OptionsWindow Dialogue box class

Introduction and Overview

An OptionsWindow dialogue object is used to provide a standard, organised manner to allow configuration of the options for an application or document through a dialogue box.

As applications grow in complexity, so their configuration requirements will grow. Many methods for configuring large numbers of options have been used throughout the life of RISC OS, with varying degrees of success. Whilst there will always be applications which have specific requirements for their configuration, it is advantageous to standardise the style of configuration tools on a particular form.

The form which is most functional and which has been standardised upon by a number of applications, best known of which was Browse has been used. This is a single window interface using a number of panes selected through radio buttons.

User interface

A OptionsWindow dialogue takes the following form:

The OptionsWindow Dialogue
The OptionsWindow Dialogue

The user interface provided by the OptionsWindow conforms to that declared in the StyleGuide and used by modern RISC OS applications. The window components perform the following functions :

- Selecting a radio icon causes the currently displayed pane to be hidden and replaced by the appropriate pane.
- The 'Set' button, where present, will cause the current settings from all panes to be configure.
  If Select is used to choose the 'Set' button, the window will be closed.
  If Adjust is used to choose the 'Set' button, the window remains open.

- The 'Cancel' button, where present, will cause the settings shown within the panes to be reset to the current configuration.

If Select is used to choose the 'Cancel' button, the window will be closed.
If Adjust is used to choose the 'Cancel' button, the window remains open.

- The 'Save' button, where present, will cause the current settings from all panes to be configured and the settings written to disc.
  If Select is used to choose the 'Save' button, the window will be closed.
  If Adjust is used to choose the 'Save' button, the window remains open.
- The 'Default' button, where present, will cause the current settings in all panes to be set to the defaults but not configured.
  The operation is the same for Select and Adjust.

- The 'Enter' key will trigger the operation of the button designated as the default button in the options window - the lowest of the buttons.
  The window will be closed.
- The 'Escape' key will trigger the operation of the button designated as the cancel button in the options window - the second lowest of the buttons (Default is never used for this operation). If no button is present, the default operation is selected.
  The window will be closed.
- Help text is provided for all buttons and radio icons.
- Where modifications have been made to the settings, the title bar will indicate this with an asterisk.
- Each pane is considered independant of its siblings. No settings within one pane affect the settings in another pane.

Technical details

The OptionsWindow object consists of a Window controlled by OptionsWindow and a number of pane Window objects, supplied by the application. These panes are referred to as components of the OptionsWindow object, and are dealt with as such in method operations on the object and events returned by the object.

Attributes

A OptionsWindow object has the following attributes which are specified in its object template and can be manipulated at run-time by the client application:

Attributes	Description
flags	flags for this component

Bit(s)	Meaning
0	indicates that the pane Windows will not be monitored for 'Modified' events automatically.
1	indicates that the object has a 'Set' button.
2	indicates that the object has a 'Cancel' button.
3	indicates that the object has a 'Save' button.
4	indicates that the object has a 'Default' button.

title	alternative title to use instead of 'Options' (0 means use default title)
windowlist	this is a comma-separated string which lists the names of the panes which should be automatically attached to the OptionsWindow.

Manipulating a OptionsWindow object

Creating and deleting a OptionsWindow object

An OptionsWindow object is created using SWI Toolbox_CreateObject.

When this object is created it will automatically attach any the objects listed in the Template (see ?TOOLINTRO.HTML#24528).

An OptionsWindow object is deleted using SWI Toolbox_DeleteObject.

The setting of the non-recursive delete bit will prevent the attached panes from being deleted with the OptionsWindow object.

Showing a OptionsWindow object

When an OptionsWindow object is displayed on the screen using SWI Toolbox_ShowObject it has the following behaviour:

Show type	Position
0 (default)	the underlying window is shown centred in the screen.
1 (full spec)	fully specified window position in R3:

Offset	Contents
+0	visible area minimum x coordinate
+4	visible area maximum y coordinate

2 (topleft)	specifying the top left corner in R3 :

Offset	Contents
+0	visible area minimum x coordinate
+4	visible area maximum y coordinate

Manipulating OptionsWindow panes

The application can either supply a list of Window objects which will be automatically attached to the OptionsWindow when it is created, or attach pane Window objects at run time. When automatically attached, the panes will be given ascending component numbers, starting at 0.

The panes used in the OptionsWindow can be controlled through a number of methods provided by the OptionsWindow object:

- *OptionsWindow_AddPane (on page 9)*
- *OptionsWindow_RemovePane (on page 10)*
- *OptionsWindow_SelectPane (on page 11)*
- *OptionsWindow_EnumeratePanes (on page 14)*

Reading the state of the OptionsWindow dialogue

The OptionsWindow object has only one piece of state information, the 'modified flag. It can be manipulated with:

- *OptionsWindow_SetModified (on page 12)*
- *OptionsWindow_GetModified (on page 13)*

Modification checks

The OptionsWindow will monitor the panes currently displayed for changes to the standard gadgets. In particular, the gadget monitors :

- OptionButton_StateChanged
- RadioButton_StateChanged
- WritableField_ValueChanged
- Slider_ValueChanged
- Adjuster_Clicked
- NumberRange_ValueChanged
- StringSet_ValueChanged
- ColourSwatch_ColourChanged

⚠ FIXME: Link these with references

Any of these events occurring on a pane Window will cause the OptionsWindow to be marked as modified automatically. This behaviour can be disabled through a flag in the OptionsWindow template. Any changes which are caused by events outside these, for example through a FontDBox object attached to the window, will need to be notified a method call to the OptionsWindow.

Pane Window objects

Each Pane window must conform to a number of very simple requirements :

- It must have a window foreground of 'transparent'.
  This will cause the title bar and window borders to be removed.
- It must have a title which will be used on the radio icon to select the window.
  Although the title is not used for the Window, it will still be present and its use on the radio icon allows the radio icons to be internationalised easily.

- The extent of the window must cover the extent used by the options within the window.
  The window will automatically be extended to fill the space required by the largest of the panes, and the main window's options and buttons.
- As a consequence of the above, there must be no gadgets outside the apparent visible area of the Window as these may be exposed when the window is extended.
- It is recommended that the title be one or two words, preferably including a noun describing the area to be configured.

Other than this, the Window can contain any gadget which it requires.

Action Buttons

The buttons displayed on the OptionsWindow object can be controlled through the flags given in the Object template. The options which can be used on the OptionsWindow are :

- Set
- Cancel
- Save
- Default

These are independant of one another, but certain combinations make little sense. In particular, no buttons or just a Default button will actually cause the Cancel button to be included on the dialogue. Without this Cancel button, the window would not be closable by the user.

User events

The OptionsWindow is intended to use modular configuration panes. That is, the changes to panes are requested, or applied individually. This should reduce the processing required to just those panes which have been accessed by the user.

A typical example of this is that when showing an OptionsWindow for the first time, the first pane will be selected. As the pane is selected, an event will be sent to the application to request that it update the pane with the current settings. The application should only update that single pane. When a new pane is selected (either by the user or through a method call), an event will be delivered for that pane.

When the user selects 'Set', the application will be sent a number of events to read the settings from them. This will only occur for the panes which have been made visible. Once set, all the panes will be marked as not having been seen and will not be reset.

When the 'Default' button is pressed, all the panes must be updated with the default settings, and events will be sent to the application to that effect.

The 'Save' operation is a combination of the 'Set' operation for each pane which had been visible and a save event for the entire settings; that is, the application will receive a number of events to read the settings, followed by a save event.

At any time, the dialogue may be closed by the user, either by pressing the

'Save', 'Set' or 'Cancel' buttons, or by the relevant key press. This will result in an event being delivered to inform the application that the dialogue has been completed. No operation should be performed on the configuration on receipt of this event, but any associated objects should be deleted or at least hidden by the application.

OptionsWindow templates

The layout of a OptionsWindow template is shown below. Fields which have types MsgReference and StringReference are those which will require relocation when they are loaded from a resource file. If the template is being constructed in memory, then these fields should be real pointers (i.e. they do not require relocation).

For more details on relocation, see ?SUPPORT310.HTML#65428.

Field	Size in bytes	Type
flags	4	word
title	4	MsgReference
windowlist	4	StringReference

OptionsWindow Wimp event handling

Wimp event	Action
Open Window	Show the dialogue box
Key Click	if Escape, then cancel this dialogue if Return pressed, perform the default action.
User Message	Events as listed in *Modification checks (on page 5)* Mark the object as modified ActionButton_Selected Perform the relevant operation for the button pressed RadioButton_StateChanged Change the pane displayed.

Toolbox methods

The following methods are all invoked by calling SWI Toolbox_ObjectMiscOp with:

Value	Meaning
R0	holding a flags word
R1	being a OptionsWindow Dialogue object id
R2	being the method code which distinguishes this method
R3-R9	potentially holding method-specific data

OptionsWindow_GetWindowID (Method &0)

Read the Window ObjectId used for this object

On entry

R0 = flags
R1 = OptionsWindow object id
R2 = 0

On exit

R0 = window object id for this OptionsWindow object

Use

This method returns the id of the underlying Window object used to implement this OptionsWindow object.

Declarations

```
extern _kernel_oserror *optionswindow_get_window_id (
unsigned int flags, ObjectId optionswindow, ObjectId
*window );
```

OptionsWindow_AddPane (Method &1)

Add a new pane to the OptionsWindow object

On entry

R0 = flags (reserved, must be 0)
R1 = OptionsWindow object id
R2 = 1
R3 = new component id for pane
R4 = Window object id for new pane

On exit

R1 - RR9 preserved

Use

This method add a new pane to the list of panes used by the OptionsWindow. If this is the first pane and the OptionsWindow is shown, the pane will be selected.

Declarations

```
extern _kernel_oserror *optionswindow_add_pane ( unsigned
int flags,
ObjectId optionswindow,
ComponentId new_componentid,
ObjectId new_panewindow
);
```

OptionsWindow_RemovePane (Method &2)

Remove an existing Pane from the OptionsWindow

On entry

R0 = flags
R1 = OptionsWindow object id
R2 = 2
R3 = component id of the pane to remove

On exit

R1 - RR9 preserved

Use

This method removes a pane from those controlled by the OptionsWindow. If the specified pane is currently selected, the OptionsWindow will revert to the first pane.

Declarations

```
extern _kernel_oserror *optionswindow_remove_pane (
unsigned int flags,
ObjectId optionswindow,
ComponentId component
);
```

OptionsWindow_SelectPane (Method &3)

Selects a pane of the OptionsWindow object for display

On entry

R0 = flags
R1 = OptionsWindow object id
R2 = 3
R3 = component id of the pane to display

On exit

R1 - RR9 preserved

Use

This method displays a given pane.

Declarations

```
extern _kernel_oserror *optionswindow_select_pane (
unsigned int flags,
ObjectId optionswindow,
ComponentId component
);
```

OptionsWindow_SetModified (Method &4)

Change the modification details for the OptionsWindow object

On entry

R0 = flags
R1 = OptionsWindow object id
R2 = 1
R3 = value

On exit

R1 - RR9 preserved

Use

This method sets whether the options have been modified or not. If the value passed in R3 is 0, this indicates that the options is not modified; any other value in R3 means the options have been modified.

Declarations

```
extern _kernel_oserror *optionswindow_set_modified (
unsigned int flags, ObjectId optionswindow, int modified
);
```

OptionsWindow_GetModified (Method &5)

Read the modification details for the OptionsWindow object

On entry

R0 = flags
R1 = OptionsWindow object id
R2 = 2

On exit

R0 = modified state (0 = unmodified; non-0 = modified)

Use

This method returns whether the options have been modified or not.

Declarations

```
extern _kernel_oserror *optionswindow_get_modified (
unsigned int flags, ObjectId optionswindow, int *modified
);
```

OptionsWindow_EnumeratePanes (Method &6)

Enumerate the panes used by the OptionsWindow

On entry

R0 = flags
R1 = OptionsWindow object id
R2 = 2
R3 = last pane component id, or -1 initially

On exit

R0 = pane Window object id if R3 contains a valid component id
R3 = component id of this pane, or -1 if no more components

Use

This method enumerates the panes attached to the OptionsWindow.

Declarations

```
extern _kernel_oserror *optionswindow_enumerate_panes (
unsigned int flags,
ObjectId optionswindow,
ComponentId last_component,
ObjectId *window,
ComponentId *component
);
```

Toolbox events

The OptionsWindow object generates the following Toolbox events:

OptionsWindow_DialogueCompleted (Event &100280)

Notification that a OptionsWindow object is no longer visible

Message

Offset	Contents
R1+8	&100280
R1+12	flags (none yet defined)

Use

This Toolbox event is raised after the OptionsWindow object has been hidden, either by the user Select clicking on the buttons, or the equivilent keyboard operation. It allows the client to tidy up its own state associated with this dialogue.

Declarations

```
typedef struct
{
ToolboxEventHeader hdr;
} OptionsWindowDialogueCompletedEvent;
```

OptionsWindow_FillInPaneDefault (Event &100281)

Request that the application fill in a pane with the default settings

Message

Offset	Contents
R1+8	&100281
R1+12	flags (none yet defined)
R1+16	pane object id

Use

This Toolbox event is raised to request that the application fill in a pane with the default settings. The application's Id Block will be filled in with the OptionsWindow object id and pane component number.

Declarations

```
typedef struct
{
ToolboxEventHeader hdr;
ObjectId pane_window;
} OptionsWindowFillInPaneEvent;
```

OptionsWindow_FillInPaneCurrent (Event &100282)

Request that the application fill in a pane with the current settings

Message

Offset	Contents
R1+8	&100282
R1+12	flags (none yet defined)
R1+16	pane object id

Use

This Toolbox event is raised to request that the application fill in a pane with the current settings. The application's Id Block will be filled in with the OptionsWindow object id and pane component number.

Declarations

```
typedef struct
{
ToolboxEventHeader hdr;
ObjectId pane_window;
} OptionsWindowFillInPaneEvent;
```

OptionsWindow_ConfigurePane (Event &100283)

Request that the application read the settings from an OptionsWindow pane

Message

Offset	Contents
R1+8	&100283
R1+12	flags (none yet defined)
R1+16	pane object id

Use

This Toolbox event is raised to request that the application read the details from a pane and apply them to the current settings. The application's Id Block will be filled in with the OptionsWindow object id and pane component number.

Declarations

```
typedef struct
{
ToolboxEventHeader hdr;
ObjectId pane_window;
} OptionsWindowConfigurePaneEvent;
```

OptionsWindow_Save (Event &100284)

Request that the application save all settings

Message

Offset	Contents
R1+8	&100284
R1+12	flags (none yet defined)

Use

This Toolbox event is raised to request that the application save its settings. The application's Id Block will be filled in with the OptionsWindow object id.

Declarations

```
typedef struct
{
ToolboxEventHeader hdr;
} OptionsWindowSaveEvent;
```

Document information

Maintainer(s): RISCOS Ltd <developer@riscos.com>

History:

Revision	Date	Author	Changes
1	07 Jun 2004	ROL	First version