified URL in the NX internal web browser if it is available, or otherwise in the user's defined default web browser.

Environment
Internal

History
Originally added in v19.0

Required License(s)
gateway

```
int UF_UI_display_url
(
    const char * url
)
```

const char *	**url**	Input	Pointer to URL to display.

UF_UI_display_url_and_activate (view source)

Defined in: uf_ui.h

Overview
Displays the specified URL in the NX internal web browser, bringing it to the front if it is available. Otherwise displays in the user's defined default web browser.

Environment
Internal

History
Originally added in NX 8.5

Required License(s)
gateway

```
int UF_UI_display_url_and_activate
(
    const char * url
)
```

const char *	**url**	Input	Pointer to URL to display.

UF_UI_display_usertool (view source)

Defined in: uf_ui.h

Overview
Displays or removes the specified User Tool dialog from the screen. The only valid tool_num is 0. Closes and clears the listing window.

Environment
Internal

See Also
UF_UI_close_listing_window
Refer to the example

History
This function was originally released in V15.0.

Required License(s)

gateway

int UF_UI_display_usertool
(
int tool_num,
logical map_flag
)

int	**tool_num**	Input	Number of tool to display or remove
logical	**map_flag**	Input	String defined flag: UF_UI_SHOW = Display tool UF_UI_HIDE = Remove display of tool

---

UF_UI_enable_quick_access (view source)

Defined in: uf_ui.h

Overview

Quick Access Menus, provides for enabling global functions and special functions in many instances where they were previously greyed out. This is done by providing an automatic cancel feature when these menu items are chosen. The two most common cases are:
- with a special function dialog open, Quick Access Menus will allow a different special function to be chosen. The original special function is canceled and replace by the new one.
- With a tool palette open in DA1 and one of its dialogs in DA2, you can now choose another tool palette or global function from the menubar. The DA2 will be canceled, and the DA1 wll be replaced by the the new DA1.

The Quick Access Menus feature may not work correctly with some applications written that directly use Motif dialogs, use certain User Function dialogs or which provide a confirmation step when a dialog is canceled.

To disable Quick Access Menus in your application, call UF_UI_disable_quick_access in your application's enter routine and UF_UI_enable_quick_access in your application's exit routine.

Environment

Internal

History

V12

Required License(s)

gateway

int UF_UI_enable_quick_access
(
void
)

UF_UI_exit_listing_window (view source)

Defined in: uf_ui_ugopen.h

Overview
Closes and clears the listing window. If in internal Open API, the window is closed and cleared. If in external Open API, the device is set to closed.

Environment
Internal and External

See Also
UF_UI_close_listing_window

History
This function was originally released in V15.0.

Required License(s)
gateway

```
int UF_UI_exit_listing_window
(
    void
)
```

UF_UI_get_DA1_coords (view source)

Defined in: uf_ui.h

Overview
Retrieves the coordinates of the Dialog Area 1 window.

This function retrieves the current position of DA1.
Therefore, if the user has moved the DA1 dialog then these new coordinates are the ones retrieved.

Environment
Internal

See Also
Refer to the example

Required License(s)
gateway

```
int UF_UI_get_DA1_coords
(
    short * x,
    short * y
)
```

short *	**x**	Output	DA1 x-coordinate
short *	**y**	Output	DA1 y-coordinate

UF_UI_get_DA2_coords (view source)

Defined in: uf_ui.h

Overview

Retrieves the coordinates of the Dialog Area 2 window. This API returns UF_UI_SUCCESS if everything goes fine, otherwise, use UF_get_fail_message to determine the exact cause of the failure. NOTE: This function retrieves the current position of DA2. Therefore, if the user has moved the DA2 dialog then these new coordinates are the ones retrieved.

Environment

Internal

See Also

Refer to the example

Required License(s)

gateway

```
int UF_UI_get_DA2_coords
(
    short * x,
    short * y
)
```

short *	**x**	Output	DA2 x-coordinate
short *	**y**	Output	DA2 y-coordinate

UF_UI_get_default_parent (view source)

Defined in: uf_ui.h

Overview

Retrieves the window to be used as the parent of any user defined dialogs in NX. This way the user defined dialog physically behaves as though it is one of NX's dialog. For example it is not able to hide behind the graphics window and it iconifies when NX is iconified.

Return

This routine returns a void pointer which is the Window to be used as the parent of user defined dialogs. On Unix you must type cast this to a Widget. On NT you must type cast this to an HWND.

Environment
Internal

See Also
This is a code fragment showing the essence of how to use this function on Unix.

Required License(s)
gateway

void * UF_UI_get_default_parent
(
void
)

UF_UI_init_attachments (view source)

Defined in: uf_ui.h

Overview
Initialize the attachments structure.

Environment
Internal

History
Originally added in v19.0

Required License(s)
gateway

int UF_UI_init_attachments
(
UF_UI_attachment_p_t attach
)

UF_UI_attachment_p_t	**attach**	Input	Pointer to the structure to initialize.

UF_UI_is_listing_window_open (view source)

Defined in: uf_ui_ugopen.h

Overview
Queries whether the Information window is open or closed. This function returns a value of FALSE if the Information window: has never been opened, has been closed with File-->Exit, File-->Close, has been closed by a call to UF_UI_close_listing_window or by selecting the Information button.

Environment

Internal and External

Required License(s)
gateway

int UF_UI_is_listing_window_open
(
logical * response
)

logical *	**response**	Output	Response flag: TRUE = Window is open FALSE = Window is closed

---

UF_UI_is_object_in_sel_list (view source)

Defined in: uf_ui.h

Overview
Inquires if object is selected.
You use this function with dialogs created with the UIStyler and with the UF_UI_select_with_class_dialog function.

Environment
Internal

See Also
UF_UI_remove_from_sel_list

History
Original release was in V13.0.

Required License(s)
gateway

int UF_UI_is_object_in_sel_list
(
UF_UI_selection_p_t select_,
tag_t object,
logical * in_list
)

UF_UI_selection_p_t	**select_**	Input	Selection pointer
tag_t	**object**	Input	Object tag
logical *	**in_list**	Output	If true, object is in the selection list, else false.

UF_UI_lock_ug_access (view source)

Defined in: uf_ui.h

Overview

Inhibits access to NX dialog area 1 and the appropriate menu items before an Interactive Open C API dialog is brought up. Both the menubar and the dialog area 1 are greyed out during the lock. UF_UI_lock_ug_access

Returns

UF_UI_LOCK_EXISTS when dialog area 1 and the NX menu bar is already inhibited.

UF_UI_LOCK_SET when dialog area 1 and the NX menu bar is inhibited successfully.

UF_UI_LOCK_ERROR when another application or NX owns the DA2

It is very important to always check your return status of UF_UI_lock_ug_access. If it is not UF_UI_LOCK_SET then you do not want to continue with attempting to bring up a dialog. You should also only call UF_UI_unlock_ug_access when a successful lock has been made.

UF_UI_lock_ug_access() and UF_UI_unlock_ug_access() must be used to surround a sequence of one or more Interactive Open C API calls, when called from a custom dialog.

NOTE: This function does not have to be used by any UIStyler dialogs that launch other type of Presentation APIs.

Return

See the return values in the description section.

Environment

Internal

See Also

Refer to the example

Required License(s)

gateway

```
int UF_UI_lock_ug_access
(
    int from_where
)
```

int	**from_where**	Input	Predefined constant used to support error checking. This should always be UF_UI_FROM_CUSTOM for custom applications.

UF_UI_message_dialog (view source)

Defined in: uf_ui.h

Overview

This routine will display a modal message dialog which supports the following dialog types: Question, Information, Error and Warning.

Multiple strings may be provided using message and num_messages. These strings will be placed on individual lines within the dialog.

If the buttons argument is set to NULL, the dialog will display the OK button only. Otherwise, you may customize the label and the return value of at most 3 buttons on the dialog. Note that return values must be within the range of 1 thru 100.

The translate flag will attempt to locate the strings within the NX Native Language Database. If it cannot locate the string, it will simply display the provided string.

Return

0 = No error
not 0 = Error code

Environment

Internal

History

Originally released in v18.0

Required License(s)

gateway

```
int UF_UI_message_dialog
(
    char * title_string,
    UF_UI_MESSAGE_DIALOG_TYPE dialog_type,
    char * * messages,
    int num_messages,
    logical translate,
    UF_UI_message_buttons_t * buttons,
    int * response
)
```

char *	**title_string**	Input	Dialog title displayed in title bar
UF_UI_MESSAGE_DIALOG_TYPE	**dialog_type**	Input	Indicates the type of dialog type
char * *	**messages**	Input	Messages to display in dialog
int	**num_messages**	Input	Number of messages associated with messages.
logical	**translate**	Input	Flag to translate messages.
UF_UI_message_buttons_t *	**buttons**	Input	Definitions of buttons to display within the dialog. The response field must be within 1-100.
int *	**response**	Output	Response from the dialog

UF_UI_open_listing_window (view source)

Defined in: uf_ui_ugopen.h

Overview

Opens and manages (displays) a motif style Information window if in internal Open API mode. The first time this window is opened it will be empty. Further calls will display the information previously displayed in the window. If in external Open API mode, sets the listing window flag to open.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UI_open_listing_window
(
    void
)
```

UF_UI_open_part (view source)

Defined in: uf_ui.h

Overview

Opens a part and makes it the work part using the File-->Open File Selection Dialog.

Environment

Internal

See Also

Refer to the example

History

NX1 - "use_disp_file" parameter is obsolete; left in interface for forward compatibility and changed name to "unused"

Required License(s)

gateway

```
int UF_UI_open_part
(
    const UF_UI_err_p_t error_fn,
    char file_name [ MAX_FSPEC_BUFSIZE ] ,
    logical * unused,
    tag_t * part,
    int * response,
    UF_PART_load_status_t * error_status
)
```

const UF_UI_err_p_t	**error_fn**	Input	Pointer to data structure and user-defined error handling function.
char	**file_name [MAX_FSPEC_BUFSIZE]**	Input / Output	On input default part name to use for part open dialog. On output the name of the part to open.
logical *	**unused**	Input	Parameter is no longer used; left for forward compatibility
tag_t *	**part**	Output	Pointer to tag of part that was just opened (only valid if response = UF_UI_OK).
int *	**response**	Output	User response from dialog: UF_UI_OK UF_UI_CANCEL
UF_PART_load_status_t *	**error_status**	Output to UF_*free*	The user allocated structure <error_status> is filled with the names and associated error codes of any parts that did not load correctly. The allocated arrays must be freed with UF_free_string_array and UF_free(). For details see the definition of UF_PART_load_status_t.

UF_UI_point_construct (view source)

Defined in: uf_ui.h

Overview

Main entry point for the associative point constructor dialog.
Creates an associative point in the work part and returns the point and coordinates of the point in terms of the absolute work part coordinate system.

The dialog will create a non-associative point in certain cases, depending upon the setting of the Reference menu in the Coordinates area of the dialog:

- If the Reference menu is set to WCS or Absolute Display Part and the work part is different from the display part and coordinates are typed into the Coordinates input fields, a non-associative point will be created.

- If the Reference menu is set to WCS or Absolute Display Part and the work part is different from the display part and offsets are typed into any of the Offset input fields, a non-associative point will be created.

Environment

Internal

See Also

uc1616
UF_SO_set_visibility_option

History
Originally added in v19.0

Required License(s)
gateway

```
int UF_UI_point_construct
(
    char * cue,
    UF_UI_POINT_base_method_t * base_method,
    tag_t * point_tag,
    double base_pt [ 3 ] ,
    int * response
)
```

char *	**cue**	Input	Displays a message in the cue line.
UF_UI_POINT_base_method_t *	**base_method**	Input / Output	Use this method as the default method. Returns the method used to create the point.
tag_t *	**point_tag**	Output	Constructed point. The point is an associative (SO) point that is invisible by default. Use UF_SO_set_visibility_option to make it visible. Use functions in uf_so.h to query the parents. See information above in the description about circumstances that will cause a non-associative point to be returned. NULL_TAG is returned if dialog is terminated with CANCEL.
double	**base_pt [3]**	Output	Coordinates of created point, in absolute work part coordinates.
int *	**response**	Output	One of the following: UF_UI_OK UF_UI_BACK UF_UI_CANCEL.

UF_UI_register_change_state_fn (view source)

Defined in: uf_ui.h

Overview
Registers or unregisters the state change callback for a persistent dialog. Your state change callback can manage the greying and ungreying of the action buttons in its persistent Motif dialogs. Call UF_UI_register_state_fn with the address of the change state callback registers the function. Calling UF_UI_register_state_fn with a NULL first argument unregisters the state change callback. NX activates the registered state change function at the appropriate times to grey or ungrey the custom persistent action button. You must unregister your change state function when your custom dialog is exiting.

NOTE: A state change is the locking and unlocking of NX main menu bar. If you are in your own custom application that has launched an Open dialog and the user goes into an all purpose function dialog (such as Layer-->Settings or Info-->Object), then this is still considered a lock state so the function in UF_UI_register_change_state_fn does not get called.

Return
Returns one of the following values:
UF_UI_FAILURE = failure
UF_UI_SUCCESS = success

Environment
Internal

See Also
Refer to the example

Required License(s)
gateway

```
int UF_UI_register_change_state_fn
(
    UF_UI_change_state_fn_t change_state,
    char * application_name
)
```

UF_UI_change_state_fn_t	**change_state**	Input	Custom supplied change state function to be registered for the application. NX unregisters the this function when UF_UI_register_change_state_fn is called with a NULL parameter.
char *	**application_name**	Input	Application name that pairs with the registered change state function.

UF_UI_remove_all_from_sel_list (view source)

Defined in: uf_ui.h

Overview
Removes all the objects from the selection list and optionally unhighlights them.

You use this function with dialogs created with the UIStyler and with the UF_UI_select_with_class_dialog function.

Environment
Internal

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_UI_remove_all_from_sel_list
(
    UF_UI_selection_p_t select_,
    logical unhighlight
)
```

UF_UI_selection_p_t	**select_**	Input	Selection pointer
logical	**unhighlight**	Input	If true, unhighlight the objects

---

UF_UI_remove_from_sel_list (view source)

Defined in: uf_ui.h

Overview

Removes objects from the selection list.

This function is to be used with dialogs created with the UIStyler and with the UF_UI_select_with_class_dialog function. It can be called from the selection callback. It cannot be called from the selection filter procedure.

If any of the objects are not in the list, an error is returned and NO objects are removed from the list.

It could be called by the selection callback to remove objects from the selection list. For example, based on objects just deselected, the application may need to remove other associated objects from the selection list.

Environment

Internal

See Also

Refer to the example

History

Original release was in V13.0.

Required License(s)

gateway

```
int UF_UI_remove_from_sel_list
(
    UF_UI_selection_p_t select_,
    int num,
    tag_t * objs,
    logical unhighlight
)
```

UF_UI_selection_p_t	**select_**	Input	Selection pointer
int	**num**	Input	Number of objects to remove
tag_t *	**objs**	Input	Array of object tags.

logical	**unhighlight**	Input	If true, unhighlight the objects

---

UF_UI_remove_ribbon (view source)

Defined in: uf_ui.h

Overview

Removes the ribbon with the given id. Once the ribbon is removed the ribbon id should not be used.

Example:

```
if (id)
UF_UI_remove_ribbon(id);
id = NULL;
```

Environment

Internal

See Also

UF_UI_create_ribbon

History

Originally released in NX9.0

Required License(s)

gateway

```
int UF_UI_remove_ribbon
(
    UF_UI_ribbon_id_t ribbon_id
)
```

UF_UI_ribbon_id_t	**ribbon_id**	Input	ribbon id of the ribbon to be removed

---

UF_UI_remove_toolbar (view source)

Defined in: uf_ui.h

Overview

Removes the toolbar with the given id. Once the toolbar is removed the toolbar id should not be used.

Example:

```
if (id)
UF_UI_remove_toolbar(id);
id = NULL;
```

Environment

Internal

See Also
UF_UI_create_toolbar

History
Originally released in V16.0

Required License(s)
gateway

int UF_UI_remove_toolbar
(
UF_UI_toolbar_id_t tool_id
)

UF_UI_toolbar_id_t	**tool_id**	Input	Toolbar id of the toolbar to be removed

---

UF_UI_resume_create_toolbar (view source)

Defined in: uf_ui.h

Overview
This routine must be used to wrapper the creation of multiple toolbars. The use of this function helps with the positioning of the toolbars when they are docked.

Example:

```
UF_UI_suspend_create_toolbar();
UF_UI_create_toolbar("file1.tbr",1, &id1);
UF_UI_create_toolbar("file2.tbr",1, &id2);
UF_UI_create_toolbar("file3.tbr",1, &id3);
UF_UI_resume_create_toolbar();
```

Environment
Internal

See Also
UF_UI_suspend_create_toolbar

History
Originally released in V16.0

Required License(s)
gateway

int UF_UI_resume_create_toolbar
(
void
)

UF_UI_resume_init_appstate (view source)

Defined in: uf_ui.h

Overview

This routine must be used to wrapper the creation of multiple toolbars. The use of this function helps with the positioning of the toolbars when they are docked. It restore toolbar state for current application.

Example:

```
UF_UI_suspend_init_appstate();
UF_UI_create_toolbar("file1.tbr",1, &id1);
UF_UI_create_toolbar("file2.tbr",1, &id2);
UF_UI_create_toolbar("file3.tbr",1, &id3);
UF_UI_resume_init_appstate();
```

Environment

Internal

See Also

UF_UI_suspend_init_appstate

History

Originally released in V18.0

Required License(s)

gateway

```
int UF_UI_resume_init_appstate
(
    void
)
```

UF_UI_resume_remove_toolbar (view source)

Defined in: uf_ui.h

Overview

This routine must be used to wrapper the removing of multiple toolbars. The use of this function helps with the correct recording of the positions of the docked toolbars in the registry.

Example:

```
UF_UI_suspend_remove_toolbar();
if (id1) UF_UI_remove_toolbar(id1);
if (id2) UF_UI_remove_toolbar(id2);
if (id3) UF_UI_remove_toolbar(id3);
UF_UI_resume_remove_toolbar();

id1 = NULL;
id2 = NULL;
id3 = NULL;
```

Environment

Internal

See Also
UF_UI_suspend_remove_toolbar

History
Originally released in V16.0

Required License(s)
gateway

```
int UF_UI_resume_remove_toolbar
(
    void
)
```

UF_UI_route_invoke_callback (view source)

Defined in: uf_ui_route.h

Overview
Execute Routing callback functions(internal\external) on passed in objects.

There must be an active part for this function to be called.

Environment
Internal

History
Original release was in NX404 IP2 MP4

Required License(s)
gateway

```
int UF_UI_route_invoke_callback
(
    char * call_back_name,
    int num_objects,
    tag_t* objects
)
```

char *	**call_back_name**	Input	Callback Name The list of available callbacks is documented in the ugroute_mech_mm.xml file under the Callbacks section.
int	**num_objects**	Input	Number of objects being passed to the callback
tag_t*	**objects**	Input	Array of objects to pass

UF_UI_save_listing_window (view source)

Defined in: uf_ui_ugopen.h

Overview
Saves the contents of the Information window to the file specified by the argument.

Note: The saved listing window information is limited to 256 characters per line. If a line is longer than 256 characters, it will be wrapped to multiple lines in the saved file.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UI_save_listing_window
(
    char * filename
)
```

char *	**filename**	Input	filename to which Information Window contents is written.

UF_UI_select_by_class (view source)

Defined in: uf_ui.h

Overview
NOTE: This function is to be obsoleted in the near future. Please use the replacement function UF_UI_select_with_class_dialog
Selects multiple objects.

Environment
Internal

See Also
Refer to the example
UF_UI_selection_options_t

Required License(s)
gateway

```
int UF_UI_select_by_class
(
    char * message,
    UF_UI_selection_options_p_t opts,
    int * response,
    int * count,
    tag_p_t * object
)
```

char *	**message**	Input	Cue line message to display.

UF_UI_selection_options_p_t	**opts**	Input	Selection options.
int *	**response**	Output	response: 1 = Back 2 = Cancel 3 = OK
int *	**count**	Output	Count of objects: 0 = No object selected
tag_p_t *	**object**	Output to UF_*free*	Object identifiers of selected objects. NULL if no object selected. Use UF_free to deallocate memory.

UF_UI_select_conehead (view source)

Defined in: uf_ui.h

Overview

Displays the given coneheads and allows the user to select one of them. The coneheads are described using the standard conehead attributes. Each conehead has a selection point. The location of the selection point is set by the input parameter selection_point. The user is then prompted for a screen location, and the conehead whose selection point is closest, is returned to the caller.

The selection_point parameter may be any value in the range 0.0 to 1.0. A value of 0.0 sets the selection point to the base of each conehead. 1.0 sets the selection point to the tips.

The display_coneheads flag determines if, all the coneheads, none of the coneheads, or just the selected conehead, is left displayed when the function ends. The selected coneheads selection point is marked with a small circle.

The user can choose Back or Cancel without selecting a conehead. UF_DISP_set_conehead_attrb describes how to set conehead attributes.

Environment

Internal

See Also

UF_DISP_set_conehead_attrb

Required License(s)

gateway

```
int UF_UI_select_conehead
(
    char * message,
    int num,
    double origins [ ] ,
    double directions [ ] ,
    char * * labels,
    UF_DISP_conehead_attrb_s * attributes,
    double selection_point,
```

```
    int display_coneheads,
    int * selected_num,
    int * response
)
```

char *	**message**	Input	The cue line message. (80 character maximum)
int	**num**	Input	The number of coneheads to display.
double	**origins [ ]**	Input	An array of conehead origin points. Contains num3 entries.
double	**directions [ ]**	Input	An array of conehead directions. Contains num3 entries.
char * *	**labels**	Input	An array of pointers to character string labels to be placed on the coneheads. If no labels are desired this may be passed as NULL. Individual entries may also be set to NULL. Contains num entries.
UF_DISP_conehead_attrb_s *	**attributes**	Input	An array of conehead attribute structures If this is set to NULL, the current conehead attributes are used for all the coneheads. For a description of this structure and how to set the current conehead attributes see UF_DISP_set_conehead_attrb. Contains num entries.
double	**selection_point**	Input	Determines where the selection points are on the coneheads. This is given as a fraction of the total conehead length, 0.0 to 1.0.
int	**display_coneheads**	Input	Which coneheads will remain displayed on **Return** UF_UI_DISP_NONE UF_UI_DISP_SELECTED UF_UI_DISP_ALL
int *	**selected_num**	Output	The conehead that was selected. 1 to num. Only valid if response = UF_UI_OK.
int *	**response**	Output	Indication of the users response: UF_UI_OK UF_UI_BACK UF_UI_CANCEL

UF_UI_select_feature (view source)

Defined in: uf_ui.h

Overview

Selects multiple features. Presents a list box of features in the work part and allows multiple selection of features from the list box or from

the graphical display. If there are no features to be presented, an empty list box is presented After the user okays the selection, the features are unhighlighted. There must be a part loaded when this function is called.

Environment
Internal

Note: If filter is NULL, boolean and UDF features will not be presented. For all booleans to be presented use UF_UI_feat_sel_type_t

Required License(s)
gateway

```
int UF_UI_select_feature
(
    char * message,
    void * filter,
    int * count,
    tag_t * * feature_tags,
    int * response
)
```

char *	**message**	Input	Cue line message to display.
void *	**filter**	Input	Must be NULL or castable to UF_UI_feat_sel_type_t.
int *	**count**	Output	Count features selected
tag_t * *	**feature_tags**	Output to UF_*free*	Allocated array of selected feature tags. After use must be freed with UF_free
int *	**response**	Output	response: UF_UI_BACK UF_UI_CANCEL UF_UI_OK

UF_UI_select_parameters (view source)

Defined in: uf_ui.h

Overview
Displays a list box of the expressions corresponding to the parameters of the input feature. You can select multiple expressions from the list box. Returns an allocated array of expression tags. The expression tag array must be freed with UF_free after use. There must be a part loaded when this function is called. The input feature must be in the work part.

Environment
Internal

Required License(s)
gateway

```
int UF_UI_select_parameters
(
    char* message,
    tag_t feature_tag,
    int* count,
    tag_t* * exp_tags,
    int* response
)
```

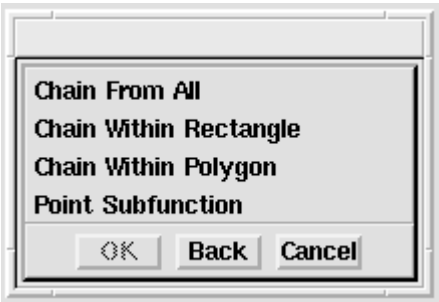
char*	**message**	Input	Cue line message.
tag_t	**feature_tag**	Input	Feature tag.
int*	**count**	Output	Number of feature parameters selected
tag_t* *	**exp_tags**	Output to UF_*free*	The allocated tag array of the corresponding expressions of the selected parameters. This array must be freed by calling UF_free.
int*	**response**	Output	Response UF_UI_BACK (Back) UF_UI_CANCEL (Cancel) UF_UI_OK (Parameters selected)

UF_UI_select_point_collection (view source)

Defined in: uf_ui.h

Overview

Allows you to select a collection of points using the Point Specification menu (see the NX Gateway manual for details)



Point Specification Method Menu

The logical value determines if coincident points are returned. This function returns a count of points and an array of structures. You must free memory using UF_free.
NOTE: object is null_tag if you use the point subfunction. pt[3] is the location in absolute space.

Environment

Internal

See Also

UF_UI_chained_points_p_t

Required License(s)

gateway

```
int UF_UI_select_point_collection
(
    char * message,
    logical coincident_points,
    UF_UI_chained_points_p_t * points,
    int * count,
    int * response
)
```

char *	**message**	Input	Displays a message (32 character maximum) in the cue line. The message is suffixed with the string "Point Specification Method".
logical	**coincident_points**	Input	TRUE = coincident points can be returned. FALSE = do not return coincident points. not valid with point constructor
UF_UI_chained_points_p_t *	**points**	Output to UF_*free*	Array of data structures containing the points. This array must be freed by calling UF_free.
int *	**count**	Output	Count of points returned.
int *	**response**	Output	A response of Back, Cancel , or OK is specified by one of the following constants: UF_UI_BACK UF_UI_CANCEL UF_UI_OK

UF_UI_select_routing_objects (view source)

Defined in: uf_ui_route.h

Overview

Select Routing objects using the standard Routing selection tool

There must be an active part for this function to be called.

Environment

Internal

History

Original release was in V18.0.

Required License(s)

gateway

```
int UF_UI_select_routing_objects
(
    char * title,
    char * message,
    int types,
    int * method,
    int scope,
    int * response,
    int * count,
    tag_p_t * objects
)
```

char *	**title**	Input	Dialog Title or Null
char *	**message**	Input	Cue line message to display
int	**types**	Input	Selectable Routing types - this should be specified by logically OR-ing the desired types from the list of types defined defined in uf_ui_route.h which have a prefix of UF_UI_ROUTE_SEL.
int *	**method**	Input / Output	Default (starting) Routing selection method. This should be one of the method definitions defined in uf_ui_route.h starting with the prefix: UF_UI_ROUTE_SEL_METHOD This will be set to last selection method used by the interactive user during this invocation. Using this, you can "remember" the user's last method setting for the next call
int	**scope**	Input	Selection scope (include uf_ui.h) UF_UI_SEL_SCOPE_NO_CHANGE UF_UI_SEL_SCOPE_ANY_IN_ASSEMBLY UF_UI_SEL_SCOPE_WORK_PART UF_UI_SEL_SCOPE_WORK_PART_AND_OCC
int *	**response**	Output	UF_UI_BACK UF_UI_CANCEL UF_UI_OK
int *	**count**	Output	Count of objects selected. 0 if no objects selected.
tag_p_t *	**objects**	Output to UF_*free*	Array of object identifiers of the selected objects. This must be freed with UF_free.

UF_UI_select_rpo_dimensions (view source)

Defined in: uf_ui.h

Overview

Displays the rpo dimensions of the specified feature in a list box which shows the corresponding expressions. You can select multiple expressions from the list box or select the dimensions graphically. Returns an allocated array of expression tags. The expression tag array must be freed with UF_free after use. An error is returned if the feature has no rpo dimensions. There must be a part loaded when this

function is called. The feature that is input must be in the work part.

Environment
Internal

Required License(s)
gateway

int UF_UI_select_rpo_dimensions
(
char * message,
tag_t feature_tag,
int * count,
tag_t * * exp_tags,
int * response
)

char *	**message**	Input	Cue line message
tag_t	**feature_tag**	Input	Feature tag
int *	**count**	Output	Number of rpo dimensions selected
tag_t * *	**exp_tags**	Output to UF_*free*	The allocated tag array of the corresponding expressions for the selected rpo dimensions. This must be freed by calling UF_free.
int *	**response**	Output	Response UF_UI_BACK (Back) UF_UI_CANCEL (Cancel) UF_UI_OK (Parameters selected)

UF_UI_select_single (view source)

Defined in: uf_ui.h

Overview
This function is to be obsoleted in the near future. Please use the replacement routine UF_UI_select_with_single_dialog . Selects a single object. The view parameter is a pointer view sequence. This tag provides the view from which the object was selected.

Environment
Internal

See Also
Refer to the example
UF_UI_selection_options_t

Required License(s)
gateway

int UF_UI_select_single
(

```
    char * message,
    UF_UI_selection_options_p_t opts,
    int * response,
    tag_p_t object,
    double cursor [ 3 ] ,
    tag_p_t view
)
```

char *	**message**	Input	Cue line message to display.
UF_UI_selection_options_p_t	**opts**	Input	Selection options. See the Data structures section of this chapter.
int *	**response**	Output	response: 1 = Back 2 = Cancel 4 = Object selected by name 5 = Object selected
tag_p_t	**object**	Output	Object identifier of selected object
double	**cursor [3]**	Output	Cursor position. This is undefined if response is 4 (object selected by name).
tag_p_t	**view**	Output	View of selection. This is NULL_TAG if response is 4 (object selected by name).

UF_UI_select_sketch (view source)

Defined in: uf_ui.h

Overview

Presents a list box of all sketches in the work part and returns the sketch selected by the user from either the list box or the graphical display. After the user okays the selection of the sketch, the sketch is unhighlighted. There must be a part loaded when this function is called. If there are no sketches to be selected an error is returned.

NOTE: This function returns old sketches (pre-V13.0) and new sketches (V13 and beyond). Since new sketches are now features, you can use UF_UI_select_feature to return new sketches. This function also provides a list box.. You can also return sketches with general selection functions like UF_UI_select_with_single_dialog (or UF_UI_select_single) and UF_UI_select_with_class_dialog (or UF_UI_select_by_class).

Environment

Internal

Required License(s)

gateway

```
int UF_UI_select_sketch
(
    char * message,
    void * mask,
```

```
    tag_t * sketch_tag,
    int * response
)
```

char *	**message**	Input	Cue line message
void *	**mask**	Input	Reserved for future. Must be set to NULL.
tag_t *	**sketch_tag**	Output	The tag of the selected sketch.
int *	**response**	Output	Response: UF_UI_BACK (Back) UF_UI_CANCEL (Cancel) UF_UI_OK (Sketch Selected)

UF_UI_select_sketch_dimensions (view source)

Defined in: uf_ui.h

Overview

All the dimensions of the specified sketch are displayed if they are not already displayed. However, only dimension constraints are selectable, i.e., reference dimensions cannot be selected. The user is presented with a list box of the correspondng expressions. You can select multiple expressions from the list box or select the sketch dimensions graphically. There must be a part loaded when this function is called. The sketch must be in the work part.

Returns an allocated array of expression tags. The expression tag array has to be freed with UF_free after use. If the sketch has no dimension constraints, an error is returned.

Environment

Internal

Required License(s)

gateway

```
int UF_UI_select_sketch_dimensions
(
    char* message,
    tag_t sketch_tag,
    int* count,
    tag_t* * exp_tags,
    int* response
)
```

char*	**message**	Input	Cue line message.
tag_t	**sketch_tag**	Input	Sketch tag.
int*	**count**	Output	Number of sketch dimension constraints selected

tag_t* *	**exp_tags**	Output to UF_*free*	The allocated tag array of the corresponding expressions of the selected sketch dimension constraints. Use UF_free to deallocate memory after use.
int*	**response**	Output	Response: UF_UI_BACK (Back) UF_UI_CANCEL (Cancel) UF_UI_OK (Dimensions selected)

UF_UI_select_tc_result_file_to_import (view source)

Defined in: uf_ui.h

Overview

Create a Result File Selection Box dialog and allows the user to select a named reference file from Teamcenter CAESolution dataset and passes back the selected Teamcenter file location.

Return

0 = No error
!= 0 = Error code

Environment

Internal

Required License(s)

gateway

```
int UF_UI_select_tc_result_file_to_import
(
    char * * file_extensions,
    int num_extensions,
    char filename [ MAX_FSPEC_BUFSIZE ] ,
    int * response
)
```

char * *	**file_extensions**	Input	Caller has to supply the list of file extension strings. The elements of the list will decide which type of the files need to be displayed in the selection dialog box for selection. The string extension of the result file which needs to be displayed should in the format ".XXX". E.g ".prt", ".sim", ".dat" etc.
int	**num_extensions**	Input	The variable will provide the size of the file extensions list.
char	**filename [MAX_FSPEC_BUFSIZE]**	Output	Caller has to select the desired result file from the selection dialog to enable the OK button to finish the selection operation. All the filtered result files of the part under investigation will be desplayed for selection.

			It is caller's responsibility to allocate enough memory for the filename string.
int *	**response**	Output	UF_UI_OK: OK was selected UF_UI_CANCEL: CANCEL was selected

---

UF_UI_select_with_class_dialog (view source)

Defined in: uf_ui.h

Overview

Select multiple objects with the class selection dialog.

If the response is UF_UI_OK, the selected objects remain highlighted. The response may be UF_UI_OK but no objects have been selected. If the response is UF_UI_CANCEL, all the selected objects are unhighlighted.

The valid selection scopes are defined in uf_ui.h. If the selection scope is changed, it is restored to its original state when the dialog is terminated.

The selection initialization procedure is an optional procedure provided by the user to specify additional selection parameters by calling other UF_UI selection functions. For more information, see UF_UI_select_with_single_dialog.

In the selection initialization procedure UF_UI_set_sel_mask can be called to specify object type filtering. The default object type mask is all standard types selectable. UF_UI_set_sel_procs can be called to specify a filter procedure and/or selection callback.

To begin with objects already selected (which allows them to be deselected), call UF_UI_add_to_sel_list from the selection initialization procedure.

There must be an active part for this function to be called.

Note: In NX5 class selection was converted to Blocked Base Menu. Selection will now inherit any selected objects from global selection. If inheritance is not desired then the global selection should be cleared (deselect the objects) before calling UF_UI_select_with_class_dialog.

The function UF_UI_set_cursor_view is necessary to enable the selection of objects within drawing member views.

Environment

Internal

See Also

UF_UI_select_with_single_dialog

See Also

UF_UI_set_cursor_view
Refer to the example

History

Original release was in V13.0.

Required License(s)

gateway

```
int UF_UI_select_with_class_dialog
(
    char * message,
    char * title,
    int scope,
    UF_UI_sel_init_fn_t sel_init_proc,
    void* user_data,
    int * response,
    int * count,
    tag_p_t* object
)
```

char *	**message**	Input	Cue line message to display
char *	**title**	Input	Dialog Title or Null
int	**scope**	Input	Selection scope UF_UI_SEL_SCOPE_NO_CHANGE UF_UI_SEL_SCOPE_ANY_IN_ASSEMBLY UF_UI_SEL_SCOPE_WORK_PART UF_UI_SEL_SCOPE_WORK_PART_AND_OCC
UF_UI_sel_init_fn_t	**sel_init_proc**	Input	Selection initialization procedure or NULL
void*	**user_data**	Input	User data for initialization procedure or NULL
int *	**response**	Output	UF_UI_CANCEL UF_UI_OK
int *	**count**	Output	Count of objects selected. 0 if no objects selected.
tag_p_t*	**object**	Output to UF_*free*	Array of object identifiers of the selected objects. This must be freed with UF_free.

UF_UI_select_with_single_dialog (view source)

Defined in: uf_ui.h

Overview

Selects a single object with the single selection dialog. The object can be selected with the cursor or by entering a name. The object is highlighted.

The valid selection scopes are defined in uf_ui.h (e.g UF_UI_SEL_SCOPE_NO_CHANGE). If the selection scope is changed, it is restored to its original state when the dialog is terminated.

The selection initialization procedure is a function the Open C API programmer can optionally provide in order to customize their selection by calling other UF_UI selection functions. NX calls the selection initialization procedure, passing the selection pointer and user data. The selection pointer is only valid during the selection

initialization procedure, and is used as an input argument to the UF_UI selection functions.

If the initialization is successful, the procedure should return UF_UI_SEL_SUCCESS. Otherwise, it should return UF_UI_SEL_FAILURE. In this case, the single selection dialog is presented and an appropriate error code is returned. In the selection initialization procedure:

UF_UI_set_sel_mask can be called to specify object type filtering. The default object type mask is all standard types selectable.

UF_UI_set_sel_procs can be called to specify a filter procedure and/or selection callback.

You can call UF_UI_select_with_single_dialog in a loop to select multiple objects. The user indicates they are done selecting by choosing OK. You must have an active part to call this function.

The function UF_UI_set_cursor_view is necessary to enable the selection of objects within drawing member views.

Environment
Internal

See Also
UF_UI_set_cursor_view

See Also
Refer to the example

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_UI_select_with_single_dialog
(
    char * message,
    char * title,
    int scope,
    UF_UI_sel_init_fn_t init_proc,
    void* user_data,
    int * response,
    tag_t * object,
    double cursor [ 3 ] ,
    tag_t * view
)
```

char *	**message**	Input	Cue line message to display
char *	**title**	Input	Dialog title or NULL
int	**scope**	Input	Selection scope UF_UI_SEL_SCOPE_NO_CHANGE UF_UI_SEL_SCOPE_ANY_IN_ASSEMBLY UF_UI_SEL_SCOPE_WORK_PART UF_UI_SEL_SCOPE_WORK_PART_AND_OCC
UF_UI_sel_init_fn_t	**init_proc**	Input	Selection initialization procedure or NULL

void*	**user_data**	Input	User data for initialization procedure or NULL
int *	**response**	Output	UF_UI_BACK UF_UI_CANCEL UF_UI_OK UF_UI_OBJECT_SELECTED UF_UI_OBJECT_SELECTED_BY_NAME
tag_t *	**object**	Output	Selected object or NULL_TAG if no object selected
double	**cursor [3]**	Output	Absolute coordinates of cursor position. This is undefined if object is selected by name.
tag_t *	**view**	Output	View object was selected in. This is NULL_TAG if object was selected by name.

UF_UI_set_cursor_view (view source)

Defined in: uf_ui.h

Overview

Sets the current mask for selection within views. Object selection can be made in the work view on a drawing layout only or any view, and this routine sets the current value.

In drafting, the default is the work view so use this function if you want to select in any view. It is recommended to call UF_UI_ask_cursor_view and save the current value so you can set it back after changing.

Environment

Internal

See Also

UF_UI_ask_cursor_view

Required License(s)

gateway

```
int UF_UI_set_cursor_view
(
    int new_cursor_view
)
```

int	**new_cursor_view**	Input	New cursor view: 0 = Any view 1 = Work view

---

UF_UI_set_dialog_directory (view source)

Defined in: uf_ui.h

Overview
Sets the directory path of a given dialog.

Environment
Internal

See Also
UF_UI_dialog_dir_id_t

Required License(s)
gateway

```
int UF_UI_set_dialog_directory
(
    UF_UI_dialog_dir_id_t id,
    char * dir_name
)
```

UF_UI_dialog_dir_id_t	**id**	Input	Enumeration constant of the dialog.
char *	**dir_name**	Input	Directory path of the dialog.

UF_UI_set_dialog_filter (view source)

Defined in: uf_ui.h

Overview
Sets the filter extension of the current dialog.

Environment
Internal

See Also
UF_UI_dialog_dir_id_t

Required License(s)
gateway

```
int UF_UI_set_dialog_filter
(
    UF_UI_dialog_filter_id_t id,
    char * fltr_name
)
```

UF_UI_dialog_filter_id_t	**id**	Input	Enumeration constant of the dialog.
char *	**fltr_name**	Input	Filter extension of the dialog.

UF_UI_set_force_unlock_flag (view source)

Defined in: uf_ui.h

Overview
Sets the unlock flag to unlock NX functions regardless of the NX state.

The call to UF_UI_set_force_unlock flag must be followed with a call to UF_UI_unlock_ug_access.

For a complete example demonstrating how to use this function please refer to the example provided in the Open kit called ufx_menuscript_ufsta.c.

Return
return code:
UF_UI_SUCCESS when the unlock flag is set.
UF_UI_FAILURE when the unlock flag is not set.

Environment
Internal

Required License(s)
gateway

```
int UF_UI_set_force_unlock_flag
(
    void
)
```

UF_UI_set_minimal_graphics_window (view source)

Defined in: uf_ui.h

Overview
This function will turn on/off NX graphics window special minimal mode

Environment
Privileged

History
Added in NX7.5.2

Required License(s)
gateway

```
int UF_UI_set_minimal_graphics_window
(
    logical set
)
```

logical	**set**	Input

UF_UI_set_minimal_graphics_window_location (view source)

Defined in: uf_ui.h

Overview
This function will set NX graphics window position in special minimal mode
If all parameters are zero then NX will be resized to fill whole current screen.

Environment
Privileged

History
Added in NX7.5.2

Required License(s)
gateway

```
int UF_UI_set_minimal_graphics_window_location
(
    int left,
    int top,
    int right,
    int bottom
)
```

int	**left**	Input	x-value at left side of graphics window
int	**top**	Input	y-value at top of graphics window
int	**right**	Input	x-value at right side of graphics window
int	**bottom**	Input	y-value at bottom of graphics window

UF_UI_set_prompt (view source)

Defined in: uf_ui.h

Overview
Displays a line of text in the NX prompt area. You call this routine after some interactive dialog invokes a change in the status bar. You can call this routine to prompt the user when a specific interactive task is required. For example, if your program requires the user to select a line end point, you could prompt the user to "Pick line end pt1".

Environment
Internal

Required License(s)
gateway

```
int UF_UI_set_prompt
(
    char * prompt_text
)
```

char *	**prompt_text**	Input	Pointer to text to display.

UF_UI_set_ribbon_vis (view source)

Defined in: uf_ui.h

Overview

This routine sets the visibility of a ribbon tab. This routine can only be used on a ribbon tab for which you have a valid ribbon id. This function only applies to ribbon tabs; for groups, galleries, cascades, and dropdowns it has no effect.

Note: Changing the visibility of the ribbon tab will override any user customization of the tabs visibility. It is preferable to use UF_UI_create_ribbon and UF_UI_remove_ribbon, which also works on groups, galleries, dropdowns and cascades.

Environment

Internal

See Also

UF_UI_ask_ribbon_vis
UF_UI_create_ribbon

History

Originally released in NX9.0

Required License(s)

gateway

```
int UF_UI_set_ribbon_vis
(
    UF_UI_ribbon_id_t ribbonl_id,
    int show
)
```

UF_UI_ribbon_id_t	**ribbonl_id**	Input	Valid Ribbon id returned from a call to UF_UI_create_ribbon
int	**show**	Input	1 = show; 0 = hide

UF_UI_set_sel_mask (view source)

Defined in: uf_ui.h

Overview

Sets the types of objects that are selectable. If this function is not called to set the mask, then the default mask is all standard types. The following is the current list of standard types.

UF_point_type
UF_line_type
UF_circle_type
UF_conic_type
UF_spline_type
UF_pattern_type
UF_kanji_type
UF_group_type
UF_drafting_entity_type
UF_dimension_type
UF_tabular_note_type
UF_margin_type
UF_coordinate_system_type
UF_plane_type
UF_component_type
UF_datum_axis_type
UF_datum_plane_type
UF_facet_topology_type
UF_view_type
UF_view_set_type
UF_route_control_point_type
UF_route_port_type
UF_route_segment_type
UF_route_part_anchor_type
UF_route_stock_type
UF_analysis_type
UF_traceline_type
UF_constraint_type
UF_solid_type

In this list of standard types, UF_solid_type only specifies solid bodies and does not include faces or edges. To select faces or edges, you must specify these object subtypes in mask_triples.

The word ALL in the two symbols UF_UI_SEL_MASK_ENABLE_ALL and UF_UI_SEL_MASK_ALL_AND_DISABLE_SPECIFIC means all standard types.

Use this function for setting the object type mask for dialogs created with the UIStyler. The function can be called from any UIStyler dialog callback or the selection callback to change the object type mask.

For UF_UI_select_with_single_dialog and UF_UI_select_with_class_dialog, this function can only be called from the selection initialization procedure. Calling this function from a selection filter procedure returns an error.

Environment
Internal

See Also
UF_UI_mask_t

History
Original release was in V13.0.

Required License(s)
gateway

int UF_UI_set_sel_mask
(
UF_UI_selection_p_t select_,
UF_UI_sel_mask_action_t action,

```
    int num,
    UF_UI_mask_t * mask_triples
)
```

UF_UI_selection_p_t	**select_**	Input	selection pointer
UF_UI_sel_mask_action_t	**action**	Input	Mask action UF_UI_SEL_MASK_ENABLE_ALL UF_UI_SEL_MASK_ENABLE_SPECIFIC UF_UI_SEL_MASK_DISABLE_SPECIFIC UF_UI_SEL_MASK_CLEAR_AND_ENABLE_SPECIFIC UF_UI_SEL_MASK_ALL_AND_DISABLE_SPECIFIC
int	**num**	Input	Number of mask triples
UF_UI_mask_t *	**mask_triples**	Input	Array of mask triples.

UF_UI_set_sel_procs (view source)

Defined in: uf_ui.h

Overview

Specify selection filter procedure for additional filtering based on application specific criteria, and/or selection callback to perform application specific processing for each selection gesture processed, and user specific data to be passed to these routines.

This function can be used with UIStyler dialogs and with UF_UI_select_with_single_dialog and UF_UI_select_with_class_dialog. The typedef of the filter procedure and selection callback are defined in this header file as follows:

typedef int (UF_UI_sel_filter_fn_t) (
tag_t object,
int type[3],
void user_data,
UF_UI_selection_p_t select_)

typedef int (UF_UI_sel_cb_fn_t) (
int num_selected,
tag_p_t selected_objects,
int num_deselected,
tag_p_t deselected_objects,
void user_data,
UF_UI_selection_p_t select_)

The filter procedure is passed the tag of the candidate object, the types of the candidate object (object type, object subtype, solid subtype - see UF_UI_set_sel_mask) , a pointer to the user's data, and a pointer to selection. The return responses are defined in uf_ui.h. The filter procedure should return UF_UI_SEL_REJECT if the object is to be discarded and UF_UI_SEL_ACCEPT if the object is a valid candidate.

The candidate object passed to the filter procedure is not adjusted for scope. Therefore, with an assembly part, the object is the occurrence and not the prototype.

You can get the prototype of the object by calling

UF_ASSEM_ask_prototype_of_occ. However, if the object is to be a promotion, for example, then the user has to do some inquiries as in the example below:

```
if (UF_ASSEM_is_occurrence(object))
{
proto = UF_ASSEM_ask_prototype_of_occ(object);
UF_MODL_ask_prom_feat_of_solid(proto, &feat));
UF_MODL_prom_map_object_up(proto, feat,
&prom));
status = check_promotion(prom);
}
```

The selection callback is different. The objects passed to it are already adjusted for scope.

The selection callback is passed an allocated array of the objects selected or deselected with the previous selection. The allocated array of tags will be freed for the user. The user can force dialog termination by returning UF_UI_CB_EXIT_DIALOG. To continue the dialog, the user should return UF_UI_CB_CONTINUE_DIALOG.

When a selection callback is used with UF_UI_select_with_single_dialog, the return is ignored and the dialog is always terminated.

Both the filter procedure and the selection callback are passed a pointer to selection which can be used as input to other UF_UI selection functions to inquire other selection data or modify selection. This selection pointer is no longer valid after the filter procedure or selection callback is exited.

Environment
Internal

See Also
Refer to the example

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_UI_set_sel_procs
(
    UF_UI_selection_p_t select_,
    UF_UI_sel_filter_fn_t filter_proc,
    UF_UI_sel_cb_fn_t sel_cb,
    void * user_data
)
```

UF_UI_selection_p_t	**select_**	Input	selection pointer
UF_UI_sel_filter_fn_t	**filter_proc**	Input	filter procedure for additional user specific filtering or NULL
UF_UI_sel_cb_fn_t	**sel_cb**	Input	selection callback for application specific processing or NULL
void *	**user_data**	Input	User data or NULL

UF_UI_set_sel_type (view source)

Defined in: uf_ui.h

Overview

Sets the selection type.

You use this function with UIStyler dialogs. It can be called from a callback to change the type of selection associated with the dialog. Valid types are defined in uf_ui.h.

A type of UF_UI_SEL_TYPE_INACTIVE_SELECTION sets selection inactive until this function is called again. UF_UI_SEL_TYPE_ROBUST_SELECTION allows single select, single deselect, reselect last, rectangle select, rectangle deselect, and chaining. If UF_UI_SEL_TYPE_SINGLE_POSITION and either UF_UI_SEL_TYPE_SINGLE_SELECTION or UF_UI_SEL_TYPE_ROBUST_SELECTION is requested, the position is returned if no object is selected with the single select gesture.

Calling this function for UF_UI_select_with_single_dialog or UF_UI_select_with_class_dialog returns an error. Calling this function from a selection filter procedure returns an error.

Environment

Internal

History

Original release was in V13.0.

Required License(s)

gateway

```
int UF_UI_set_sel_type
(
    UF_UI_selection_p_t select_,
    int type
)
```

UF_UI_selection_p_t	**select_**	Input	selection pointer
int	**type**	Input	UF_UI_SEL_TYPE_INACTIVE_SELECTION or bit mask of selection types: UF_UI_SEL_TYPE_SINGLE_SELECTION UF_UI_SEL_TYPE_SINGLE_DESELECTION UF_UI_SEL_TYPE_ROBUST_SELECTION UF_UI_SEL_TYPE_SINGLE_POSITION UF_UI_SEL_TYPE_RECTANGLE_POSITION

UF_UI_set_select_mask (view source)

Defined in: uf_ui.h

Overview

Determines which object types to mask.

This function modifies a shared filter only used by a few NX commands and does not modify the global selection filter. To establish a filter to use for selection, see function UF_UI_set_sel_mask(), which can be used to establish a filter for a UIStyler dialog, or the dialog launched by either UF_UI_select_with_class_dialog or UF_UI_select_with_single_dialog; both of these functions take in a proc of type UF_UI_sel_init_fn_t. In your implementation of that proc, you can set up the selection filter for the dialog using UF_UI_set_sel_mask().

Environment

Internal

See Also

UF_UI_set_sel_maks
UF_UI_select_with_class_dialog
UF_UI_select_with_single_dialog

Required License(s)

gateway

int UF_UI_set_select_mask
(
int action,
int num_items,
int * items_to_mask
)

int	**action**	Input	Mask action: UF_UI_enable_all UF_UI_disable_all UF_UI_enable_specific UF_UI_disable_specific
int	**num_items**	Input	Number of object types in array
int *	**items_to_mask**	Input	Object types to be masked

UF_UI_set_status (view source)

Defined in: uf_ui.h

Overview

Displays a line of text in the NX status area.

Environment

Internal

Required License(s)

gateway

int UF_UI_set_status
(

```
    char * status_text
)
```

char *	**status_text**	Input	Pointer to text to display

UF_UI_set_toolbar_vis (view source)

Defined in: uf_ui.h

Overview

This routine sets the visibility of a toolbar. This routine can only be used on toolbars that you have valid toolbar id for.

Environment

Internal

See Also

UF_UI_ask_toolbar_vis
UF_UI_create_toolbar

History

Originally released in V16.0

Required License(s)

gateway

```
int UF_UI_set_toolbar_vis
(
    UF_UI_toolbar_id_t tool_id,
    int show
)
```

UF_UI_toolbar_id_t	**tool_id**	Input	Valid Toolbar id returned from a call to UF_UI_create_toolbar
int	**show**	Input	1 = show; 0 = hide

UF_UI_set_usertool_menu_entry (view source)

Defined in: uf_ui.h

Overview

Replaces the user tool definition file specified by a user tools menubar option. The option number range starts at one and its maximum is defined by the original length of the ".utm" file. For example, if the ".utm" file only includes four entries, it is not possible to extend the size of the menubar entries by setting the option number to 5. The replacement for the user tool definition does not take effect until Reload-->Default is selected. Therefore, hiding and then showing the tool does not change the definition.

Environment
Internal

Required License(s)
gateway

```
int UF_UI_set_usertool_menu_entry
(
    int option_number,
    char * label,
    char * filename
)
```

int	**option_number**	Input	Menu item number to be replaced.
char *	**label**	Input	Pointer to label text
char *	**filename**	Input	Pointer to new tool definition filename

---

UF_UI_specify_csys (view source)

Defined in: uf_ui.h

Overview
This function launches a dialog to let the user choose a csys. The csys specified by the input csys_tag is the default. If the csys_tag input is NULL_TAG, then the initial CSYS is used as the default. The output csys_tag is the csys the user specified in the dialog.

Though 40 characters are allowed in the title,
it should be noted that the first option (ORIGIN,X-PT,Y-PT) appends the point number (PT1, PT2, and PT3) on the end of the message. Thus, only 36 characters are effectively allowed.

Return
Users Response
1 = Back
2 = Cancel
3 = OK
7 = No Active Part
8 = Disallowed state, unable to bring up dialog

Environment
Internal

Required License(s)
gateway

```
int UF_UI_specify_csys
(
    char * title,
    int * option,
    double csys_matrix [ 9 ] ,
    double origin [ 3 ] ,
    tag_t * csys_tag
```

)

char *	**title**	Input	Menu Title (40 char max)
int *	**option**	Input / Output	On input the default CSYS Option. On output the CSYS option used. 0 = Inferred 1 = Origin, X-pt, Y-pt 2 = X-axis, Y-axis 3 = X-pt, Z-axis 4 = CSYS of Object 5 = WCS (Dynamic) 6 = Offset CSYS 7 = Absolute CSYS 8 = CSYS of Current View 9 = Three Planes 10 = Origin, X-axis, Y-axis 11 = Point, Perpendicular Curve 12 = Plane, Vector 13 = Plane, X-Axis, Point 14 = Origin, Z-Axis, X-Axis 15 = Origin, Z-Axis, Y-Axis
double	**csys_matrix [9]**	Output	CSYS Matrix (9 element array) - relative to Absolute CSYS of Displayed Part
double	**origin [3]**	Output	CSYS Origin (3 element array) - relative to Absolute CSYS of Displayed Part
tag_t *	**csys_tag**	Input / Output	CSYS tag - On input the default CSYS. On output the CSYS picked. If csys_tag is not NULL, "option" will reflect the type of the CSYS If csys_tag is a non-associative CSYS, option will be set to Dynamic

---

UF_UI_specify_plane (view source)

Defined in: uf_ui.h

Overview

Allows you to select the plane subfunction default mode. A temporary plane is created from the specified subfunction.

Environment

Internal

See Also

Refer to the example

Required License(s)

gateway

```
int UF_UI_specify_plane
(
    char * message,
    int * mode,
    int display,
```

```
    int * response,
    double orientation [ 9 ] ,
    double origin [ 3 ] ,
    tag_t * plane_eid
)
```

char *	**message**	Input	Cue line message (80 character maximum)
int *	**mode**	Input / Output	On input default plane subfunction. On output the plane subfunction actually used: -1 = Enable ok without default selection 0 = Inhibit default selection 1 = Three Points 2 = Two Lines 3 = Point, Perp Curve 4 = Plane of Arc/Conic 5 = Plane of WCS 6 = Plane of CSYS 7 = Principal Plane 8 = Existing Plane 9 = Two Tangent Faces 10 = Point, Tangent Face 11 = Coefficients 12 = Parallel Thru Pt 13 = Parallel At Dist 14 = Perp, Thru Line
int	**display**	Input	0 = Display temporary plane in all active views 1 = Do not display temporary plane
int *	**response**	Output	1 = Back 2 = Cancel 3 = Ok
double	**orientation [9]**	Input	Plane orientation in absolute coordinate
double	**origin [3]**	Input	Plane origin in absolute coordinate
tag_t *	**plane_eid**	Output	Object identifier of plane if mode 8 was selected NULL_TAG for other modes.

UF_UI_specify_screen_position (view source)

Defined in: uf_ui.h

Overview

This function allows you to indicate a screen position by pressing MB1 in the graphics window. The screen position and the tag of the view it is in are returned.

An empty dialog is displayed with only the Back and Cancel buttons enabled.

The function UF_UI_set_cursor_view affects the screen_pos and view_tag that is returned and passed to the motion callback. This is particularly true with respect to the display of a drawing view. The two values for the new_cursor_view parameter of UF_UI_set_cursor_view affect the view_tag and screen_pos parameters of UF_UI_specify_screen_position as follows:

. If new_cursor_view is set to "Any View" and the cursor is in a drawing member view, then the return values for the view_tag and screen_pos are the tag of the member view and the position in Absolute Coordinates in that member view.

. If the new_cursor_view is set to "Work View", then regardless of whether the cursor is in a member view or not, the return values for the view_tag and screen_pos are the tag of the drawing and the position in drawing coordinates.

If Grid Snap is presently enabled, the screen position is automatically snapped. This also applies to the position passed to the motion callback.

This function accepts a motion callback which will be called in response to each detected movement (i.e. "motion") of the cursor within the graphics window. The callback will be passed the current position and view of the cursor, and the client data pointer.
The typedef for motion callbacks is defined in uf_disp.h as follows:

```
typedef void (UF_UI_motion_fn_t)(
double screen_pos[3],
UF_UI_motion_cb_data_p_t motion_cb_data,
void data );
```

All of the above parameters are input parameters to the callback function:

screen_pos is the current position of the crosshair, given in Work Part Absolute Coordinates (as described above).

motion_cb_data is a pointer to a data structure; presently, only the following field of this structure should be referenced:
motion_cb_data->view_tag is the tag of the view of the current crosshair position.

Finally, the third parameter to the motion callback, data, is the client data pointer initially passed to UF_UI_specify_screen_position along with the callback.

In general, a motion callback will generate some graphical feedback based on the current cursor position, using Overlay Graphics primitives. Overlay Graphics primitives are defined using the UF_DISP_display_ogp_ functions. Overlay Graphics primitives generated from a motion callback will be displayed immediately following the invocation of the callback, and will be automatically erased just before the next invocation, and upon the completion of the call to UF_UI_specify_screen_position.

Please see the Overview of the section on Overlay Graphics primitive functions in the Display chapter for further information regarding their behavior and usage.

Keep in mind that your motion callback will be invoked in response to every detected movement of the cursor (in the graphics window). If you find that the display of the cursor appears to be "choppy", or that it doesn't seem to be "keeping up" with your movement of the mouse, it may be that you are attempting to do too many calculations and/or define too many primitives from your motion callback.

There must be a part loaded when this function is called.

Environment

Internal Only

See Also
UF_UI_set_cursor_view
Refer to the example

Required License(s)
gateway

```
int UF_UI_specify_screen_position
(
    char * message,
    UF_UI_motion_fn_t motion_cb,
    void * motion_cb_data,
    double screen_pos [ 3 ] ,
    tag_t * view_tag,
    int * response
)
```

char *	**message**	Input	Cue line message (132 Character maximum), or NULL
UF_UI_motion_fn_t	**motion_cb**	Input	Motion callback function, or NULL
void *	**motion_cb_data**	Input	Client data pointer, or NULL; will be passed to motion_cb
double	**screen_pos [3]**	Output	The screen position in Work Part Absolute Coords, projected "through the screen" onto the WCS XY plane. This is given in Work Part Absolute Coordinates. This is only returned if the response returned is UF_UI_PICK_RESPONSE.
tag_t *	**view_tag**	Output	Tag of the view in which the screen position was indicated. This is only returned if the response returned is UF_UI_PICK_RESPONSE.
int *	**response**	Output	One of the following: UF_UI_PICK_RESPONSE UF_UI_BACK UF_UI_CANCEL

UF_UI_specify_vector (view source)

Defined in: uf_ui.h

Overview
Queries the user to specify a vector using the vector subfunction.
Optionally, a temporary conehead is displayed at the vector specified.

Environment
Internal

See Also
Refer to the example

Required License(s)

gateway

```
int UF_UI_specify_vector
(
    char * message,
    int * mode,
    int display_conehead,
    double direction [ 3 ] ,
    double origin [ 3 ] ,
    int * response
)
```

char *	**message**	Input	Cue line message (80 character maximum)
int *	**mode**	Input / Output	On input default vector subfunction mode. On output the subfunction mode actually used: UF_UI_INFERRED UF_UI_TWO_POINTS UF_UI_AT_ANGLE UF_UI_EDGE_CURVE UF_UI_TANGENT_TO_CURVE UF_UI_FACE_NORMAL UF_UI_DATUM_PLANE UF_UI_DATUM_AXIS UF_UI_XC_AXIS UF_UI_NEGATIVE_XC_AXIS UF_UI_YC_AXIS UF_UI_NEGATIVE_YC_AXIS UF_UI_ZC_AXIS UF_UI_NEGATIVE_ZC_AXIS
int	**display_conehead**	Input	Display temporary vector in all active views: UF_UI_DISP_TEMP_VECTOR UF_UI_DISP_NO_VECTOR
double	**direction [3]**	Output	Vector direction (unitized)
double	**origin [3]**	Output	Vector origin in absolute coordinate
int *	**response**	Output	User response: UF_UI_OK UF_UI_BACK UF_UI_CANCEL

UF_UI_suspend_create_toolbar (view source)

Defined in: uf_ui.h

Overview

This routine must be used to wrapper the creation of multiple toolbars.
The use of this function helps with the positioning of the toolbars when they are docked.

Example:

```
UF_UI_suspend_create_toolbar();
UF_UI_create_toolbar("file1.tbr",1, &id1);
UF_UI_create_toolbar("file2.tbr",1, &id2);
```

```
UF_UI_create_toolbar("file3.tbr",1, &id3);
UF_UI_resume_create_toolbar();
```

Environment
Internal

See Also
UF_UI_resume_create_toolbar

History
Originally released in V16.0

Required License(s)
gateway

```
int UF_UI_suspend_create_toolbar
(
    void
)
```

UF_UI_suspend_init_appstate (view source)

Defined in: uf_ui.h

Overview
This routine must be used to wrapper the creation of multiple toolbars. The use of this function helps with the positioning of the toolbars when they are docked. It restores toolbar state for current application.

Example:

```
UF_UI_suspend_init_appstate();
UF_UI_create_toolbar("file1.tbr",1, &id1);
UF_UI_create_toolbar("file2.tbr",1, &id2);
UF_UI_create_toolbar("file3.tbr",1, &id3);
UF_UI_resume_init_appstate();
```

Environment
Internal

See Also
UF_UI_resume_init_appstate

History
Originally released in V18.0

Required License(s)
gateway

```
int UF_UI_suspend_init_appstate
(
    void
)
```

UF_UI_suspend_remove_toolbar (view source)

Defined in: uf_ui.h

Overview

This routine must be used to wrapper the removing of multiple toolbars. The use of this function helps with the correct recording of the positions of the docked toolbars in the registry.

Example:

```
UF_UI_suspend_remove_toolbar();
if (id1) UF_UI_remove_toolbar(id1);
if (id2) UF_UI_remove_toolbar(id2);
if (id3) UF_UI_remove_toolbar(id3);
UF_UI_resume_remove_toolbar();

id1 = NULL;
id2 = NULL;
id3 = NULL;
```

Environment

Internal

See Also

UF_UI_resume_remove_toolbar

History

Originally released in V16.0

Required License(s)

gateway

```
int UF_UI_suspend_remove_toolbar
(
    void
)
```

UF_UI_toggle_stoplight (view source)

Defined in: uf_ui.h

Overview

Changes the status of the NX stoplight indicator to either busy or not busy.

Environment

Internal

Required License(s)

gateway

```
int UF_UI_toggle_stoplight
(
    int toggle_on_off
)
```

int	**toggle_on_off**	Input	Flag to toggle stoplight. 1 = the stoplight indicates busy (red) 0 = the stoplight indicates not busy (green)

UF_UI_ugmgr_ask_create_part_file_name (view source)

Defined in: uf_ui.h

Overview

This routine just brings up the part selection dialog to prompt for the new part name. This routine does not actually create the part. This routine was written to provide the routine that does the same work which UF_UI_ask_create_part_filename does in native NX. In NX Manager, UF_UI_ask_create_part_filename creates the specified part, this routine can be used when you want to get the filename but not yet create the part.

See Also

UF_UI_ask_create_part_filename

Environment

Internal

History

Originally released in NX 5.0 and is mandatory if Longer IDs functionality is enabled NX/Manager

Required License(s)

gateway

```
int UF_UI_ugmgr_ask_create_part_file_name
(
    char * * filename,
    char * * part_type,
    char * * template_name,
    int * response
)
```

char * *	**filename**	Output to UF_*free*	The new part name in CLI format
char * *	**part_type**	Output to UF_*free*	The part type of the part
char * *	**template_name**	Output to UF_*free*	The seed part name
int *	**response**	Output	The response from the user

UF_UI_unlock_ug_access (view source)

Defined in: uf_ui.h

Overview
Enables access to NX dialog area 1 and the appropriate menu items after an Interactive Open C API dialog has been used. UF_UI_unlock_ug_access

Returns
UF_UI_NO_LOCK_EXISTED when dialog area 1 and NX menu bar were not locked.

UF_UI_UNLOCK_SET when dialog area 1 and NX menu bar is enabled successfully.

Return
See the return values in the description section.

Environment
Internal

See Also
Refer to the example

Required License(s)
gateway

```
int UF_UI_unlock_ug_access
(
    int from_where
)
```

int	**from_where**	Input	Predefined constant used to support error checking. This should always be UF_UI_FROM_CUSTOM for custom applications.

UF_UI_update_listing_window (view source)

Defined in: uf_ui.h

Overview
This function forces the listing window to redisplay to show all text that has been written to it. Long operations can use this to show progress by writing to the listing window and periodically forcing an update. However the update itself takes time so should only be done when necessary as otherwise performance can suffer.

Environment
Internal

Required License(s)
gateway

```
int UF_UI_update_listing_window
(
    void
)
```

UF_UI_write_listing_window (view source)

Defined in: uf_ui_ugopen.h

Overview

Writes a character string to the Information window. You should add your own new line after the string is placed. If in internal Open API, text is displayed in the Information window. If in external Open API, text is printed to standard out. The Information window must be opened before you can write to it.

Once UF_terminate has completed in an external program, you can no longer write to the console window. It must be reinitialized with system calls.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UI_write_listing_window
(
    const char * string
)
```

const char *	**string**	Input	pointer to character string
