# UF_UI_ask_default_parent_info (view source)

**Defined in: uf_ui_xt.h**

## Overview

When creating a Motif dialog it is required to use the visuals from NX. To do this you must ask for this information to be returned back to you. You must then set your visuals for your widget:

```
Visual visual;
Colormap cmap;
int depth;
.
.
.
UF_UI_ask_default_parent_info (&visual, &cmap, &depth);

n = 0;
XtSetArg( args[n], XmNvisual, visual); n++;
XtSetArg( args[n], XmNcolormap, cmap); n++;
XtSetArg( args[n], XmNdepth, depth); n++;
.
.
.
form = (Widget) XmCreateFormDialog ( parent, "form", args, n );
```

## Environment

Internal

## Required License(s)

gateway

```
int UF_UI_ask_default_parent_info
(
    Visual * * visual,
    Colormap * cmap,
    int * depth
)
```

| | | | |
|---|---|---|---|
| Visual * * | **visual** | Input | The default parent visual. |
| Colormap * | **cmap** | Input | The default parent colormap. |
| int * | **depth** | Input | The default parent depth. |

---

# UF_UI_exit_dialog (view source)

**Defined in: uf_ui_xt.h**

## Overview

Causes the event loop of the currently executing user Motif dialog to exit which, in turn, causes UF_UI_run_dialog() (used to start the dialog) to execute a return. This is only used in relation with UF_UI_run_dialog. Always check the return status of UF_UI_exit_dialog to be sure that exiting the dialog was successful.

If using the UF_UI_cancel_uf_dialog protocol then UF_UI_exit_dialog should be called via a callback from XtAppAddTimeOut, otherwise just call UF_UI_exit_dialog directly. The call to UF_UI_exit_dialog causes the event loop to be terminated which, in turn, causes UF_UI_run_dialog to return to its caller.

### Environment
Internal

### See Also
See the example

### Required License(s)
gateway

**int UF_UI_exit_dialog**
**(**
**void**
**)**

---

## UF_UI_run_dialog (view source)

**Defined in: uf_ui_xt.h**

### Overview
Handles managing and unmanaging of the dialog. The sequence of events is:

Manage the main widget of the dialog (dialog_widget).

Enter the Ungraphics event loop and stay there until UF_UI_exit_dialog is called from a dialog callback function.

When the event loop is exited, the main widget of the dialog is unmanaged.

Return to the caller.

When creating a dialog, you must use the visuals from NX. To do this you must use UF_UI_ask_default_parent_info, to get the information.

After the dialog has been defined (i.e., all widgets created as necessary), the main widget is passed to UF_UI_run_dialog. This function manages the main widget and enters the NX event loop allowing the dialog to "run" (i.e., react to keyboard input and mouse gestures, execute callbacks, etc.). UF_UI_run_dialog does not return to the user until UF_UI_exit_dialog has been called.

When the UF_UI_run_dialog call has been made, some of the NX application becomes inhibited. This includes the non-global functions in the main menubar and various DA1s. Even though access to parts of NX is inhibited, NX does not consider itself in a "lock" state. This only occurs with the call to UF_UI_lock_ug_access. Unlike launching a custom dialog with XtManageChild, with UF_UI_run_dialog you do not need to call UF_UI_lock_ug_access when calling a Presentation API. However, if you do not call UF_UI_lock_ug_access then you are not allowed to call UF_UI_cancel_uf_dialog.

When the dialog handling part of the Open API program (i.e., the dialog's callback functions) determines that it is time to terminate

the dialog (for example, after you press OK or Cancel on the dialog), UF_UI_exit_dialog should be called. This function should only be called from a callback function associated with a dialog initiated by a call to UF_UI_run_dialog.

Always make sure to check the status of UF_UI_run_dialog. This will return an error messages if something goes wrong. You can use the Open API UF_UI_get_fail_message to determine the exact problem UF_UI_run_dialog and UF_UI_exit_dialog experienced.

A user of a custom dialog launched via UF_UI_run_dialog may not launch another Open application that uses the lock & unlock mechanism or is launched via UF_UI_run_dialog.

This example will launch a custom dialog with two push buttons. Prior to launching the custom dialog with UF_UI_run_dialog a call to UF_UI_lock_ug_access is made. This implies the necessary handshaking has been done for the entire session, and therefore is not needed around each and every Open dialog or UIStyler dialog call. Both buttons within the custom dialog always remain active so calls to UF_UI_cancel_uf_dialog have been made to ensure the correct handling of the dialog when the user select another button in the custom dialog while the Open dialog is up. Also, a call to UF_UI_cancel_uf_dialog has been made from within the cancel button navigation callback. Since, this custom dialog is always locked there is no need to check the lock status prior to calling this function. Because no Open API should be called immediately following UF_UI_cancel_uf_dialog, a call to XtAppAddTimeOut is made, which has a callback making the necessary call to UF_UI_exit_dialog. Once the UF_UI_run_dialog has terminated then the unlocking of the custom dialog is done and the call to UF_terminate is made.

## Environment

Internal

## See Also

See the example

## Required License(s)

gateway

**int UF_UI_run_dialog**
**(**
**Widget dialog_widget**
**)**

| | | | |
|---|---|---|---|
| Widget | **dialog_widget** | Input | Main Widget of the Motif dialog. |