A WorkLife FrameWork™ Palettes

The following sections describe each of the individual Palettes that A WorkLife FrameWork™ provides. In each case he Palette is broken up into its individual buttons and each button is described. The documentation for each Palette can be easily accessed from that Palette by clicking on the **?** button at the top right of the palette.

General Properties of A WorkLife FrameWork™'s Palettes

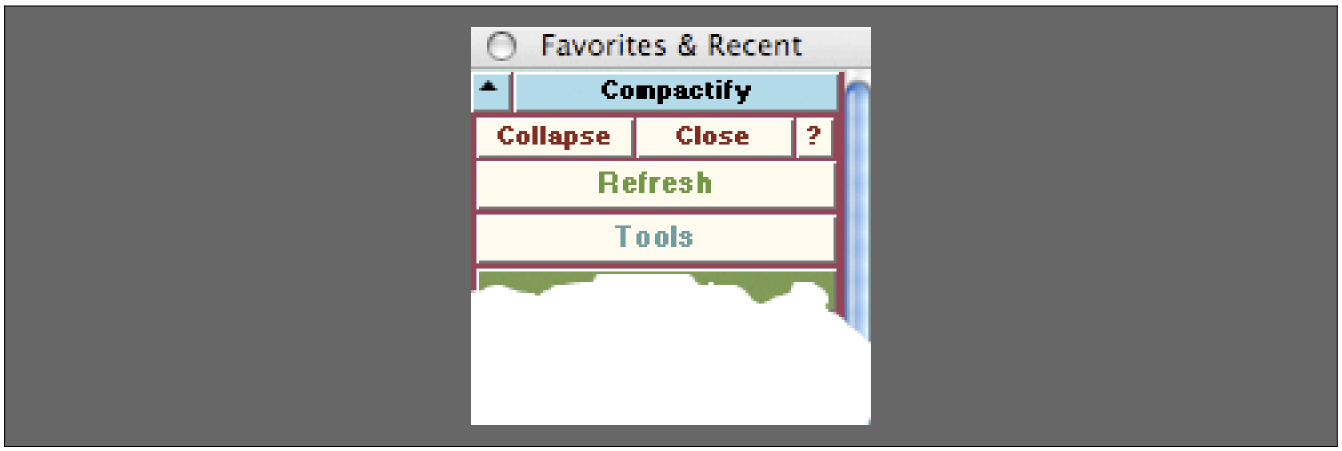
Common Administrative Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Most Palettes amongst A WorkLife FrameWork™'s Palettes have a set of buttons at the top that are "administrative." They close the Palette, expand or compactify the Palette, refresh the Palette, and open up the home Palette (also known as the WorkLife Tools Palette).

Not all of these functions are available in each Palette; for example, the Refresh button is only present for those Palettes that might need to be refreshed (so that the contents of their buttons update).

Here is an example of such as set of heading buttons from the Favorites & Recent Palette:



Heading Buttons from the Favorites & Recent Palette

Palette Administrative Buttons

The behaviors of each of these buttons is as follows for the case of the Favorites & Recent Palette:

Within a Palette may be sub-palettes that may be open or closed. Clicking on the **Compactify** button will close all of the sub-palettes. Clicking on the ▲ button expands all of the sub-palettes. Not all palettes have this pair of buttons—it is only present if there are sub-palettes within the given Palette.

Clicking on the **?** button opens the *Mathematica* Help Browser at the section where the given Palette is documented..

Clicking on the **Close** button closes the Palette—the package remembers its position on the screen.

Clicking on the **Collapse** button collapses the Palette into a small Palette that acts as a placeholder for the Palette on the screen. The resulting Palette looks like:

Favorites & Recent Palette in its Collapsed Form

In this collapsed Palette, clicking on the **Expand** button returns the Palette to its original form. Clicking on the Close button closes the Palette but the package remembers its position on the screen.

The **Refresh** button regenerates the buttons on the Palette.

The **Tools** button opens up the WorkLife Tools Palette.

The WorkLife Tools Palette and Its Sub-Palettes

The following ten Palettes are directly accessible from the top level of the **WorkLife Tools** Palette. They are listed in the order that they appear on the **WorkLife Tools** Palette.

WorkLife Tools Palette

The Palette

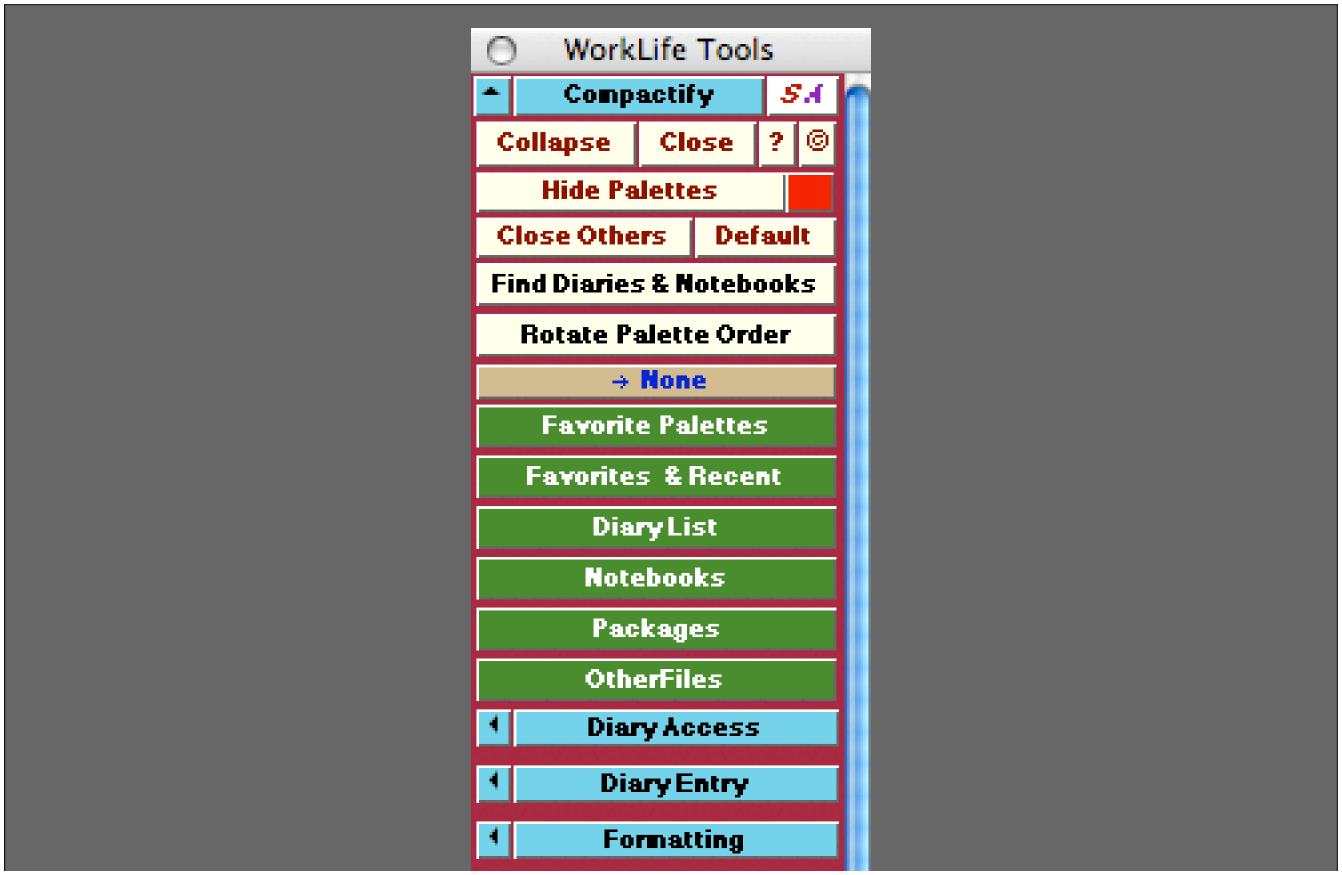
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

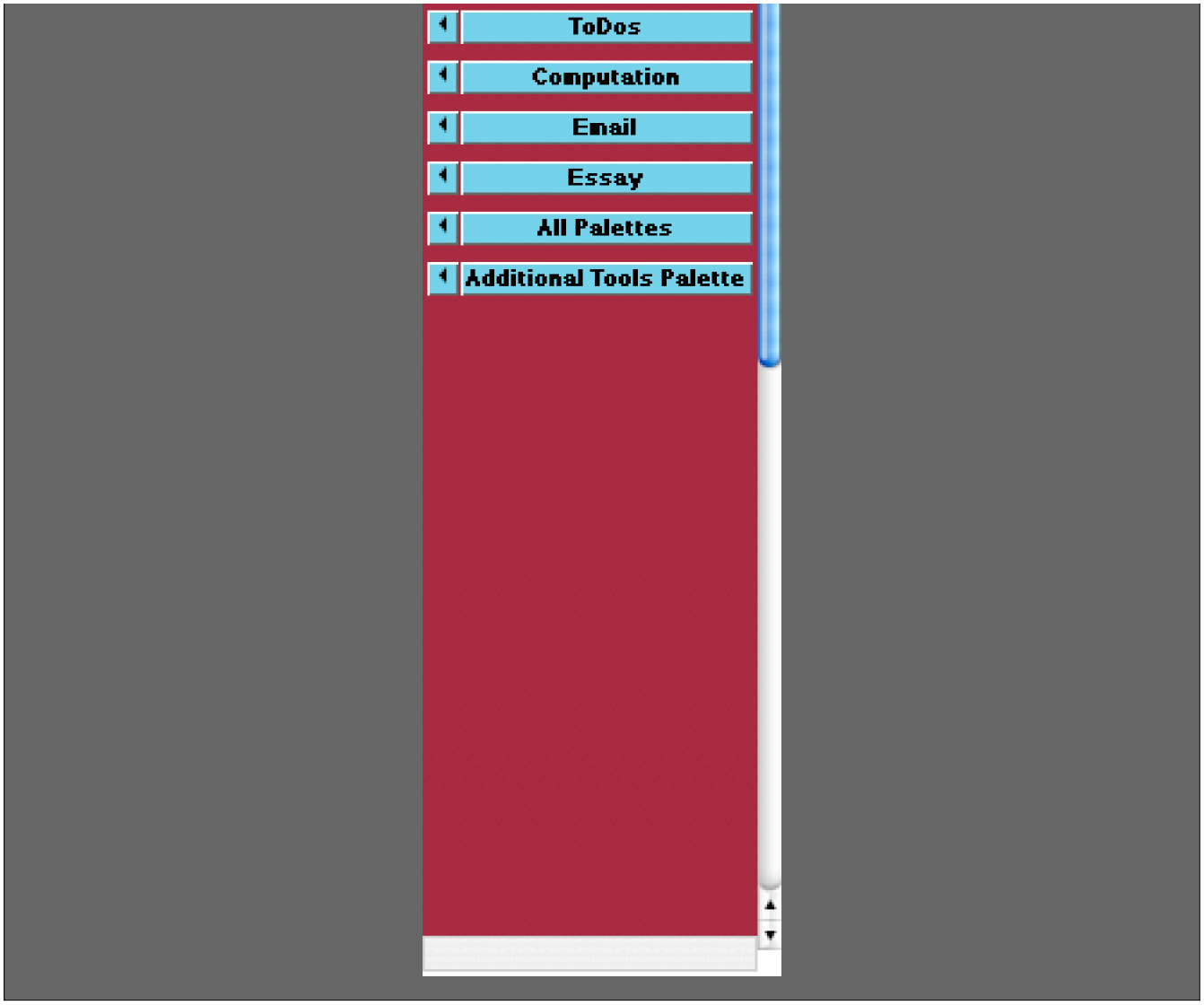
The WorkLife Tools Palette is the general starting point for using the WorkLife FrameWork™ Package. If it is not visible it can be opened by executing

```
WorkLifeToolsPalette[];
```

by clicking the **WorkLife Tools** button on the All Palettes Palette, or by clicking on the **Tools** button near the top of most Palettes in the Package.

When first opened in a new session of A WorkLife FrameWork, the WorkLife Tools Palette looks like the Palette below (and it also looks like this is has been when compactified by clicking on the **Compactify** button at the top of the Palette). If, for example, the Diary Access sub-palette is opened, it looks as on the right.





The WorkLife Tools Palette as it appears when first opened

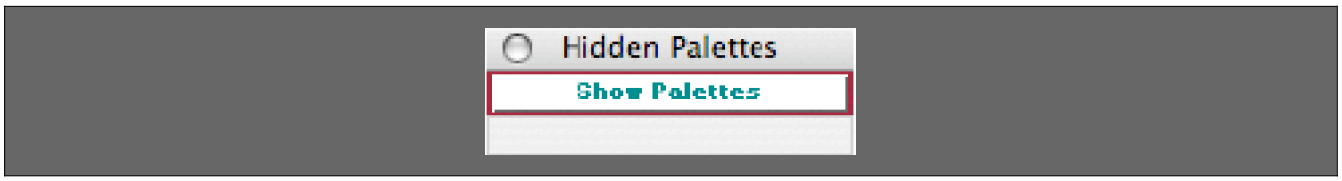
In this image of the WorkLife Tools Palette all of its sub-palettes are seen to be closed. You can toggle between each of these sub-paletts by clicking on their main button—the **Diary Access** button in the cse of the **Diary Access** Palette. If you click on the ◂ button to its left, the **Diary Access** Palette will open as a separate Palette in its own right, and this is the case for the other sub-palettes as well. For more on the Diary Access Palette see its documentation.

The Administrative buttons on WorkLife Tools Palette include the familiar **Compactify** buttons with the addition of a small button to open the *Scientific Arts* web site,

the **Close**, **Collapse**, and **?** buttons along with a Copyright © button,

and finally three buttons that are unique to the WorkLife Tools Palette.

The first two include the **Hide Palettes** button which hides all of the Palettes from the screen and replaces them with a single place holder Palette that looks like:



When the **Show Palettes** button is clicked, the original configuration of the Palettes is reopened.

The red button to the right of the **Hide Palettes** button has a similar purpose. When clicked, all of the Palettes from the screen are hidden and a single green button,  , is placed on the screen in one corner. Clicking on this button reverts to the original configuration of the Palettes.

> *The location of the*  *button is determined by the value of the parameter* `$NuggetLocation`. *Its default value is* `"LeftTop"` *(a string). The three other possibilities are* `"LeftBottom"`, `"RightTop"`, *and* `"RightBottom"`.

The **Close Others** button closes all currently open Palettes except for the WorkLife Tools Palette.

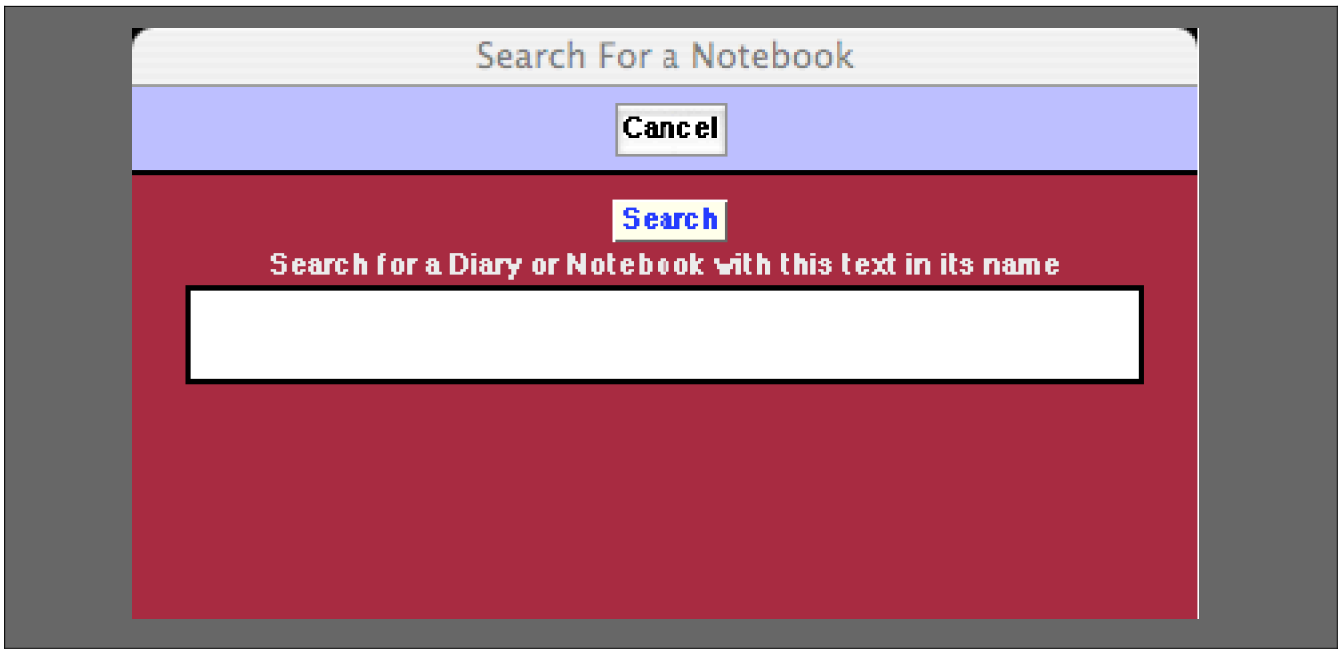
> *The specification of which Palettes are left open when this button is clicked is determined by the* `LeaveOpen` *option to the function* `CloseAllPalettes` *which has the default value:* `LeaveOpen→{"$WorkLife ToolsPaletteNotebook"}`.

The **Default** button resets the currently open set of Palettes to be only those in the Default Set of Palettes. To specify your default set of palettes use the **Choose Default Set** Button from the **Favorite Palettes** Palette.

This opens a dialog that allows you to search for Diaries and Notebooks by typing a portion of their name. This can also be opened programmatically by executing:

```
FindDiaryOrNotebookOrPackageDialog[];
```

The dialog looks like the following:



The Search For a Notebook dialog initiated by the **Find Diaries & Notebooks** button.

In this dialog you can type any snippet of the name of a Notebook or of a Diary and the Search button will yield all examples of known Diaries or Notebooks that match that form. You can also use wildcard such as *.

Rotate Palette Order

When there are a large number of overlapping palettes on the screen, this button cyclically rotates them so that you can bring whichever pallet you wish to access to the front. An alternative way to do this is to use the **All Notebooks** Palette.

↓Documentation Notes

This button shows the name of the current Diary. It is refreshed whenever a Diary is made the current Diary. In this example it is the Diary named **Documentation Notes**. If there is no current Diary then this button reads **None**. The current Diary can be opened (or brought to the front, if it is already open) by clicking on this button itself.

Favorite Palettes

Opens the **Favorite Palettes** Palette. This can also be opened programmatically by executing:

```
FavoritePalettesPalette[];
```

Diary Access Palette

The Palette

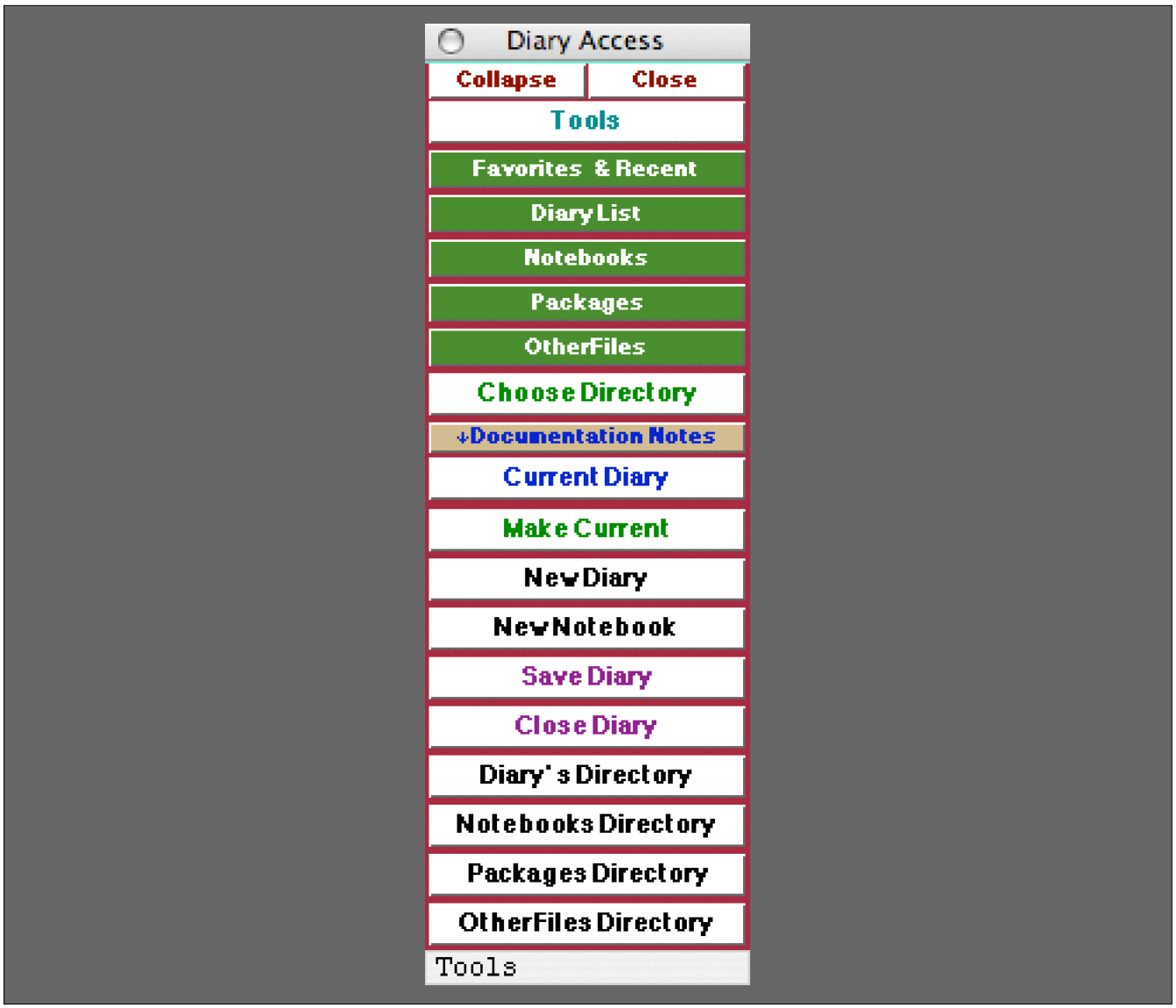
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **Needs["Diary`Diary`"]**, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the ◂ button to the left of **Diary Access** on the WorkLife Tools Palette, you will open the Diary Access Palette. You can also open the Diary Access Palette by executing:

```
DiaryAccessPalette[];
```

or by clicking the Diary Access button on the All Palettes Palette.

The Diary Access Palette looks like:



The Diary Access Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Opens the **Favorites & Recent** Palette. This can also be opened programmatically by executing:

```
FavoritesAndRecentPalette[];
```

Opens the **Diary List** Palette. This can also be opened programmatically by executing:

```
DiaryListPalette[];
```

The Diaries listed in the **Diary List** Palette are those in the Default Diary Directory. This directory can be specified using the **Choose Directory** button below.

Opens the **Notebooks** Palette. This can also be opened programmatically by executing:

```
NotebooksPalette[];
```

Opens the **Packages** Palette. This can also be opened programmatically by executing:

```
PackagesPalette[];
```

Opens the **OtherFiles** Palette. This can also be opened programmatically by executing:

```
OtherFilesPalette[];
```

This opens the Directory Browser dialog window. In this window you can navigate your computer's directories and choose the one that you wish to open a Diary from. The resulting directory is then the value of `$DefaultDiaryDirectory`.

```
$DefaultDiaryDirectory
```

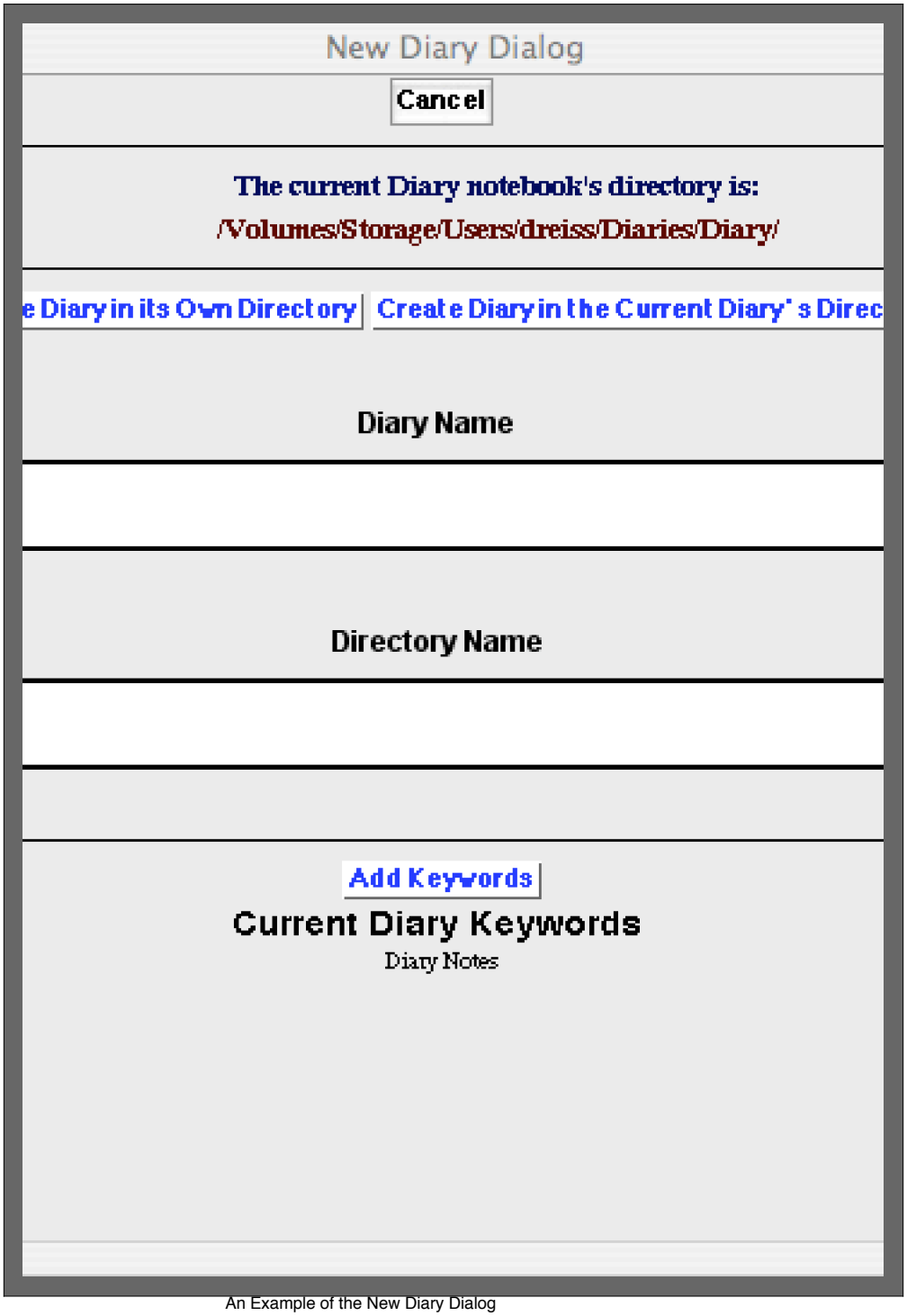
This button shows the name of the current Diary. It is refreshed whenever a Diary is made the current Diary. In this example it is the Diary named **Documentation Notes**. If there is no current Diary then this button reads **None**. The current Diary can be opened by using the **Current Diary** button below or by clicking on this button itself. Note that when the **Diary Access** palette buttons are opened as a sub-palette of the **WorkLife Tools** palette, this button does not appear. Rather, it appears towards the top of the WorkLife Tools Palette.

This opens the **Current Diary** if it has been chosen. If not, an error message is generated.

If the current Diary is open, then its window is brought to the front. If it is not opened, then it is opened and any Default Code Cells are executed.

If a Diary other than the current Diary is the topmost notebook, then clicking on this button will designate topmost notebook as the current Diary.

This opens up a dialog window to create a **New Diary**. The **New Diary** dialog looks like:



An Example of the New Diary Dialog

At the top of this dialog is information on the current Diary directory.

If you enter a name in the **Diary Name** field and click on **Create Diary in the Current Diary's Directory**, then a new Diary with that name will be created there and that Diary will open and become the current Diary. (In this case any entry into the **Directory Name** field will be ignored.)

If, instead, you enter a name in the **Diary Name** field and a directory name in the **Directory Name** field and click on **Create Diary in its own Directory**, then a new Diary with that name will be created in a subdirectory of the current Diary's directory. The new Diary will open and become the current Diary and the new directory will become the current Diary's directory. (If no name is entered in the **Directory Name** field an error message will be returned if you click on **Create Diary in its own Directory**.)

> *If the name of the Diary does not contain any of the listed keywords, then an advisory error message will be generated. That diary will not be listed on the Diary List Palette. Additional keywords can be added to the list of keywords via the dialog that appears when you click on the* **Add Keywords** *button. You can make the Diary List Palette accept all diary names by executing* `AddDiaryKeywords[All]`.

This opens up a dialog window to create a **New Notebook**. This notebook is placed by default in the Notebooks subdirectory of the current Diary's directory.

This **Save**s the current **Diary**. (This may not be the current InputNotebook in the Front End.) The Diary is processed prior to saving to make sure that all cells are properly tagged and dated.

This saves and then **Close**s the current **Diary**. (This may not be the current InputNotebook in the Front End.) The Diary is processed prior to saving to make sure that all cells are properly tagged and dated. An alert dialog is displayed as this processing occurs.

This opens the current **Diary's Directory**.

This opens the **Notebooks** subdirectory of the current Diary's directory.

This opens the **Packages** subdirectory of the current Diary's directory.

This opens the **OtherFiles** subdirectory of the current Diary's directory.

Diary Entry Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

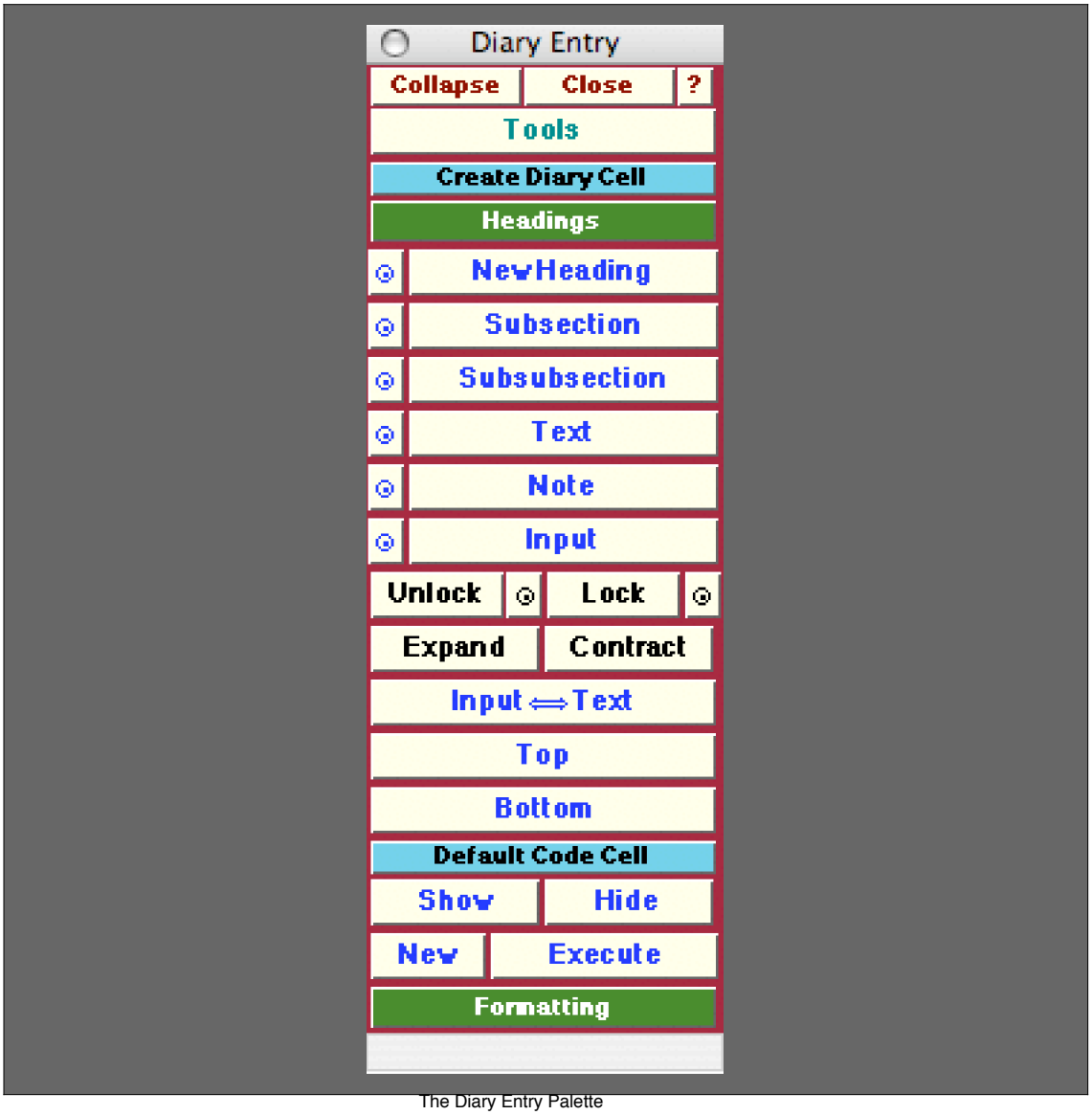
By clicking on the on the ◂ button to the left of **Diary Entry** on the WorkLife Tools Palette, you will open the Diary Entry Palette. You can also open the Diary Entry Palette by executing:

```
DiaryEntryPalette[];
```

or by clicking the Diary Entry button on the All Palettes Palette.

Do not confuse the DiaryEntryPalette with DiaryEntriesPalette

The Diary Entry Palette looks like:



The Diary Entry Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

- **Create Diary Cell**

This button toggles open and closed the **Create Diary Cell** sub-palette containing the following six buttons concerning the insertion of cells into the current Diary.

This opens the Diary **Headings** Entry dialog.

The **New Heading** button places a new Section cell at the end of the Current Diary. If the Diary is the topmost window then it is brought to the top. If the Diary is not open then it is opened.

The button with the "⊙" on it also places a new Heading cell in the current Diary. However, rather than at the end of the Diary, it places it at the current insertion point. (If the insertion point is within a cell, then the **New Heading** Cell is placed below that cell.)

The **Subsection** button places a new Subsection cell at the end of the Current Diary. If the Diary is the topmost window then it is brought to the top. If the Diary is not open then it is opened.

The button with the "⊙" on it also places a new Subsection cell in the current Diary. However, rather than at the end of the Diary, it places it at the current insertion point. (If the insertion point is within a cell, then the new Subsection Cell is placed below that cell.)

The **Subsubsection** button places a new Subsubsection cell at the end of the Current Diary. If the Diary is the topmost window then it is brought to the top. If the Diary is not open then it is opened.

The button with the "⊙" on it also places a new Subsubsection cell in the current Diary. However, rather than at the end of the Diary, it places it at the current insertion point. (If the insertion point is within a cell, then the new Subsubsection Cell is placed below that cell.)

The **Text** button places a new Text cell at the end of the Current Diary. If the Diary is the topmost window then it is brought to the top. If the Diary is not open then it is opened.

The button with the "⊙" on it also places a new Text cell in the current Diary. However, rather than at the end of the Diary, it places it at the current insertion point. (If the insertion point is within a cell, then the new Text Cell is placed below that cell.)

The **Note** button places a new Note cell at the end of the Current Diary. If the Diary is the topmost window then it is brought to the top. If the Diary is not open then it is opened.

The button with the "⊙" on it also places a new Note cell in the current Diary. However, rather than at the end of the Diary, it places it at the current insertion point. (If the insertion point is within a cell, then the new Note Cell is placed below that cell.)

The **Input** button places a new Input cell at the end of the Current Diary. If the Diary is the topmost window then it is brought to the top. If the Diary is not open then it is opened.

The button with the "⊙" on it also places a new Input cell in the current Diary. However, rather than at the end of the Diary, it places it at the current insertion point. (If the insertion point is within a cell, then the new Input Cell is placed below that cell.)

Additional CellStyles can be added to the Diary Entry Palette at this location by using the function `AddCellStylesToDiaryEntryPalette`.

The **Unlock** button unlocks all of the cells in the current Diary. The button with the "⊙" adjacent to the Unlock button unlocks the currently selected cell in the current Diary.

The **Lock** button unlocks all of the cells in the current Diary. The button with the "⊙" adjacent to the Lock button unlocks the currently selected cell in the current Diary.

The **Expand** button expands all cell groups in the current Diary. The **Contract** button contracts all cell groups in the current Diary.

This button toggles the default behavior when typing into a new cell in the current Diary. The default for a Diary is to create a Text cell. When clicked, this button changes this behavior to creating an **Input** cell. Clicking it again toggles it back to the creation of a **Text** cell.

This moves the insertion point to the **Top** of the current Diary. If the Diary is the topmost window then it is brought to the top. If the Diary is not open then it is opened.

This moves the insertion point to the **Bottom** of the current Diary. If the Diary is the topmost window then it is brought to the top. If the Diary is not open then it is opened.

This button toggles open and closed the sub-palette containing the following two pairs of buttons concerning the default code cells of the current Diary.

The **Show** button scrolls to and opens the Default Code Cells in the current Diary notebook. The **Hide** button hides the Default Code Cells in the current Diary notebook.

The **New** Button creates a new Default Code Cell in the current Diary notebook and places it below the current Default Code Cells while showing them all. The **Execute** button executes all of the Default Code Cells in the current Diary notebook.

This button opens the **Formatting** Palette.

Formatting Palette

The Palette

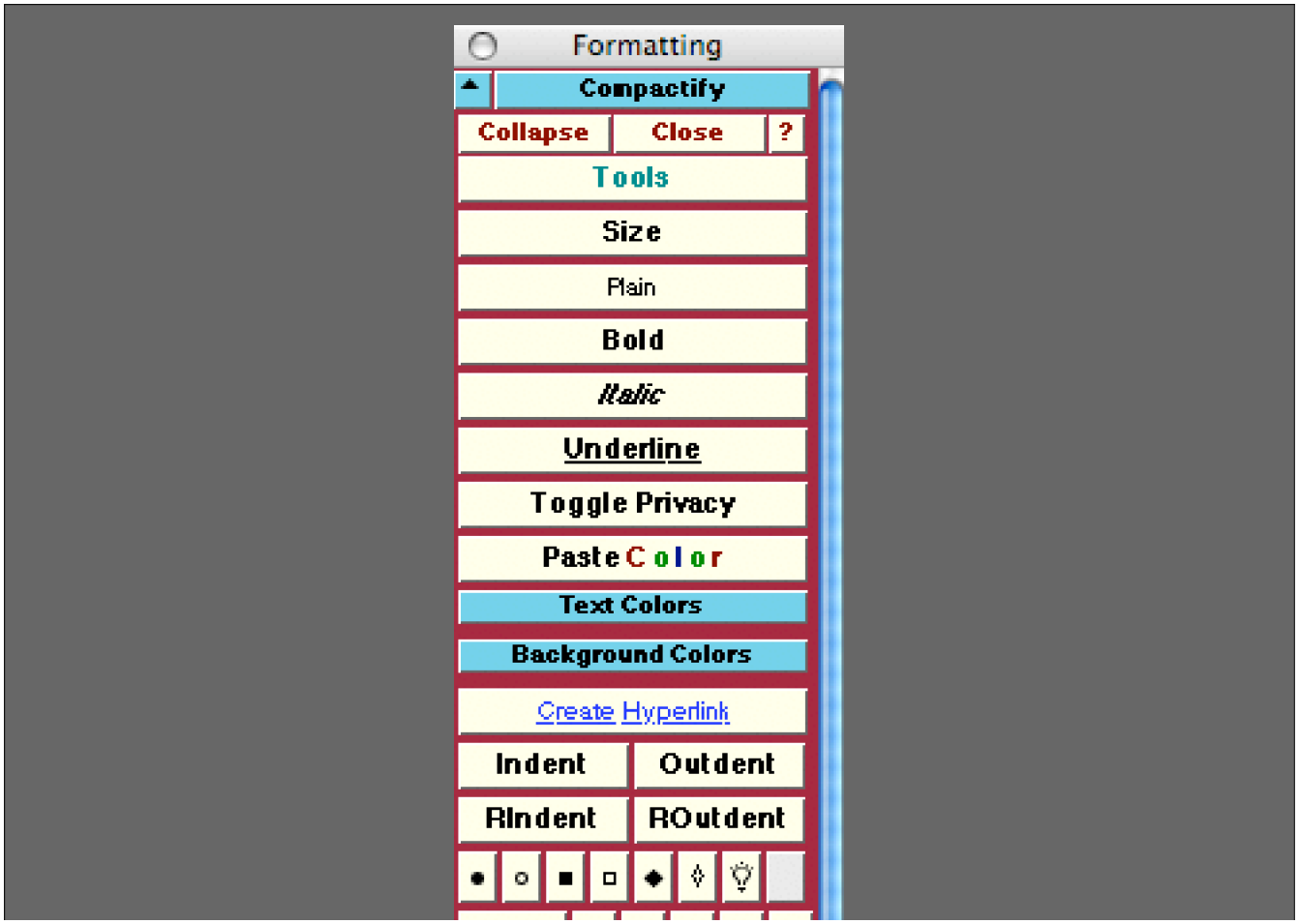
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

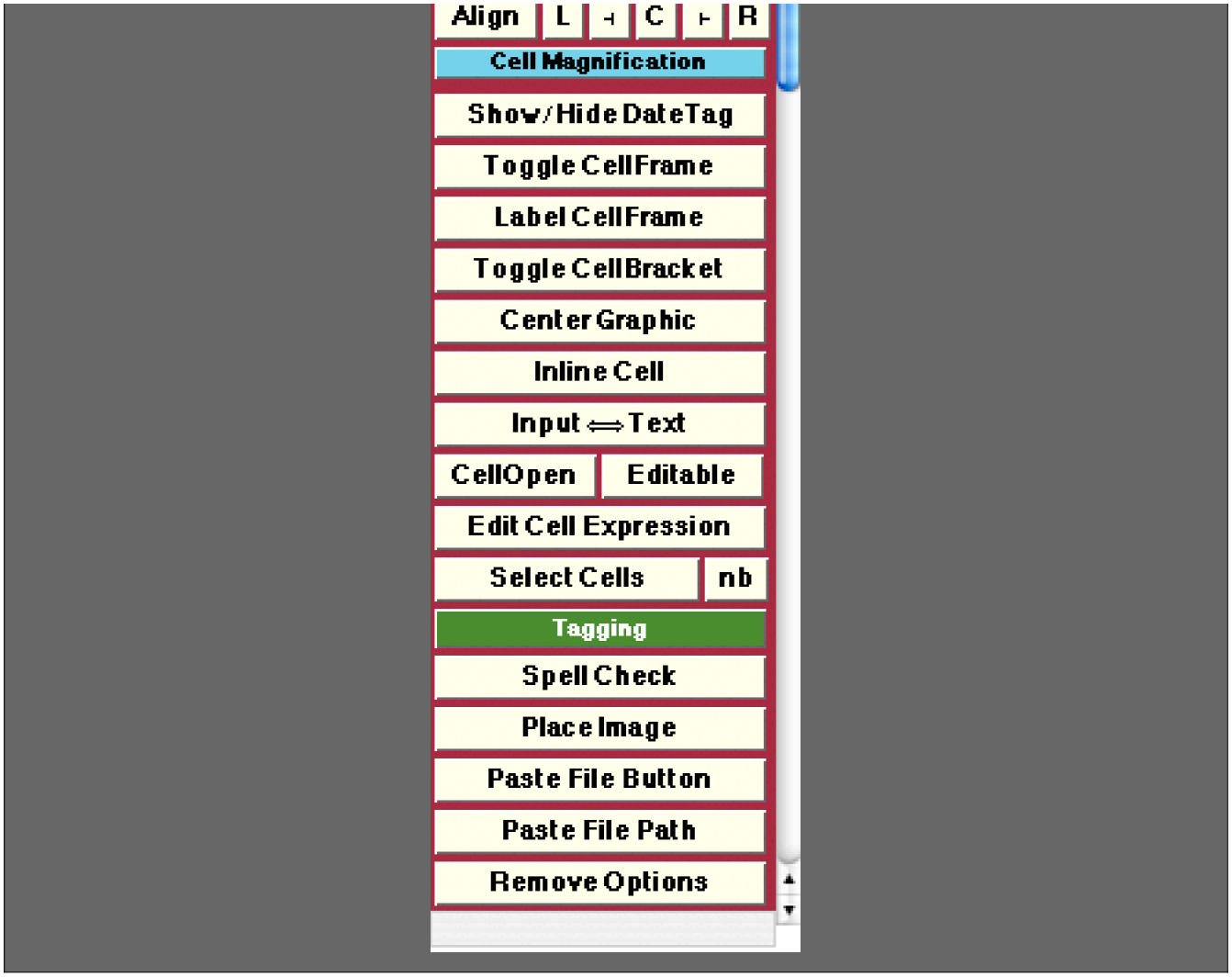
By clicking on the on the ◂ button to the left of **Formatting** on the WorkLife Tools Palette, you will open the Formatting Palette. You can also open the Formatting Palette by executing:

```
FormattingPalette[];
```

or by clicking the Formatting button on the All Palettes Palette.

The Formatting Palette looks like:





The Diary Formatting Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Size

This button brings up a dialog to choose the font size of the current selection in the InputNotebook.

This button changes the selected text in the InputNotebook to Plain.

This button changes the selected text in the InputNotebook to **Bold**.

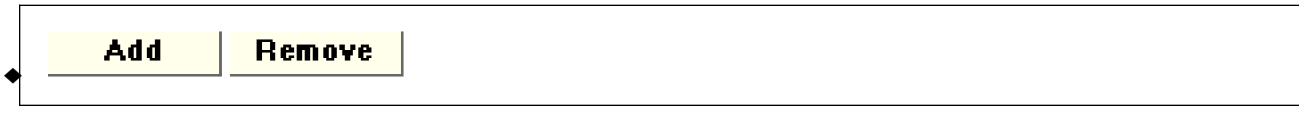
This button changes the selected text in the InputNotebook to ***Italic***.

This button changes the selected text in the InputNotebook to **Underline**.

This button toggles the currently selected cell in the InputNotebook between private and public. A private cell has its contents reduced in size and rendered in an incomprehensible font. This is useful for writing when in a public place if you do not want what you are writing to be publicly readable. Such a cell would look like the following:

This button brings up a color selector dialog. The selected color will then be pasted into the current InputNotebook in the form of an `RGBColor` directive.

Opens and closes the **Text Colors** sub-palette which includes the seven buttons that follow. To add additional colors to this sub-palette, use the function `AddFormattingTextColors`. To interactively create a color to use as an argument to `AddFormattingTextColors`, use the preceding button or execute `PasteColorFromDialog[]`. To remove colors other than those in the default set, use the function `DeleteFormattingTextColors`.



Add brings up a dialog that allows you to add a custom color to the list below. **Remove** brings up a dialog that allows

you to remove a custom color that you have perviously added from the list below.

- **Color**

Brings up a **Color** selector dialog. The color selected via this dialog is applied to the text of the selection in the InputNotebook.

- **Black**

Colors the selected text in the InputNotebook **Black**.

- **White**

Colors the selected text in the InputNotebook **White**.

- **Red**

Colors the selected text in the InputNotebook **Red**.

- **Green**

Colors the selected text in the InputNotebook **Green**.

- **Blue**

Colors the selected text in the InputNotebook **Blue**.

- **Background Colors**

Opens and closes the **Background Colors** sub-palette which includes the nine buttons that follow. To add additional colors to this sub-palette use the function `AddFormattingBackgroundColors`. To interactively create a color to use as an argument to `AddFormattingBackgroundColors` use the **Paste Color** button or execute `PasteColorFromDialog[]`. To remove colors other than those in the default set use the function `DeleteFormattingBackgroundColors`.

- **Add** **Remove**

Add brings up a dialog that allows you to add a custom color to the list below. **Remove** brings up a dialog that allows you to remove a custom color, that you have perviously added to it, from the list below.

- 

Brings up a **Color** selector dialog. The color selected via this dialog is applied as the background of the currently selected cells in the InputNotebook.

- 

Renders the background of the selected cells in the InputNotebook **Black**.

- 

Renders the background of the selected cells in the InputNotebook **White**.

- 

Renders the background of the selected cells in the InputNotebook **10% Gray**.

- 

Renders the background of the selected cells in the InputNotebook **33% Gray**.

- 

Renders the background of the selected cells in the InputNotebook **50% Gray**.

- 

Renders the background of the selected cells in the InputNotebook **66% Gray**.

- 

Renders the background of the selected cells in the InputNotebook **90% Gray**.

- 

Brings up a dialog to convert the selected text in the InputNotebook into a hyperlink.

The **Indent** button indents the currently selected cells by an amount given by the parameter `$IndentCellDefault`. If multiple cells are chosen, then all are indented. If the level of indentation of the selected cells are different from one another, then all are brought into alignment.

The **Outdent** button decreases the indentation of the currently selected cells by an amount given by the parameter `$IndentCellDefault`. If multiple cells are chosen, then all are outdented. If the level of indentation of the cells are different from one another, then all are brought into alignment.

The **RIndent** button indents the currently selected cells *from the right* by an amount given by the parameter `$IndentCellDefault`. If multiple cells are chosen, then all are right ndented. If the level of right indentation of the selected cells are different from one another, then all are brought into alignment.

The **ROutdent** button decreases the indentation of the currently selected cells *from the right* by an amount given by the parameter `$IndentCellDefault`. If multiple cells are chosen, then all are right outdented. If the level of right ndentation of the cells are different from one another, then all are brought into alignment.

These assign the indicated dingbats to the currently selected cells in the InputNotebook. The final gray button removes all dingbats from these cells.

These buttons align the text in the selected cells. The **L** button aligns to the left. The ⊣ button aligns the text at 25%. The **C** button centers the text. The ⊢ button aligns the text at 75 %. And the **R** button aligns the text to the right. The **Align** button returns the alignment of the text to the default value for the cell.

Opens and closes the **Cell Magnification** sub-palette which includes the single line of buttons that follow.

Each of these buttons assigns a cell magnification of the indicated factor to the currently selected cell in the InputNotebook.

Toggles between showing and hiding the date tag of the selected cell if it has one.

Toggles a cell frame around the selected cell in the InputNotebook.

Opens a dialog which allows you to place a cell frame label on the selected cell in the InputNotebook. Choices of placement are `Left`, `Top`, `Bottom`, and `Right`.

Toggles the cell bracket visible/invisible for the selected cell in the InputNotebook.

Centers a graphic in its cell.

Creates an **Inline Cell** at the insertion point of the InputNotebook.

This button toggles the default behavior when typing into a new cell in the InputNotebook. The default for a Notebook is to create an **`Input`** cell. When clicked, this button changes this behavior to creating an Text cell. Clicking it again toggles it back to the creation of an **`Input`** cell.

CellOpen opens the currently selected cell in the InputNotebook if it is currently closed. If it is not Editable, the **Editable** button will make it so.

Edit Cell Expression opens a notebook where the Cell expression for the currently selected cell in the InputNotebook can be edited. In that notebook, when you are done editing the Cell expression, you can click the button **Apply Edits** to update the originally selected cell.

Select Cells selects all cells in the current InputNotebook that have the same cell style as the currently selected cell. **nb** selects, copies, and creates a new Notebook with those cells.

Opens the **Tagging** Palette.

Starts the *Mathematica* spelling checker starting at the current selection or input point of the InputNotebook. This button is equivalent to *Mathematica*'s Edit ▹ Check Spelling... menu item.

The **Place Image** button opens up a dialog that allows you to choose an image file on file system. This image file is read and its data is converted to a bitmap that is placed at the current location in the InputNotebook.

Opens a dialog that allows you to chose a file. Once chosen, a button will be pasted at the current insertion point in the InputNotebook. When this button is clicked on, it will open up the chosen file in its default application. If anything is selected in the InputNotebook then that material will be replaced by the resulting button.

Opens a dialog to chose a file. The file path of that file is pasted at the current insertion point in the InputNotebook. If anything is selected in the InputNotebook then that material will be replaced by the file path. This button is equivalent to *Mathematica*'s Input ▹ Get File Path... menu item.

Opens the **Remove Options** dialog to act on the current selection in the InputNotebook. This button is equivalent to *Mathematica*'s Format ▹ Remove Options... menu item. In *Mathematica* Version 6.x it reads **Clear Formatting** and corresponds to the Format ▹ Clear Formatting... menu item.

ToDos Palette

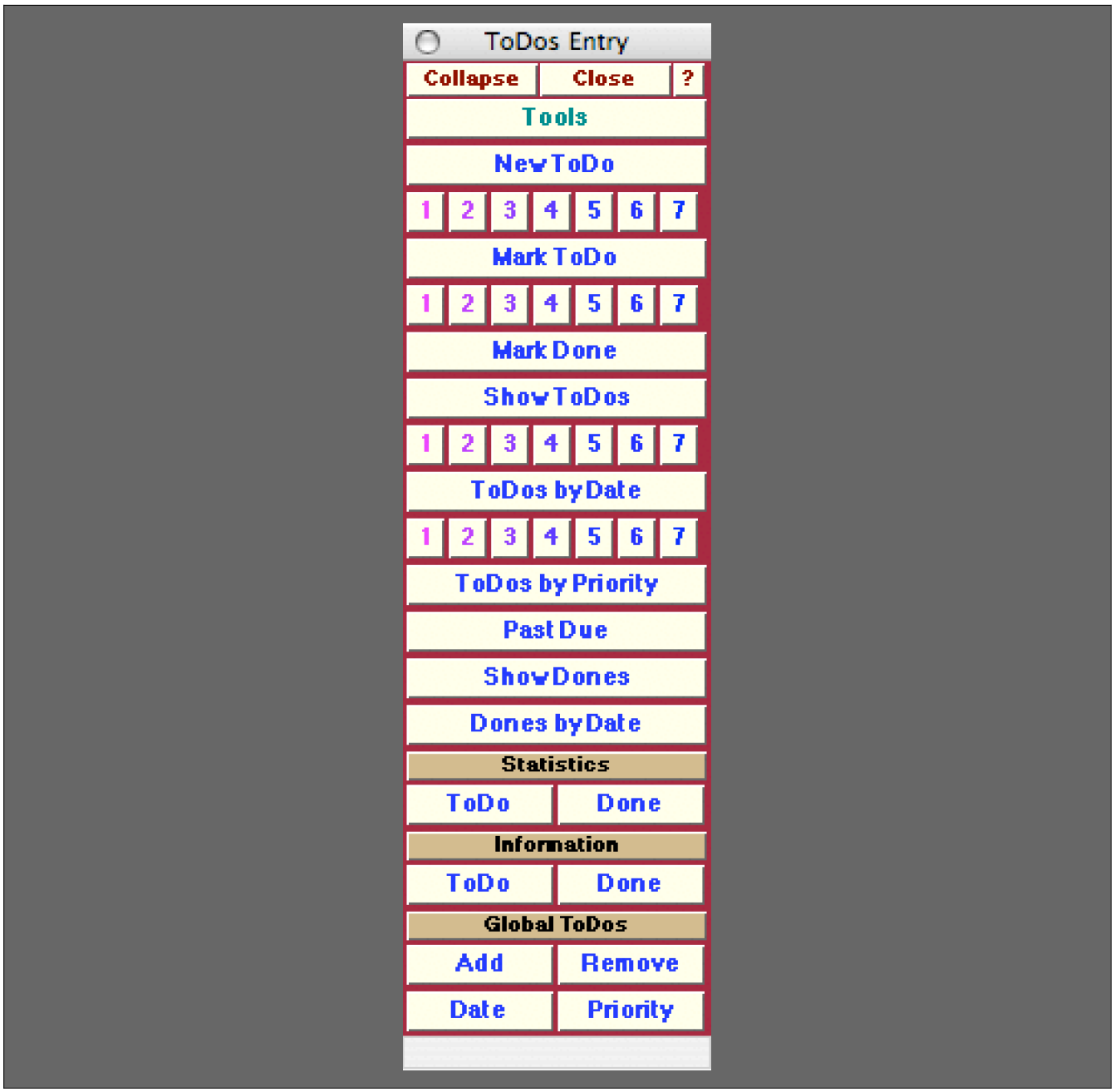
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **Needs["Diary`Diary`"]**, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the ◂ button to the left of **ToDos** on the WorkLife Tools Palette, you will open the ToDos Palette. You can also open the ToDos Palette by executing:

```
ToDosPalette[];
```

or by clicking the ToDos button on the All Palettes Palette.



The ToDos Entry Palette

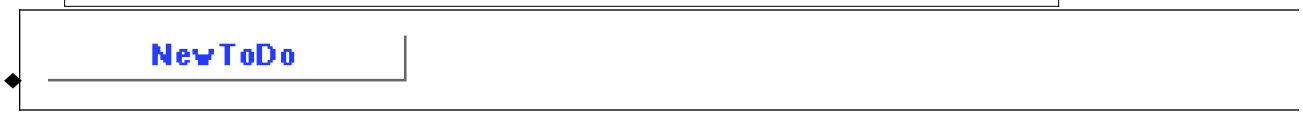
ToDos in A WorkLife FrameWork™™ are ways of organizing material within a Diary that you want to be able to easily access and act upon in various ways. They can be used in a conventional way as prioritized reminders of things that need to be addresses. They can also be used as a way to keep a variety of lists since the "priority" of a ToDo is simply a tag associated with it. Thus one "priority" might actually be a list type, such as a student's name or a particular project. Although you can certainly use them for the purpose, ToDos in A WorkLife FrameWork™™ are not narrowly intended to replace the functionality of a personal organizer.

In effect, the ToDos in A WorkLife FrameWork™ are a specialized class of tagged cells. For some of the tasks that you might make use of ToDos, you could also define special tags and make use of the **TaggingPalette**. However, ToDos are much more intimately integrated into the structure of a Diary and provide added functionality beyond tagging. Such as

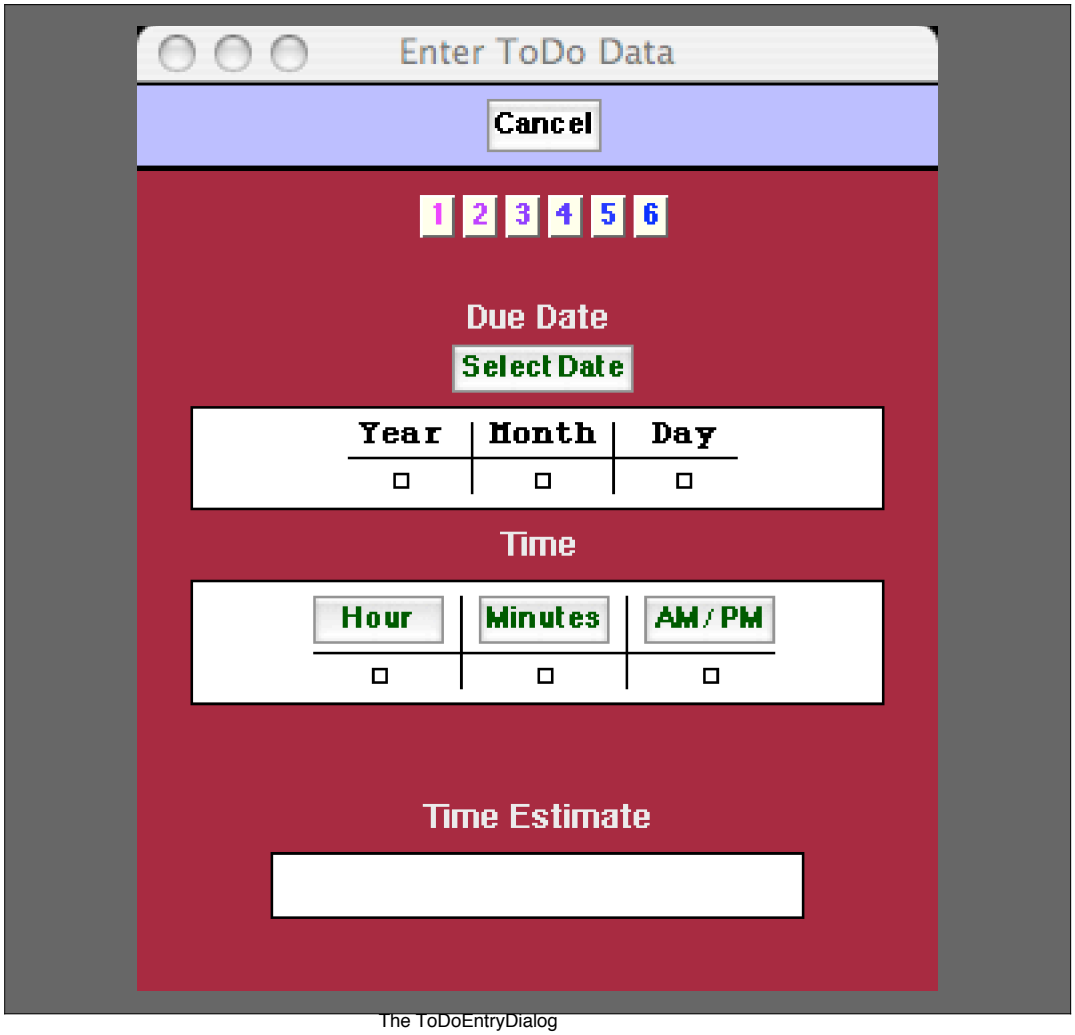
associating a "due date" and a "time estimate" for the ToDo and to retain informatoin on the ToDo even after it has been Marked "Done."

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**



The **New ToDo** button opens the `ToDoEntryDialog` shown below. When filled out and one of the buttons labeled **1, 2, 3, 4, 5, 6** is clicked a new ToDo will be created at the end of the current Diary notebook with the indicated "priority." The button **Select Date** opens a calendar dialog that allows you to chose a specific **Due Date** for the ToDo. The buttons **Hour**, **Minutes** and **AM/PM** each open drop-down menus to allow you to chose the time of day the ToDo is Due. The text field labeled **Time Estimate** allows you to input how long you expect the ToDo to take. All of these entries are optional.



The ToDoEntryDialog

- 1 2 3 4 5 6

Each of this set of buttons creates a new ToDo with the indicated "priority" in the current Diary notebook.

- Mark ToDo

This button opens up a **Mark ToDo** dialog similar to the `ToDoEntryDialog` above. This is used to either mark a cell that is not a ToDo cell as one with the specified properties, or to change the properties of an already existing ToDO cell.

- 1 2 3 4 5 6

This set of buttons marks an existing Text Cell as a ToDo with the indicated "priority", or changes an existing ToDo cell's "priority" to the indicated "priority" in the current Diary.

The **Mark Done** button marks the currently selected ToDo in the current Diary notebook as "Done."

The **Show ToDos** button opens up a new notebook containing all of the ToDos from the current Diary.

Each of this set of buttons opens up a new notebook containing all of the ToDos from the current Diary with the indicated priority..

The **ToDos by Date** button opens up a new notebook containing all of the ToDos from the current Diary sorted by the date that they were created.

Each of this set of buttons opens up a new notebook containing all of the ToDos from the current Diary with the indicated priority sorted by the date that they were created.

The **ToDos by Type** button opens up a new notebook containing all of the ToDos from the current Diary sorted by the their priority.

The **Past Due** button opens up a new notebook containing all of the ToDos from the current Diary that have a `Due Date` that is earlier than the current time.

The **Show Dones** button opens up a new notebook containing all of the Dones from the current Diary.

The **Dones by Date** button opens up a new notebook containing all of the Dones from the current Diary sorted by the date that they were created.

This is a heading button for the pair of buttons that follow.

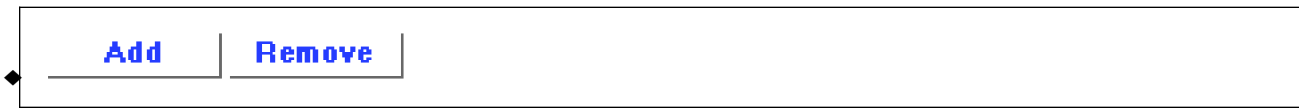
The **ToDo** and the **Done** Statistics buttons open up a new window with a bar graph showing the statistics of the ToDos and the Dones in the current Diary.

This is a heading button for the pair of buttons that follow.

The **ToDo** and the **Done** Information buttons open up a small new window with information on the selected ToDo or Done in the current Notebook. The information includes the date that the ToDo or Done was created, its Priority, and other information specific to the particular button chosen.

This is a heading button for two sets of buttons that follow which have to do with **Global ToDos**.

While many ToDos are in fact associated with a single Diary, you can also keep a list of "Global ToDos" which are known to A WorkLife FrameWork™ as a whole and accessable independent of what Diary is specified as your current Diary. These Global ToDos act in effect as a way to access certain information from Diaries without needing to have any of those Diaries explicitly be open.



The **Add** and the **Remove** buttons respectively add and remove the currently selected ToDo from the **Global ToDos** database. Those that are not in the global ToDos database are local to the Diary that they appear in. In a Diary a ToDo will have a cell dingbat that starts with the light bulb character: 💡 . If the ToDo is also in the **Global ToDos** database then the ToDo will start with a script G and the light bulb character: 𝒢💡 .

The **Date** button displays a notebook containing all of the **Global ToDos** sorted according to the date that they were originally marked as a ToDo. The **Priority** button displays a notebook containing all of the **Global ToDos** sorted according to their priority that they were originally marked as a ToDo.

Computations Palette

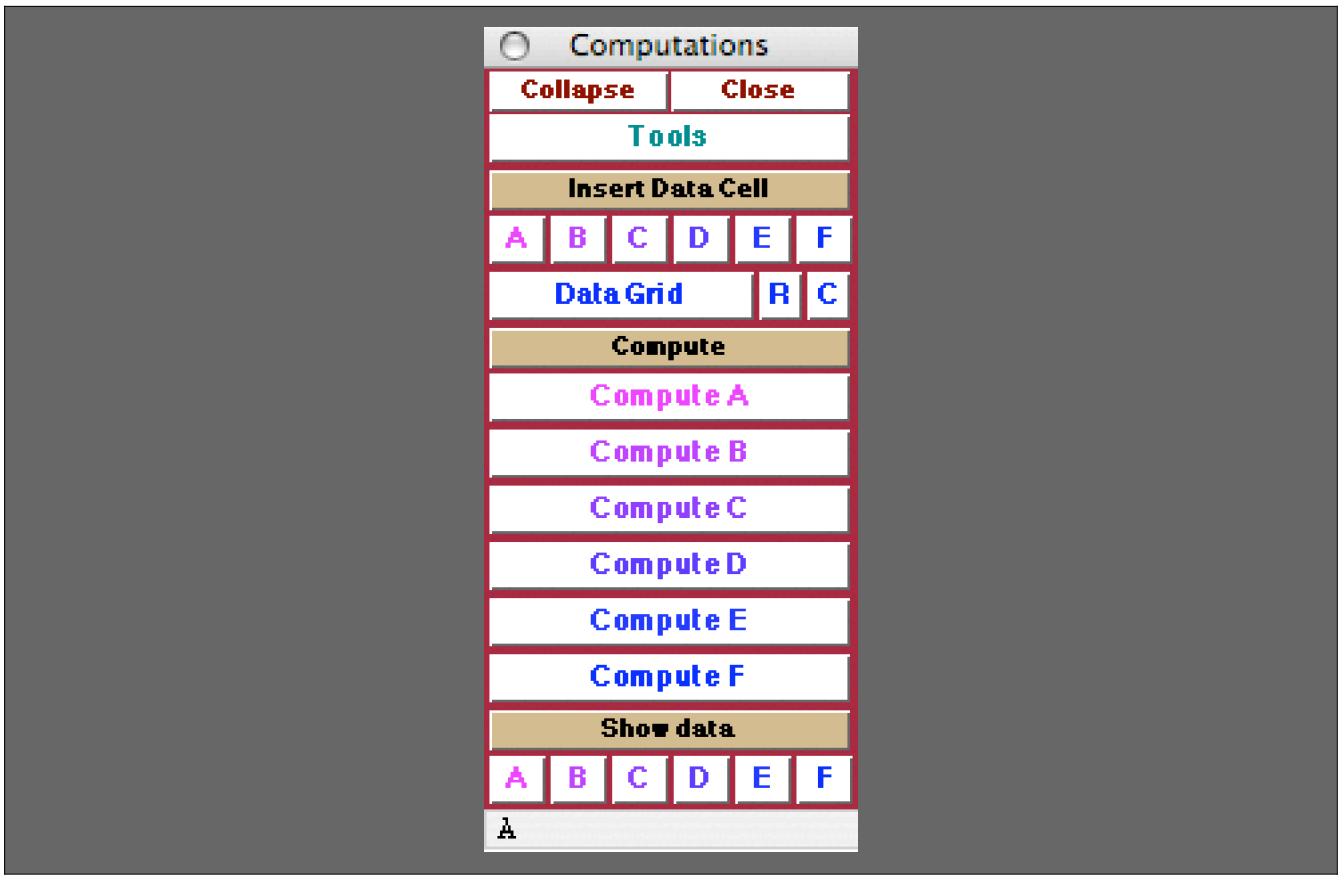
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the ◂ button to the left of **Computations** on the WorkLife Tools Palette, you will open the Computation Palette. You can also open the Computation Palette by executing:

```
ComputationPalette[];
```

or by clicking the Computations button on the All Palettes Palette.



The Computation Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

- **Insert Data Cell**

This is a heading button for the two sets of buttons that follow which have to do with adding data cells to the current Diary notebook.

- A B C D E F

Clicking on any of these buttons inserts a data cell of the indicated label at the end of the current Diary notebook.

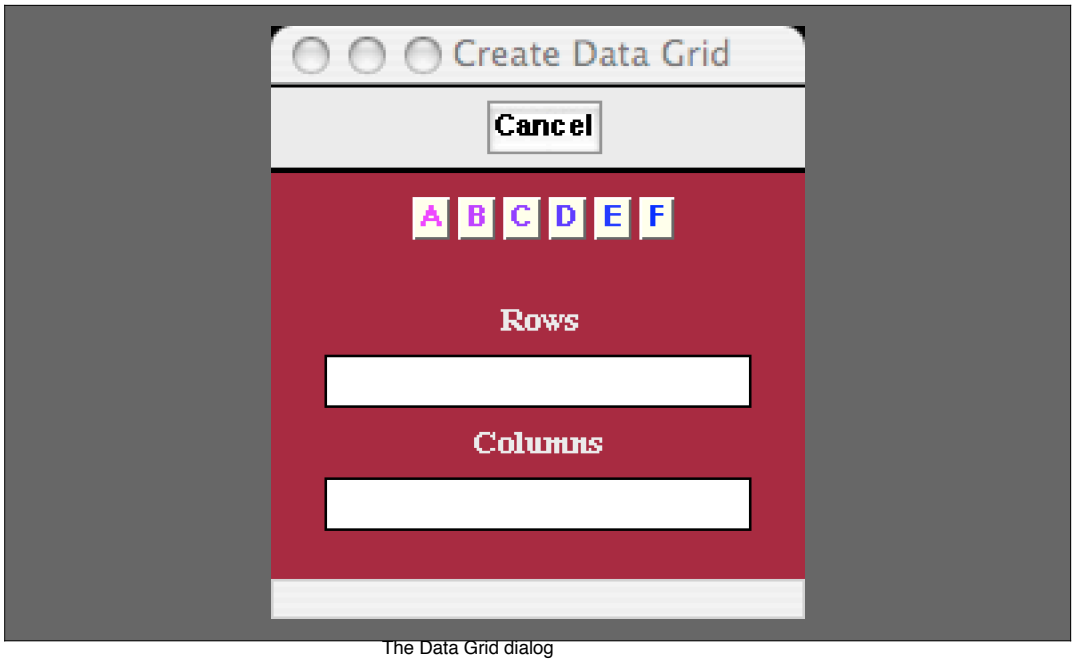
A data cell is formatted like the following (for the case of the label **A**):

A □

where the place holder, □, is ready for the insertion of data.

Data Grid | R | C

The **Data Grid** button opens up a dialog that lets you choose the dimensions of a data grid to be placed as a data cell at the end of the current Diary notebook. The **Data Grid** dialog looks like:



The Data Grid dialog

In the **Data Grid** dialog the number of rows and columns (each an integer) are input and then by clicking on one of the buttons labeled **A**, **B**, **C**, **D**, **E**, **F**, a blank data grid is placed at the end of the current Diary notebook as a computation cell of the chosen label.

The buttons **R** and **C**, when clicked, each add respectively one row or one column to the data grid at the insertion point in the data grid.

Compute

This is a heading button for the buttons that follow which have to do with computations data cells to the current Diary notebook.

Note that the names of these cells are the default generic ones. On a Diary by Diary basis these names can be changed by defining values of the function `ComputationCellOperatorName`. This is illustrated below.

Compute A

Clicking on this button causes the computation defined by `ComputationCellOperator["A"]` to be performed on the data contained in computation cells with label "**A**" in the current Diary notebook. The default value of each of the `ComputationCellOperators` is just `(#&)`, which returns the data contained in the computation cells with the given label.

If we had made the definition

```
ComputationCellOperatorName["A"] = "Total hours";
```

then the button would look like

And an example of a function that would add together the contents of the A data cells would be

```
ComputationCellOperator["A"] = 1/2 Tr[#1[[All, 1]]] &;
```

Causes the computation defined by `ComputationCellOperator["B"]` to be performed on the data contained in computation cells with label "**B**" in the current Diary notebook. See the discussion above for the first Computation button.

Causes the computation defined by `ComputationCellOperator["C"]` to be performed on the data contained in computation cells with label "**C**" in the current Diary notebook. See the discussion above for the first Computation button.

Causes the computation defined by `ComputationCellOperator["D"]` to be performed on the data contained in computation cells with label "**D**" in the current Diary notebook. See the discussion above for the first Computation button.

Causes the computation defined by `ComputationCellOperator["E"]` to be performed on the data contained in computation cells with label "**E**" in the current Diary notebook. See the discussion above for the first Computation button.

Causes the computation defined by `ComputationCellOperator["F"]` to be performed on the data contained in computation cells with label "**F**" in the current Diary notebook. See the discussion above for the first Computation button.

This is a heading button for the buttons that follow which have to do with computations data cells to the current Diary notebook.

Each of these buttons opens up a new notebook containing the data cells of the indicated label from the current Diary notebook.

Email Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the ◂ button to the left of **Email** on the WorkLife Tools Palette, you will open the Email Palette. You can also open the Email Palette by executing:

```
EmailPalette[];
```

or by clicking the Email button on the All Palettes Palette.

The Email Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

The following buttons are for the creation and manipulation of emails within Diaries and Notebooks.

The **New Email** button creates a new email at the end of the current Diary notebook. The button with the "⊙" on it places the email near the current insertion point in the current Diary notebook.

The parameter `$EmailInCurrentDiary` *determines whether a new email is placed in the current Diary notebook or in the current InputNotebook. The default behavior is* `$EmailInCurrentDiary=True` *which causes all new emails to be created in the current Diary notebook.*

Adds a **Body Cell** to the current email. The insertion point must be within an email. The button with the "⊙" on it places the body cell below the current insertion point in the current Diary notebook. Body cells are where the text of an email is composed. There may be any number of body cells.

Adds a **Notes Cell** to the current email. The insertion point must be within an email. The button with the "⊙" on it places the notes cell below the current insertion point in the current Diary notebook. Notes Cells are for keeping notes within an email that will not be sent with the email.

The **Add CC** button adds a CC field to the email form and the **Delete** button deletes any CC field that is currently in the email form.

The **Add BCC** button adds a BCC field to the email form and the **Delete** button deletes any BCC field that is currently in the email form.

Adds **Separator Text** to the current email. The insertion point must be within an email. The separator text that is added is given by the function `EmailSeparatorString`. This function, in turn, depends on the following parameters:

	Definition
`Character`	Character that is repeated in the separator string
`StartString`	Initial character string for the separator string
`EndString`	Ending character string for the separator string
`StringLength`	The number of times that $EmailSeparatorCharacter is repeated between $EmailSeparatorStartString and $EmailSeparatorEndString

The default value of `EmailSeparatorString[]` is `"**********"`.

Open Client

Opens the user's default email client.

Remove Email

Removes the email. The insertion point must be within the email. Removed emails are copied to the clipboard. The clipboard will be overwritten whenever something else is copied to it.

Note if you accidentally remove an email and wish to recover it you should ***immediately*** *paste it back into the notebook. A number of functions in this package make use of the clipboard and may overwrite the temporarily copied email. Also the Mathematica* Edit ▹ Undo *Menu command (or its keyboard equivalent) will remove the email from the clipboard.*

Show Emails

Opens a notebook containing all of the emails that are in the current InputNotebook.

Find

Opens a dialog that allows you to find an email based on its locator. When an email is sent from A WorkLife Frame-Work the email that is passed to your email client will contain a code at its end. This code—the email locator—is a combination of two integers linked by the characters ELS. This code, when placed in the dialog that is opened when this button is clicked, will locate the Diary/Notebook that the email was sent from, and highlight the location of the email.

Show Find Code

Opens a notebook that has the alphanumeric string that you would use in the `EmailFindDialog` that is used by the preceeding **Find** button.

The following buttons work with mailboxes in the MBOX format. Email clients that support the MBOX format include Thunderbird, Eudora, Mozilla, Netscape, and others. Other Email clients generally permit export of email into MBOX format.

This opens a dialog that allows you to specify those MBOX files on your system you want to include in creating a network visualization of emails. It generates a network graph of the network of emails in the supplied mailbox files and displays this network graphic in a new Notebook window.

This opens a dialog that allows you to specify those MBOX files you wish to extract emal addresses from. All email addresses are extracted from the files and are presented as a list in a new Notebook window. "

Essay Palette

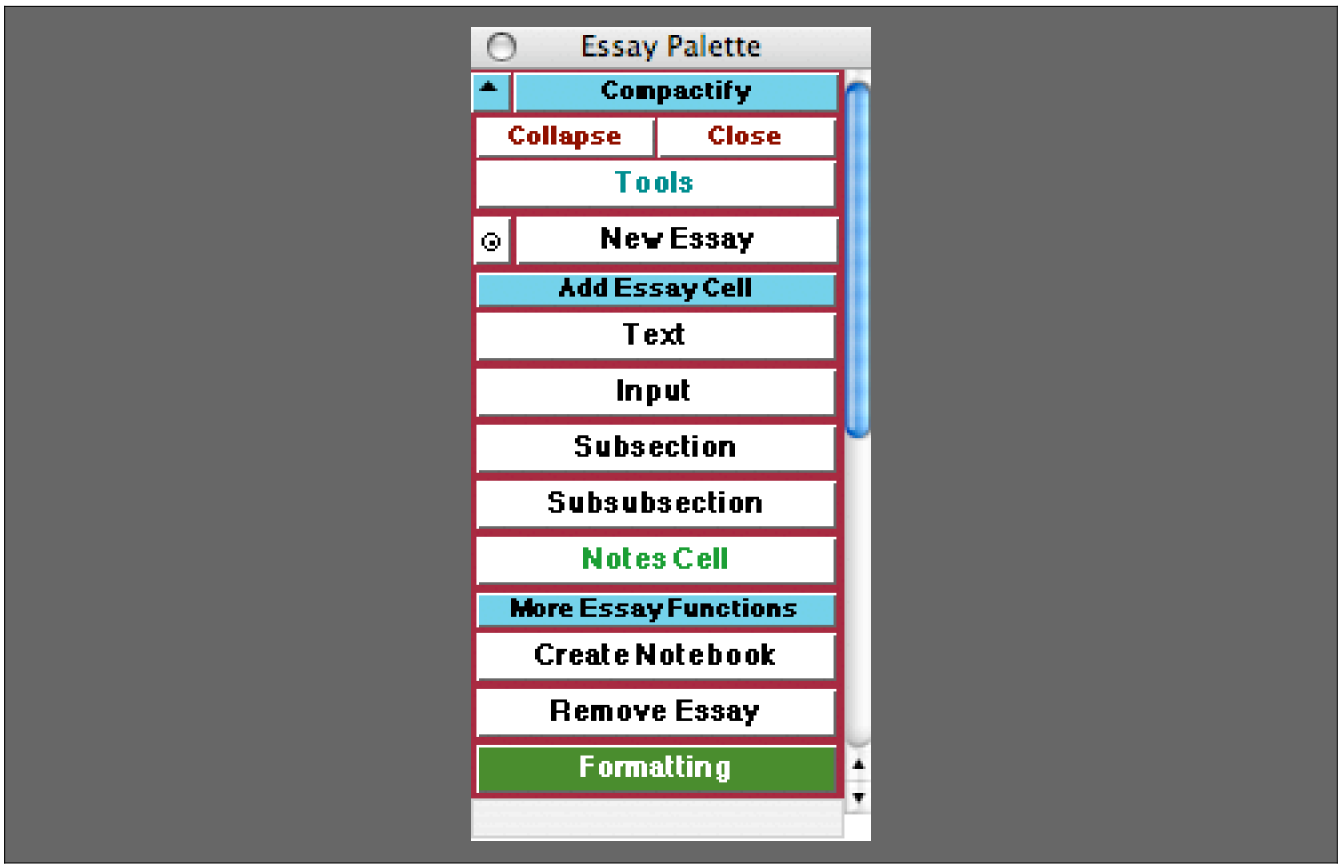
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the ◂ button to the left of **Essay** on the WorkLife Tools Palette, you will open the Essay Palette. You can also open the Essay Palette by executing:

```
EssayPalette[];
```

or by clicking the Essay button on the All Palettes Palette.



The Essay Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

New Essay

The **New Essay** button creates a new essay at the end of the current Diary notebook. The button with the "⊙" on it places the essay near the current insertion point in the current Diary notebook.

The parameter `$EssayInCurrentDiary` *determines whether a new essay is placed in the current Diary notebook or in the current InputNotebook. The default behavior is* `$EssayInCurrentDiary=True` *which causes all new essays to be created in the current Diary notebook.*

Opens and closes the **Add Essay Cell** sub-palette which includes the five buttons that follow.

Adds a new **Text** cell within the essay. The insertion point must be within the essay. The new **Text** cell is created below the insertion point.

Adds a new **Input** cell within the essay. The insertion point must be within the essay. The new **Input** cell is created below the insertion point.

Adds a new **Subsection** cell within the essay. The insertion point must be within the essay. The new **Subsection** cell is created below the insertion point.

Adds a new **Subsubsection** cell within the essay. The insertion point must be within the essay. The new **Subsubsection** cell is created below the insertion point.

Adds a new **Notes Cell** within the essay. The insertion point must be within the essay. The new **Notes** cell is created below the insertion point. When the **Create Notebook** button (described below) is used, **Notes Cell**s are not included in the resulting notebook.

Additional CellStyles can be added to the Essay Palette at this location by using the function `AddCellStyles ToEssayPalette`.

Opens and closes the **More Essay Functions** sub-palette which includes the two buttons that follow.

Create Notebook

This button creates a new notebook containing just the content of the essay. The insertion point must be within the essay. The resulting notebook does not contain any of the essay's **Notes Cell**s

Remove Essay

Removes the essay. The insertion point must be within the essay. Removed essays are copied to the clipboard. The clipboard will be overwritten whenever something else is copied to it.

*Note if you accidentally remove an essay and wish to recover it you should **immediately** past it back into the notebook. A number of functions in this package make use of the clipboard and may overwrite the temporarily copied essay. Also the Mathematica* Edit ▹ Undo *Menu command (or its keyboard equivalent) will remove the essay from the clipboard.*

Formatting

Opens the **Formatting** Palette. The Formatting palette can also be opened programmatically by executing the command,

```
FormattingPalette[];
```

All Palettes Palette

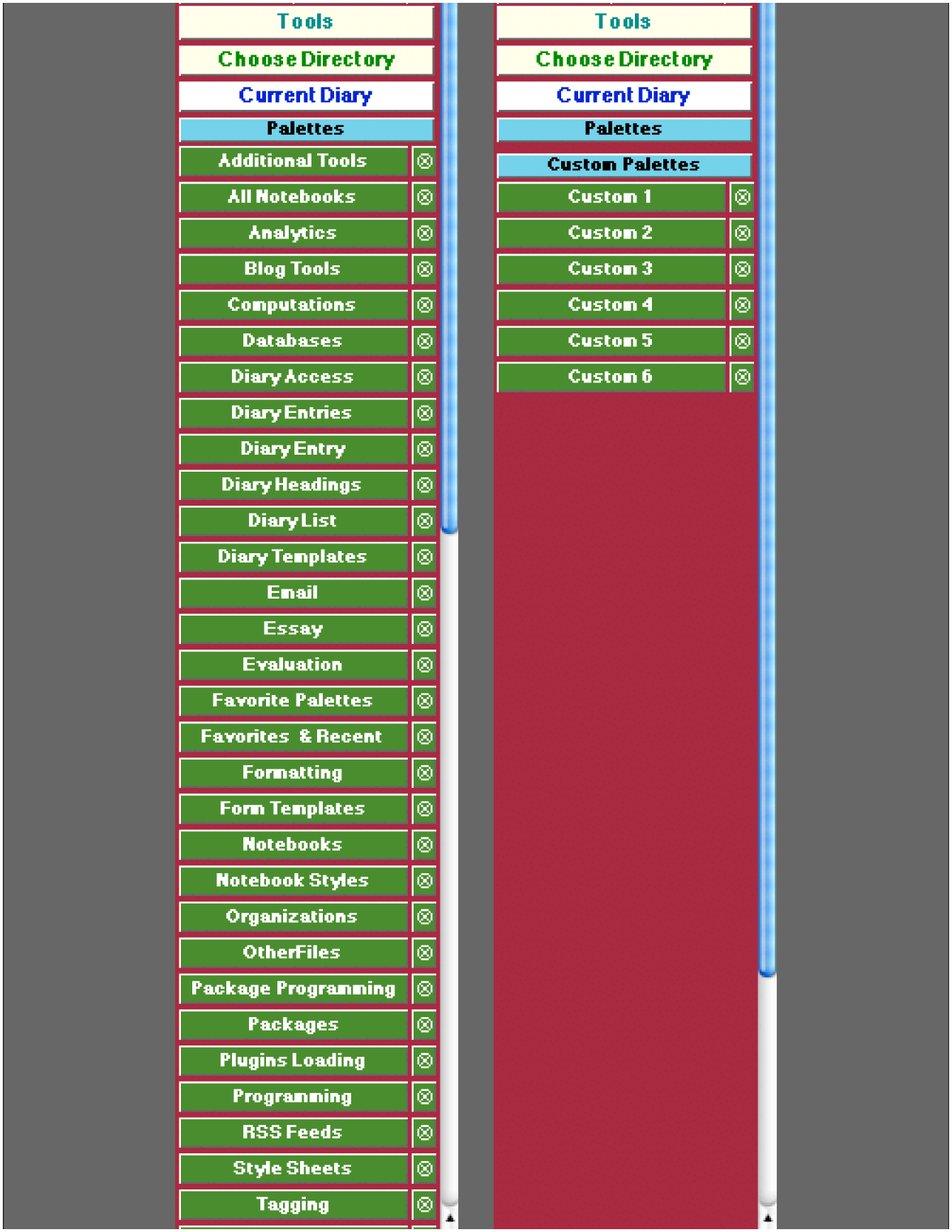
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the ◂ button to the left of **All Palettes** on the WorkLife Tools Palette, you will open the All Palettes Palette. You can also open the All Palettes Palette by executing:

```
AllPalettesPalette[];
```

The All Palettes Palette



The All Palettes Palette 1) showing the non-Custom Palettes and 2) partially compactified to show only the Custom Palettes

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

- **Choose Directory**

This opens the Directory Browser dialog window. In this window you can navigate your computer's directories and choose the one that you wish to open a Diary from. The resulting directory is then the value of `$DefaultDiaryDirectory`.

- **Current Diary**

This opens the **Current Diary** if it has been chosen. If not, an error message is generated.

If the current Diary is open then its window is brought to the front. If it is not opened then it is opened and any Default Code Cells are executed.

- **Palettes**

Opens and closes the **Palettes** sub-palette which lists all of the Palettes (described following) in the WorkLife Frame-Work™ Package aside from the Custom Palettes.

- **Additional Tools** ⊗

Opens the **Additional Tools** Palette. This can also be opened programmatically by executing:

```
AdditionalToolsPalette[];
```

The ⊗ button closes the Palette if it is currently open.

- **All Notebooks** ⊗

Opens the **All Notebooks** Palette. This can also be opened programmatically by executing:

```
AllOpenNotebooksPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Analytics** Palette. This can also be opened programmatically by executing:

```
AnalyticsPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Blogging** Palette. This can also be opened programmatically by executing:

```
BlogPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Computations** Palette. This can also be opened programmatically by executing:

```
ComputationPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Databases** Palette. This can also be opened programmatically by executing:

```
DatabasesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Diary Access** Palette. This can also be opened programmatically by executing:

```
DiaryAccessPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Diary Entry** Palette. This can also be opened programmatically by executing:

```
DiaryEntryPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Diary Entries** Palette. This can also be opened programmatically by executing:

```
DiaryEntriesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Diary Heading** Palette. This can also be opened programmatically by executing:

```
DiaryHeadingsPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Diary List** Palette. This can also be opened programmatically by executing:

```
DiaryListPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Diary Templates** Palette. This can also be opened programmatically by executing:

```
DiaryTemplatesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Email** Palette. This can also be opened programmatically by executing:

```
EmailPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Essay** Palette. This can also be opened programmatically by executing:

```
EssayPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Evaluation** Palette. This can also be opened programmatically by executing:

```
EvaluationPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Favorite Palettes** Palette. This can also be opened programmatically by executing:

```
FavoritePalettesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Favorites & Recent** Palette. This can also be opened programmatically by executing:

```
FavoritesAndRecentPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Formatting** Palette. This can also be opened programmatically by executing:

```
FormattingPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Form Templates** Palette. This can also be opened programmatically by executing:

```
FormTemplatesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Notebooks** Palette. This can also be opened programmatically by executing:

```
NotebooksPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Notebook Styles** Palette. This can also be opened programmatically by executing:

```
NotebookStylesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Organizations** Palette. This can also be opened programmatically by executing:

```
OrganizationsPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **OtherFiles** Palette. This can also be opened programmatically by executing:

```
OtherFilesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Package Programming** Palette. This can also be opened programmatically by executing:

```
PackageProgrammingPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Packages** Palette. This can also be opened programmatically by executing:

```
PackagesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Plug-ins Loading** Palette. This can also be opened programmatically by executing:

```
PluginsLoadingPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Programming** Palette. This can also be opened programmatically by executing:

```
ProgrammingPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Ross Feeds** Palette. This can also be opened programmatically by executing:

```
RSSFeedsPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Style Sheets** Palette. This can also be opened programmatically by executing:

```
StyleSheetsPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Tagging** Palette. This can also be opened programmatically by executing:

```
TaggingPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **ToDos** Palette. This can also be opened programmatically by executing:

```
ToDosPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Web Search** Palette. This can also be opened programmatically by executing:

```
WebSearchPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **WorkFlows** Palette. This can also be opened programmatically by executing:

```
WebSearchPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens and closes the **Custom Palettes** sub-palette which lists all of the Custom Palettes that the you can define in the WorkLife FrameWork™ Package.

Opens the **Custom1** Palette. This can also be opened programmatically by executing:

```
Custom1Palette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Custom2** Palette. This can also be opened programmatically by executing:

```
Custom2Palette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Custom3** Palette. This can also be opened programmatically by executing:

```
Custom3Palette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Custom3** Palette. This can also be opened programmatically by executing:

```
Custom3Palette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Custom5** Palette. This can also be opened programmatically by executing:

```
Custom5Palette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Custom6** Palette. This can also be opened programmatically by executing:

```
Custom6Palette[];
```

The ⊗ button closes the Palette if it is currently open.

Additional Tools Palette

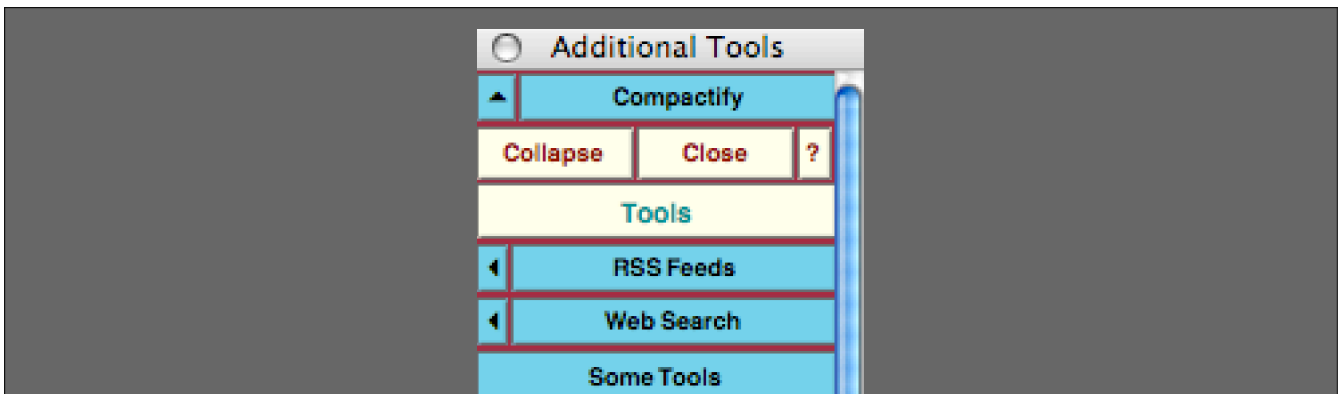
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the ◂ button to the left of **Additional Tools** on the WorkLife Tools Palette, you will open the Additional Tools Palette. You can also open the Additional Tools Palette by executing:

```
AdditionalToolsPalette[];
```

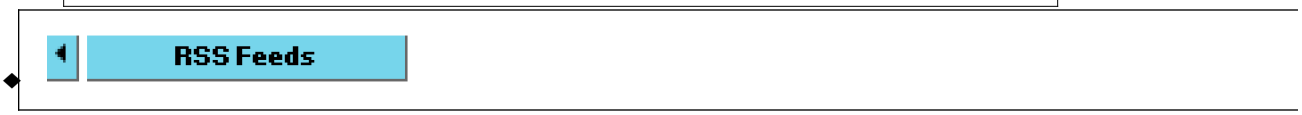
or by clicking the **Additional Tools** button on the **All Palettes Palette**.



The Additional Tools Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**



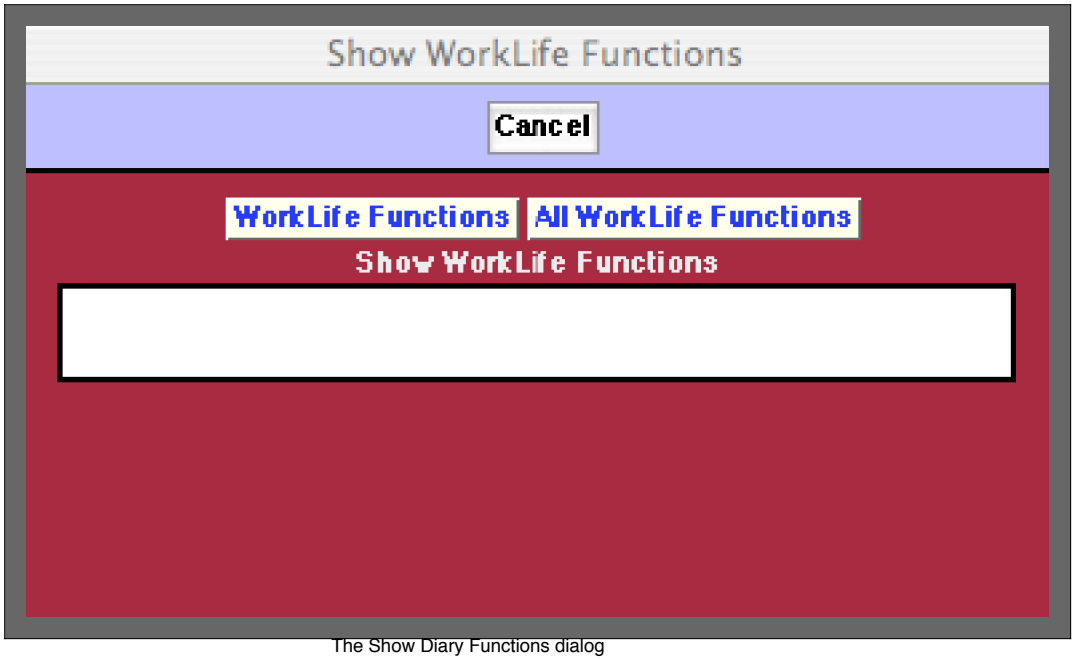
Opens and closes the **RSS Feeds** sub-palette. See the documentation for the RSS Feeds Palette for details. By clicking on the on the ◂ button the RSS Feeds Palette will open.

Opens and closes the **Web Search** sub-palette. See the documentation for the Web Search Palette for details. By clicking on the on the ◂ button the Web Search Palette will open.

Opens and closes the **Some Tools** sub-palette which includes the five buttons that follow.

The **WorkLife Functions** button opens up the Show WorkLife Functions dialog. Use this dialog to quickly find the properties of specific functions in this package.

This dialog looks like:



The Show Diary Functions dialog

To make use of this dialog type in a partial name of a function in this package, then click on **WorkLife Functions**. A new window will open up with a list of all of the functions in this package that contain the partial name that was given. Each item in the list is a hyperlink that, when clicked on, gives information on its function.

These two buttons control the "Dashboard" functionality in this package. The **Dashboard** button opens up all of the dashboard elements that the user has set up. The **Toggle** button toggles between making the dashboard elements visible and invisible.

This button opens up *Mathematica*'s **Slide Show** Palette. Note that the "Convert Notebook" button on *Mathematica*'s **Slide Show** Palette replaces the current notebook with the converted one. The following two buttons convert Diaries and NOtebooks to Slide shows; however, they do this in a non-destructive way by creating a distinct slide show notebook while maintaining the original one unchanged.

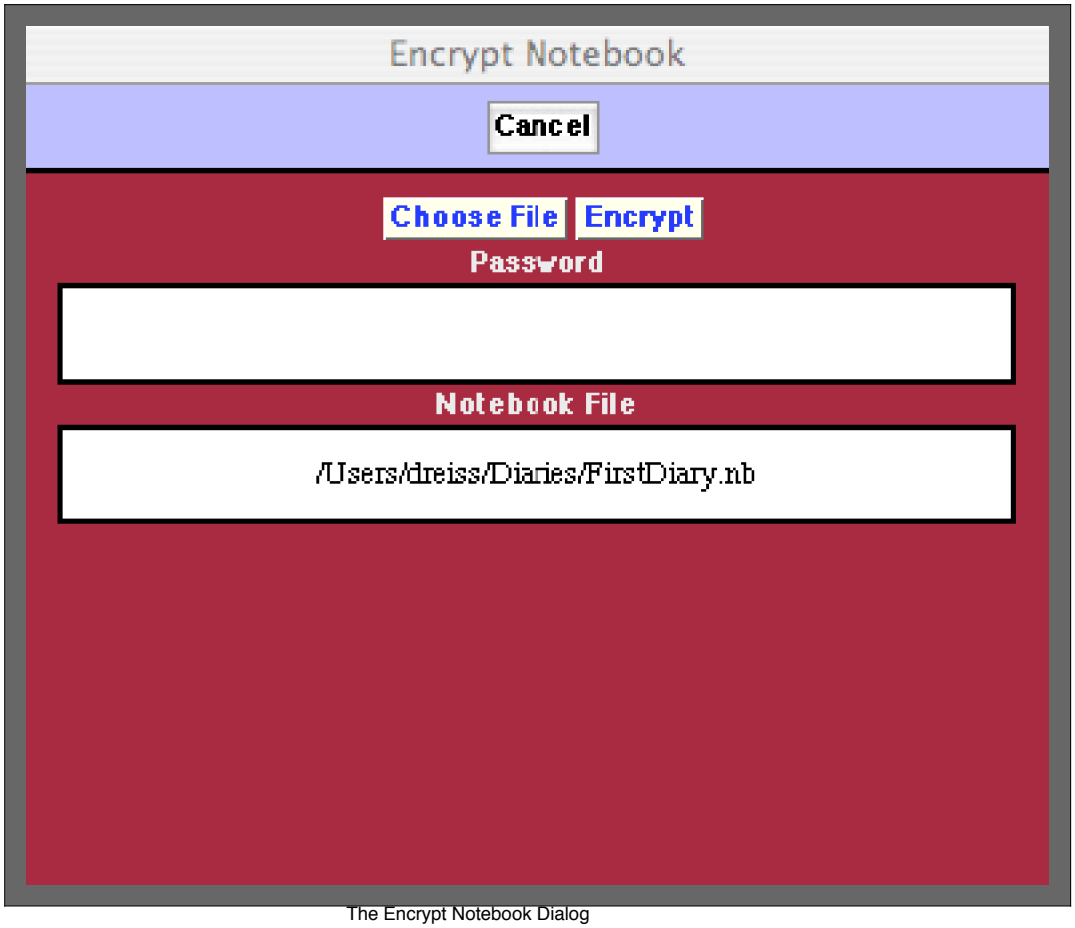
This button converts the current Diary into a slideshow and saves the resulting notebook in the Notebooks subdirectory of the current Diary's directory. The resulting slide show notebook's name is the same as the Diary's name with the characters "SS" appended to the end. If the current Diary is not open it is opened first.

This button converts the current InputNotebook into a slideshow and saves the resulting notebook in the Notebooks subdirectory of the current Diary's directory. The resulting slide show notebook's name is the same as the notebook's name with the characters "SS" appended to the end. If there is no notebook selected then an error message is returned.

The **Backup Notebook** button backs up the current state of the Input Notebook in its directory. An Alert Dialog appears while the backup is in progress.

The **Encrypt Notebook** button opens a dialog that allows you to create an encrypted file that contains a copy of a notebook.

The resulting dialog looks like:



The Encrypt Notebook Dialog

When this dialog opens the full file path for the currently selected InputNotebook appears in its **Notebook File** field. A different notebook can be chosen by clicking the **Choose File** button. To create the encrypted file, a password must be typed into the **Password** field. Then, clicking the Encrypt button creates a file in the same directory as the Notebook File with the suffix .m and the same name as the original file but with the string ENB appended to its end. Thus in the case in the illustration above the encrypted version of the file `/Users/dreiss/Documents/Diaries/First`⁞ `Diary.nb` will be `/Users/dreiss/Documents/Diaries/FirstDiaryENB.m`. The resulting file can be decrypted using the **Decrypt Notebook** button below or by using the function `Decrypt`⁞ `Notebook["file","password"]`.

If you wish to send an encrypted notebook to someone who does not own a copy of A WorkLife FrameWork™, you can send them a piece of *Mathematica* code that will allow them to decrypt the file so long as they have the password that you Encrypted the notebook with. You can generate a Notebook with the required decrypting code with the function `ExportableDecryptingFunction`.

ExportableDecryptingFunction[] opens a notebook with a function DecryptEncryptedNotebook
have access to the WorkLife FrameWork™ Package to decrypt notebooks that have been

Usage Message for `ExportableDecryptingFunction`

When a notebook is encrypted the default behavior is to not *remove the original, so that both an encrypted version and the original version will exist. The function that is used to encrypt a notebook is* `EncryptNote`⁞.

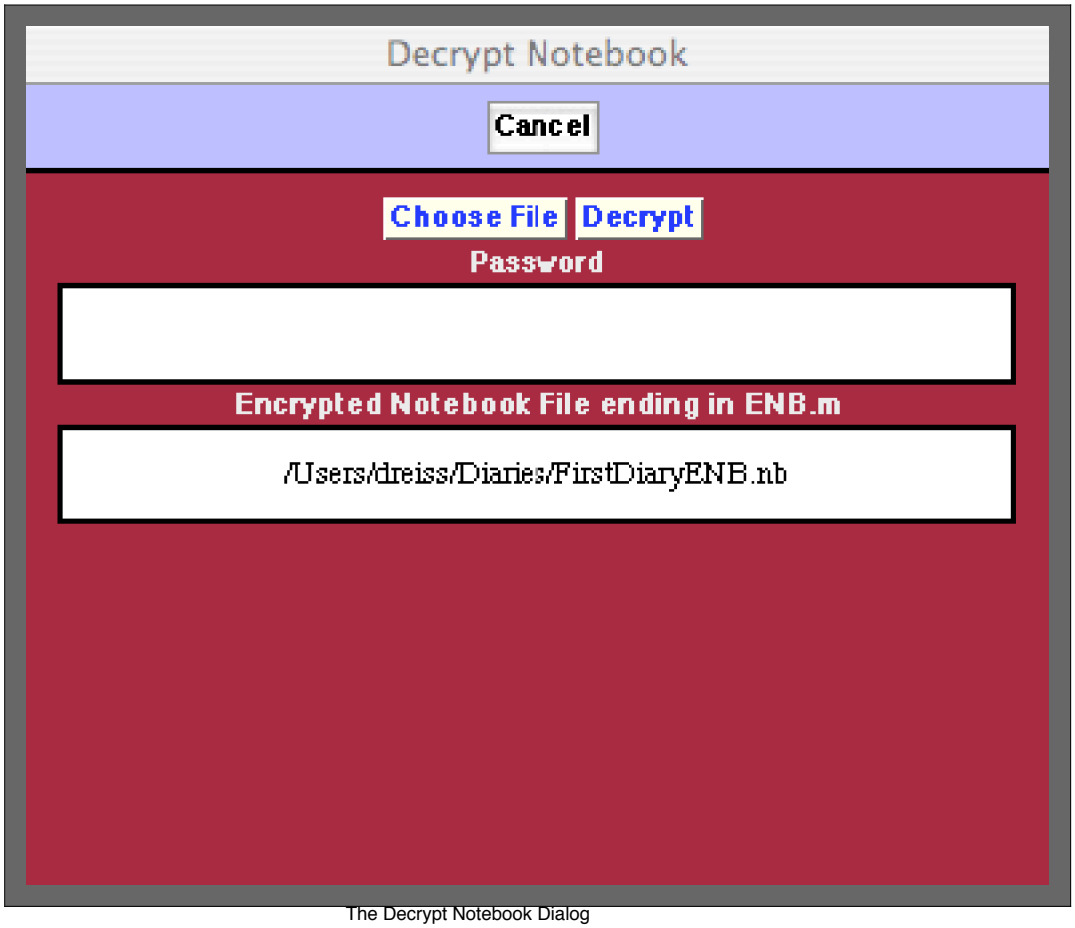
book, *an this function has an option,* DeleteEncryptedOriginal, *which has the default value* DeleteEncryptedOriginal→False.

To Password Protect a Diary or a Notebook you can use the **Password Protect** *Button on the Top Toolbar of the Diary or Notebook. In this case, the give Diary or Notebook will be* replaced *by the Password Protected version, and the original will be removed. (In effect, the* **Password Protect** *Button uses* EncryptNotebook *with* DeleteEncryptedOriginal→True.*) Note however that any backup copies of the Diary or Notebook will not be password protected. So, if you are password protecting a Diary or Notebook in this way for privacy on your own system (rather than for transmitting via unsecure channels it or saving it to another location) then you must remember that your backup versions are not protected and should be Encrypted using* EncryptNotebook *with the option* DeleteEncryptedOriginal→True.

Decrypt Notebook

The **Decrypt Notebook** button opens a dialog that allows you to create an encrypted file that contains a copy of a notebook.

The resulting dialog looks like:



The Decrypt Notebook Dialog

In this image the file Users/dreiss/Documents/Diaries/FirstDiaryENB.m has been chosen. When the proper password is entered into the **Password** field and the **Decrypt** button is clicked a version of the original notebook

is opened. Note that the resulting notebook is not saved to disk unless you explicitly save it.

The **Archive Diary** button opens a dialog asking you whether you want to archive the current Diary. When a Diary is archived all of its content is removed except for any outstanding ToDos. The ToDos are placed in a Section at the beginning of the Diary titled ToDos From Archived Diary. Before this is done the Diary is backed up and a copy of the Diary is also placed in a directory that has the original Diary's name with the string "XV" appended to it.

If the current Diary has previously been archived the **Open Last Archive** button will open the most recent Archive.

Opens a Notebook with information on the **Archives** of the current Diary.

Opens up the **Archives Directory** for your current Diary.

Opens and closes the **Other Tools** sub-palette which includes the ten buttons that follow.

Opens an informational notebook that displays the current values of key **Directories and Files**.

This button opens the directory where the current InputNotebook is located. If this notebook has not yet been saved an error message is returned.

Opens the directory where *Mathematica* keeps user applications. In particular, the WorkLife FrameWork™ package is located in this directory along with its various settings files (and their automatically generated backups).

The **Copy→OtherFiles** button opens up a browsing window that allows you to select a file. A copy of this file will then placed in the OtherFiles subdirectory that is within the Current Diary's directory. If the selected file is a notebook an error message will be returned. For notebooks you should use the **Copy→Notebooks** button below.

The **Copy→Notebooks** button opens up a browsing window that allows you to select a notebook. A copy of this notebook will then placed in the Notebooks subdirectory that is within the Current Diary's directory. If the selected file is not a notebook an error message will be returned. For files other than notebooks you should use the **Copy→OtherFiles** button below.

Clicking on the **Refresh Palettes** button causes all of the currently open Palettes in this package to be refreshed.

More Palettes

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

The following twenty-five Palettes are directly accessible from the top level of the **All Palettes** Palette (and from the **All Palettes** sub-palette on the **WorkLife Tools** Palette) . They are listed in alphabetical order.

All Notebooks Palette

The Palette

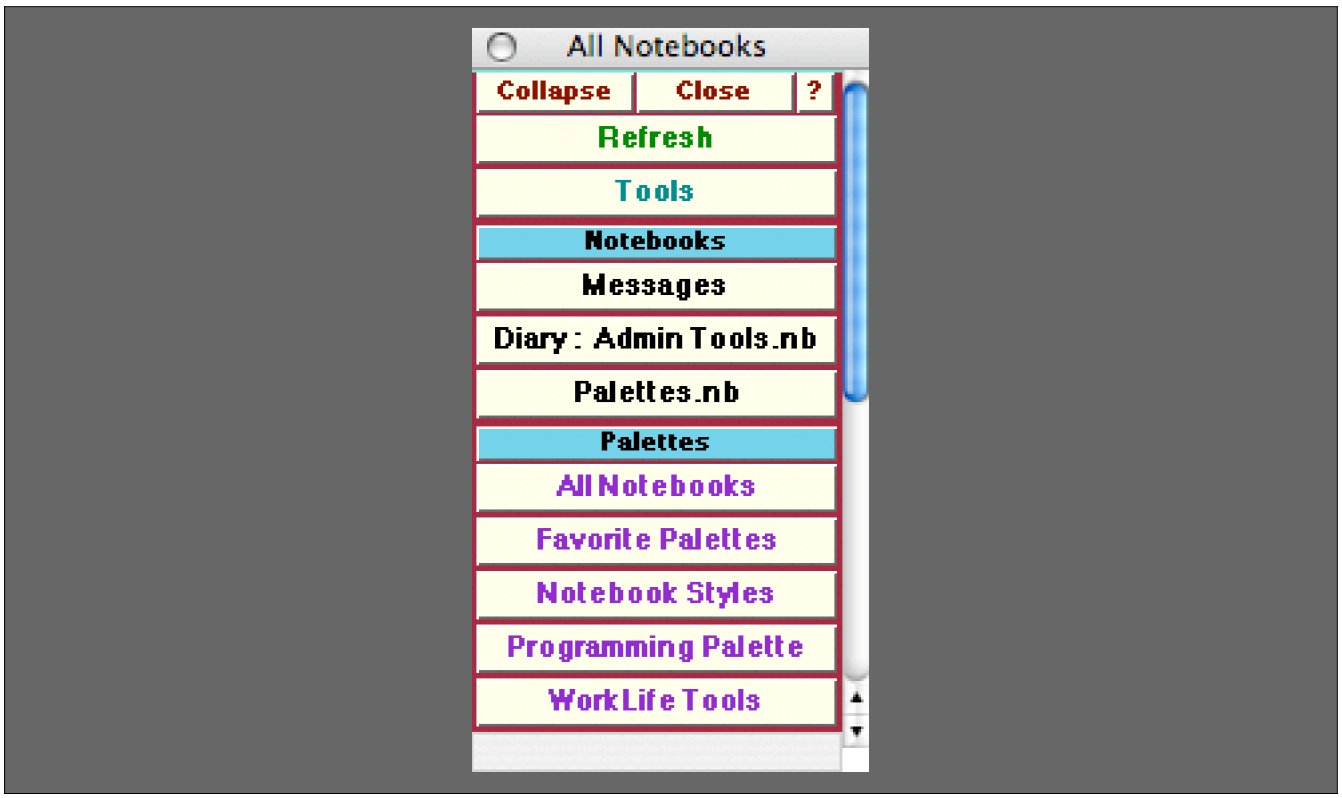
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **All Notebooks** button on the All Palettes Palette, you will open the All Notebooks Palette. You can also open the All Notebooks Palette by executing:

```
AllOpenNotebooksPalette[];
```

The All Notebooks Palette lists all of the notebooks (and Palettes) that are currently open in *Mathematica*'s Front End. Clicking on any of the buttons will bring that Notebook to the front of the other Notebooks.

This Palette lists both `Visible` *and Invisible Notebooks. (An Invisible Notebook is one that has a value of its* `Visible` *option* `Visible→False`*. Note that the Messages Notebook is always present and, when it is not seen it is Invisible.) If the button of a Notebook that is not* `Visible` *is clicked, then that Notebook is made* `Visible` *and it is brought to the front.*



The All Open Notebooks Palette

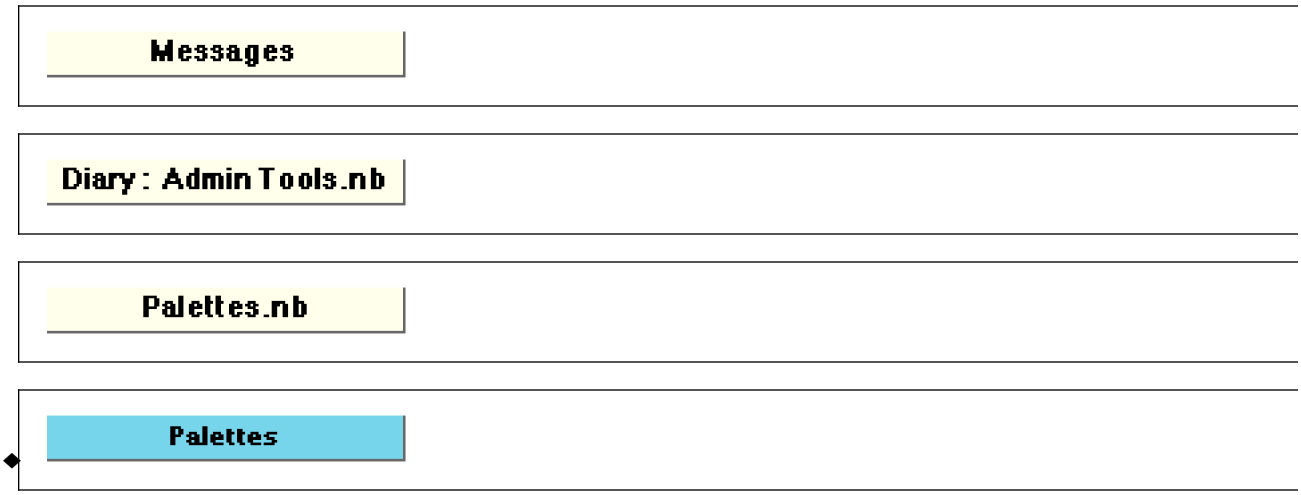
The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Notebooks

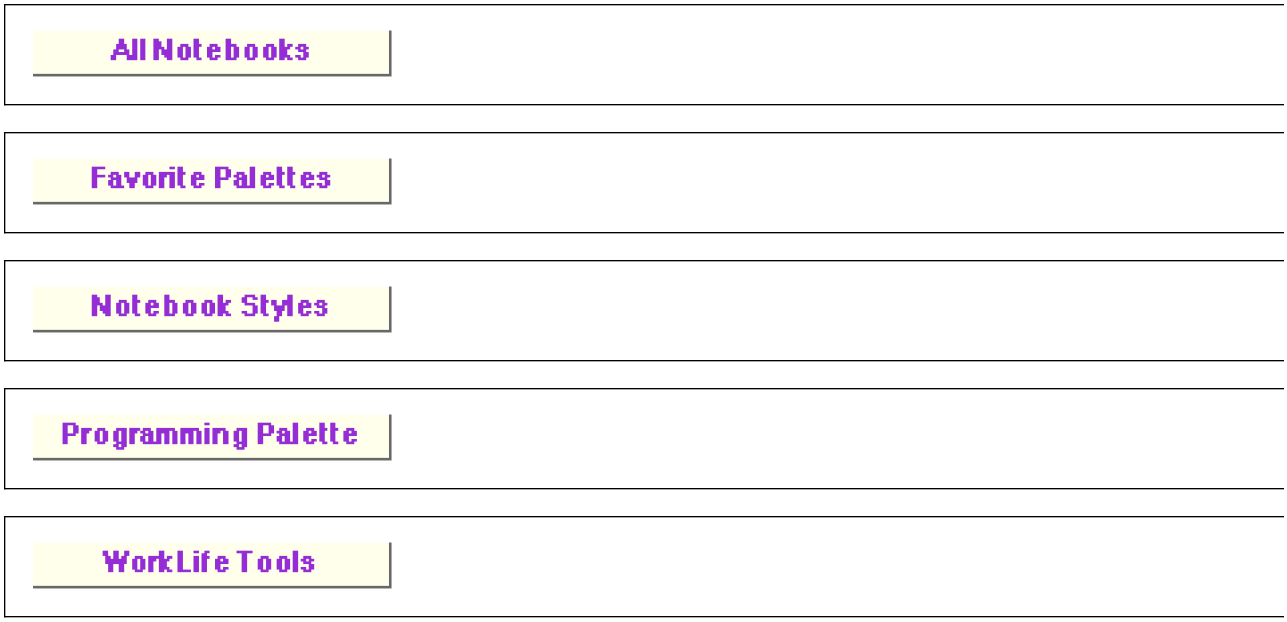
Opens and closes the **Notebooks** sub-palette which includes the buttons that follow it, listing all of the currently open Notebooks in the *Mathematica* front End.

In the example above there are 3 non-Palette Notebooks currently open in the Front End: **Messages**, **Diary: Admin Tools.nb**, and **Diary: Palettes.nb**.



Opens and closes the **Palettes** sub-palette which includes the buttons that follow it, listing all of the currently open Palettes in the *Mathematica* front End.

In the example here there are 5 Palettes currently open in the Front End: **All Notebooks**, **Favorite Palettes**, **Notebook Styles**, **Programming Palette**, and **WorkLife Tools**.



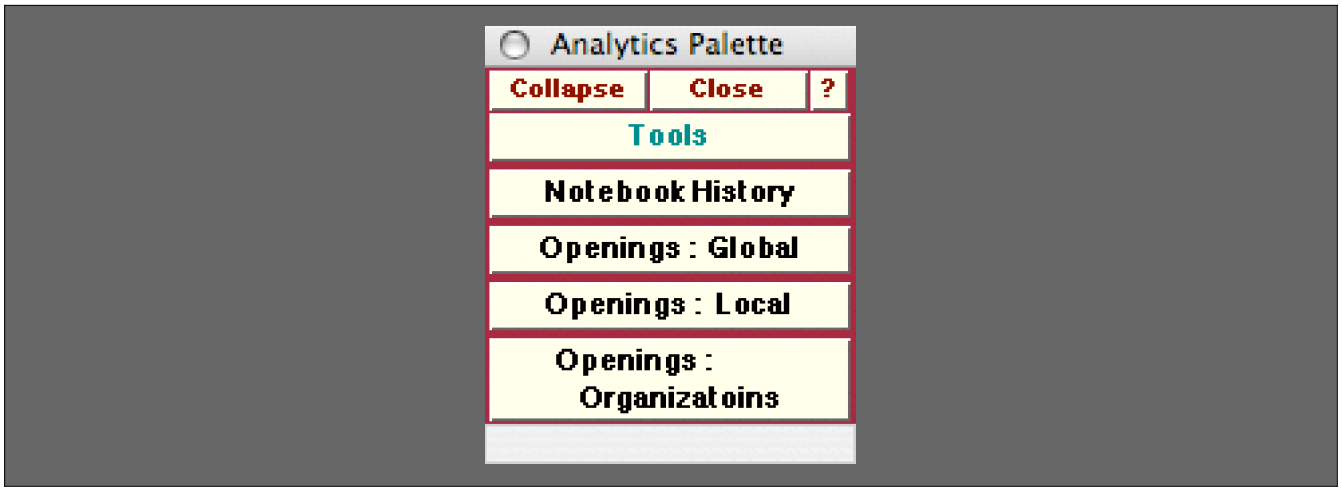
Analytics Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Analytics** button on the All Palettes Palette, you will open the All Notebooks Palette. You can also open the All Notebooks Palette by executing:

```
AnalyticsPalette[];
```



The Analytics Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Notebook History

The **Notebook History** button opens up a notebook that shows graphical and numerical information on the history of the

current InputNotebook.

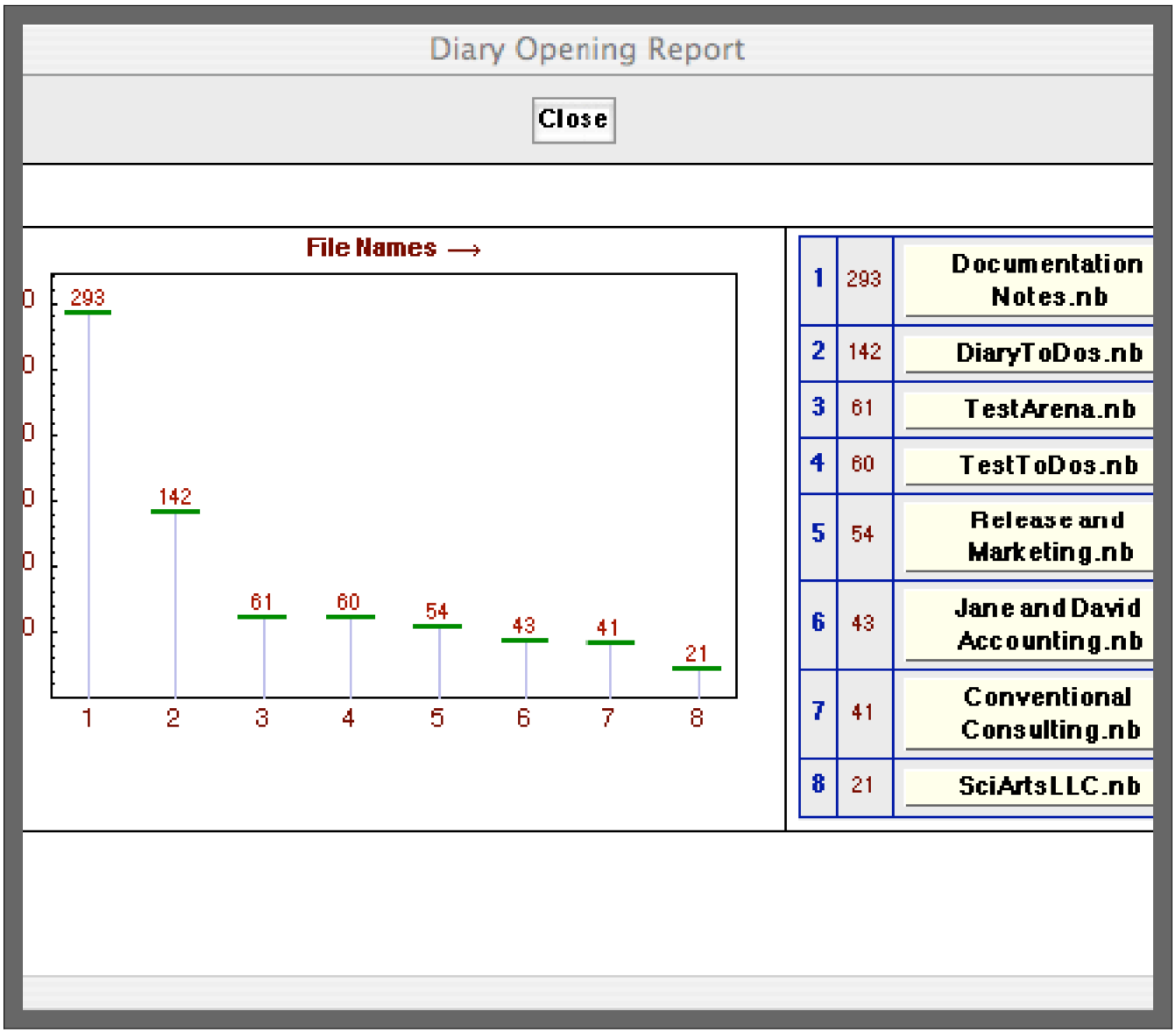
Openings : Global

The **Openings: Global** button opens a dialog that allows you to see graphical and numerical information on the statistics of the opening of all Diaries and of Notebooks.

The dialog looks like:

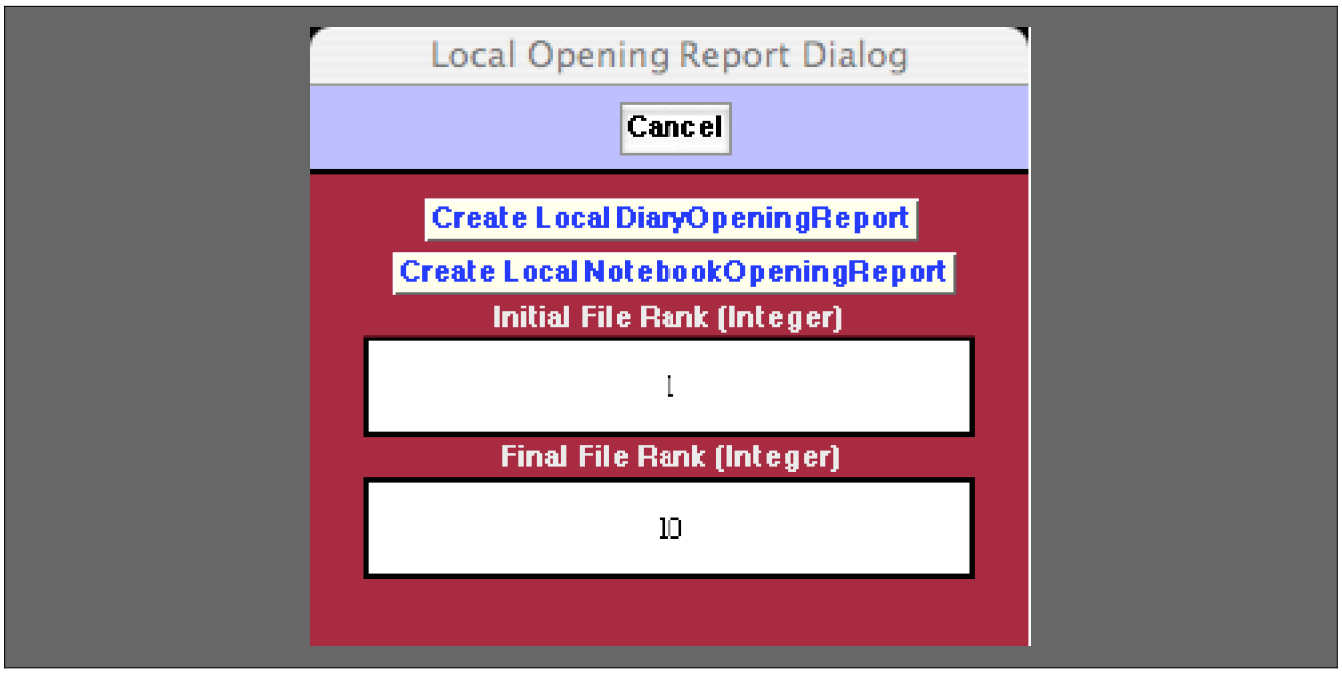
The Global Opening Report Dialog

In this dialog you specify the initial and final file ranks and then choose whether you want to view the opening report for Diaries or Notebooks. The file ranks are simply the rank of the file in a list of the files ordered by how many times they were opened. The first one is the one that has been opened the most, the second one opened the second largest number of times, and so on.



The **Openings: Local** button opens a dialog that allows you to see graphical and numerical information on the statistics of the opening of all Diaries in the current Diary directory and of Notebooks in the current Diary's Notebooks subdirectory.

The dialog looks like:



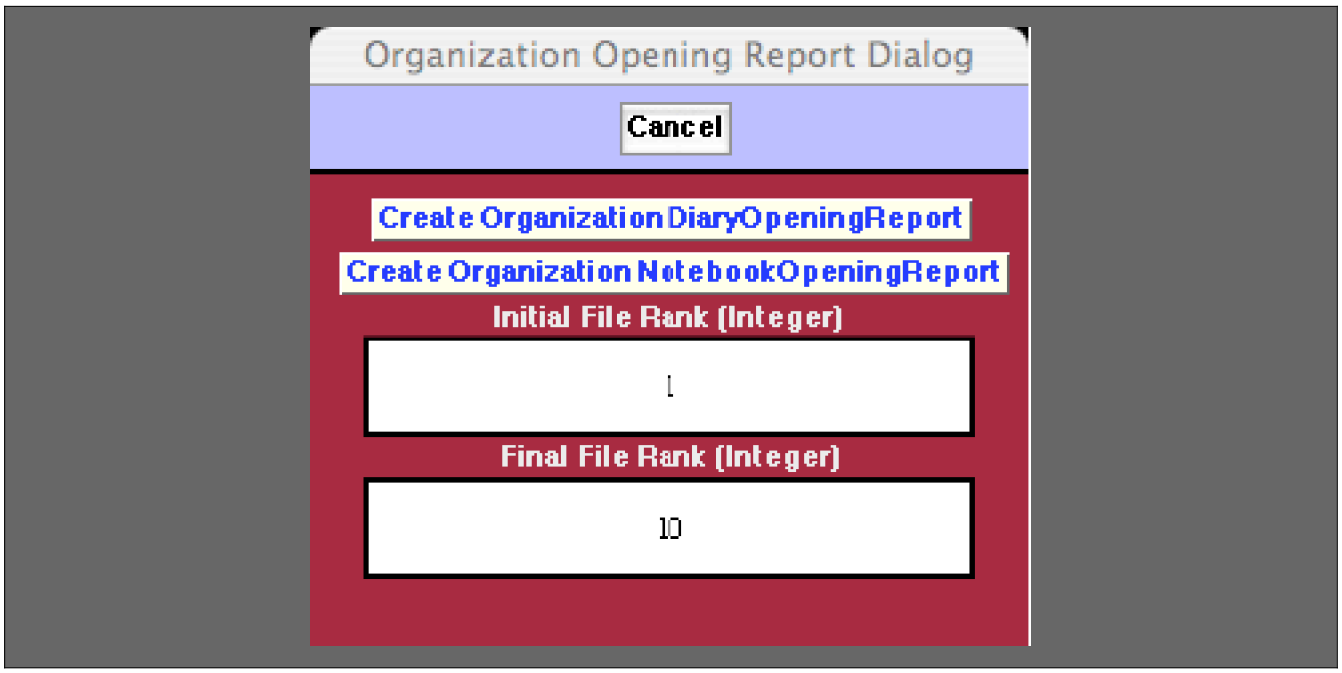
The Local Opening Report Dialog

In this dialog the meanings of the fields are similar to those in the Global Opening Report Dialog. The only difference is that the buttons generate a report for just the Diaries that are in the current Diary's directory and the Notebooks in the current Diary's Notebooks subdirectory. Thus it is a localized view.

- **Openings : Organizations**

The **Openings: Organizations** button opens a dialog that allows you to see graphical and numerical information on the statistics of the opening of all Diaries that are members of a particular Organization and of Notebooks that are members of a particular Organization.

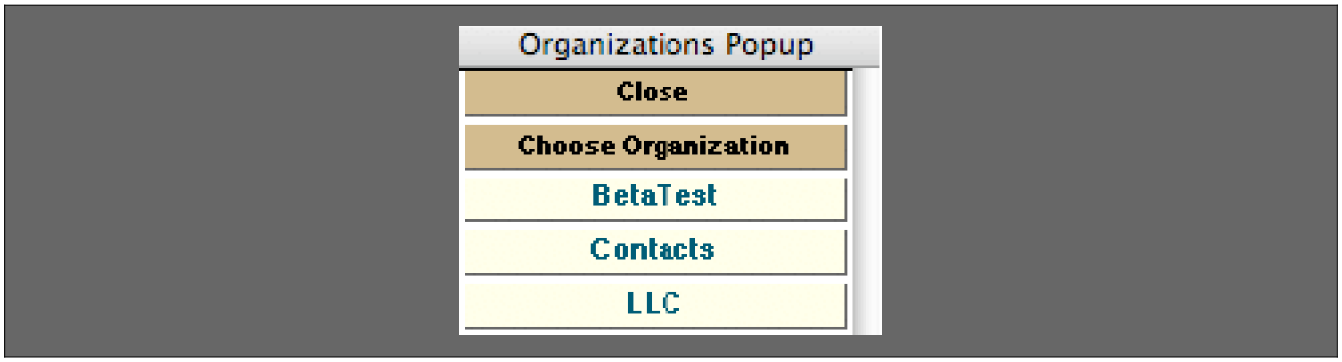
The dialog looks like:



The Organization Opening Report Dialog

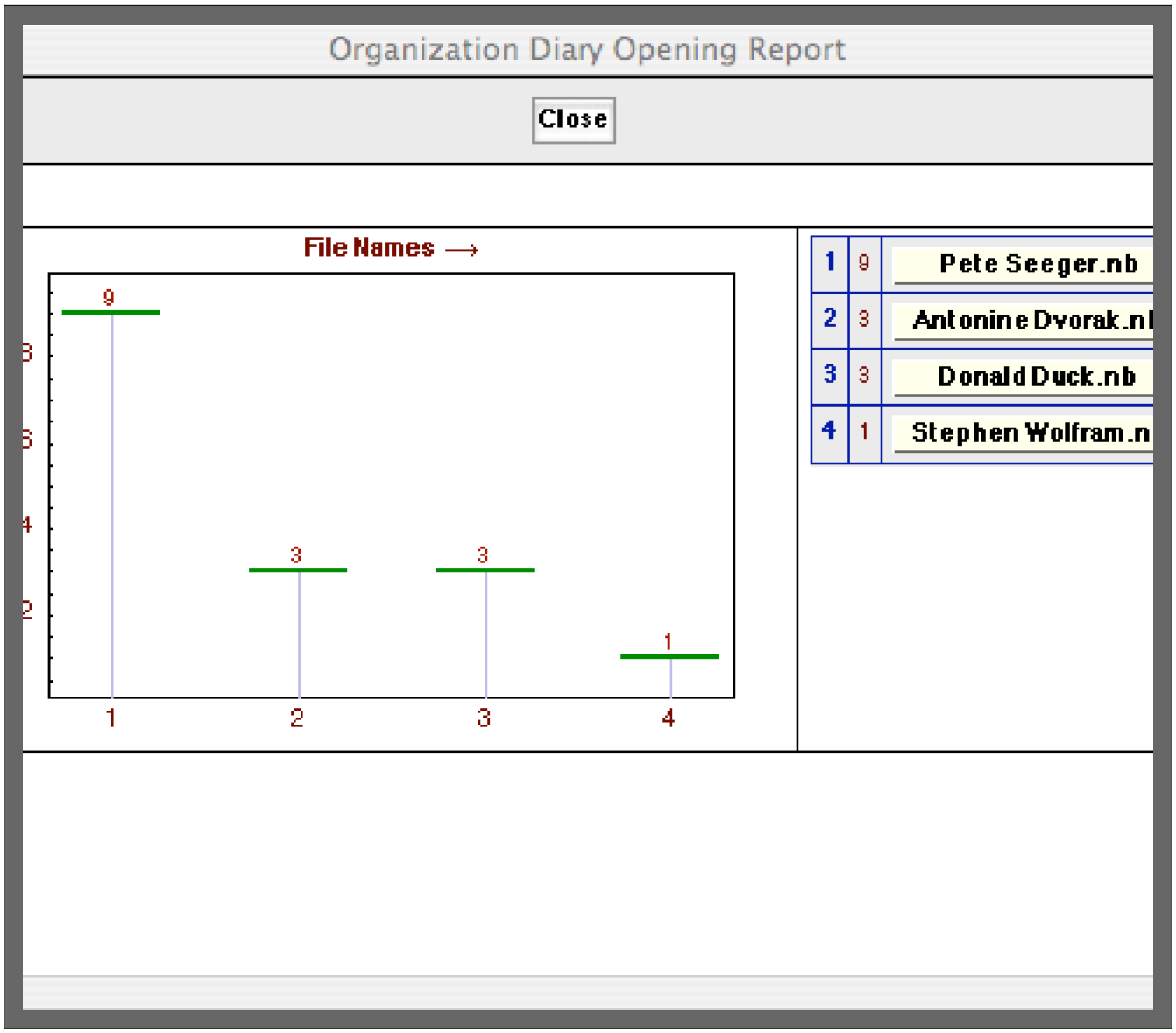
In this dialog the meanings of the fields are similar to those in the Global Opening Report Dialog. The only difference is that the buttons generate a report for just the Diaries that are in an Organization (to be chosen in the next step) or just the Notebooks that are in an Organization. Thus it is a localized view of a different sort than provided by the **Openings: Local** button .

When one of the buttons **Create OrganizationDiaryOpeningReport** or **Create OrganizationNotebookOpeningReport** are clicked, a popup menu appears that allows you to select the Organization that you want the report on. This popup menu looks like the following, although the choices will be different depending on what Organizations you have created:



The Organizations Popup

When, for example, the **Contacts** button is clicked on, a report is generated that might look like the following:



In this case there were only four Diaries within the Organization "Contacts."

Blog Tools Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Blog Tools** button on the All Palettes Palette, you will open the Blog Tools Palette. You can also open the Blog Tools Palette by executing:

```
BlogPalette[];
```

This Palette provides tools for the creation, entry into, and manipulation of Blogs created from within *Mathematica*.

Other essential blogging functionality is provided by means of the buttons in the Blog toolbar which appears at the top of each Blog entry.

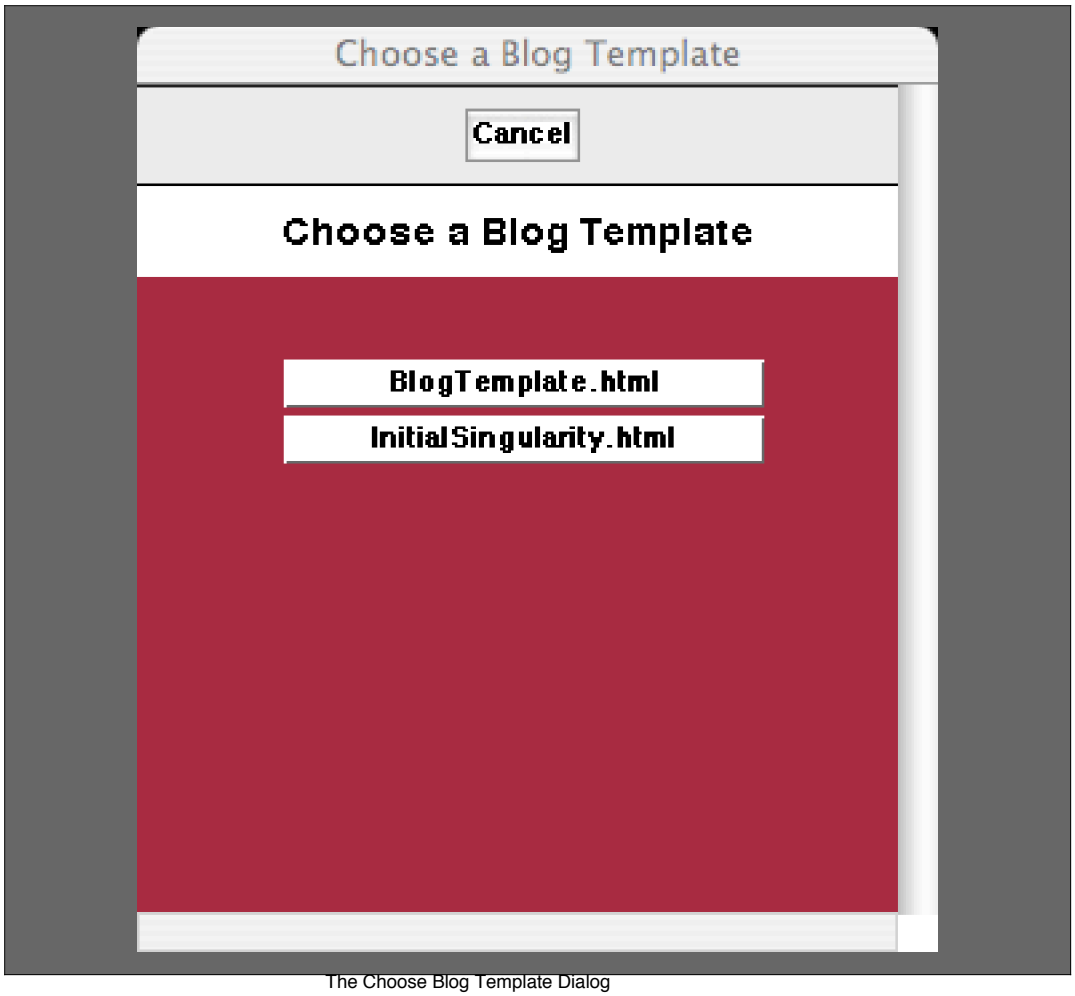
The Blog Tools Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Select Template

The **Select Template** button opens a dialog which allows you to select an HTML template to use in a particular Blog. This dialog looks like:



The Choose Blog Template Dialog

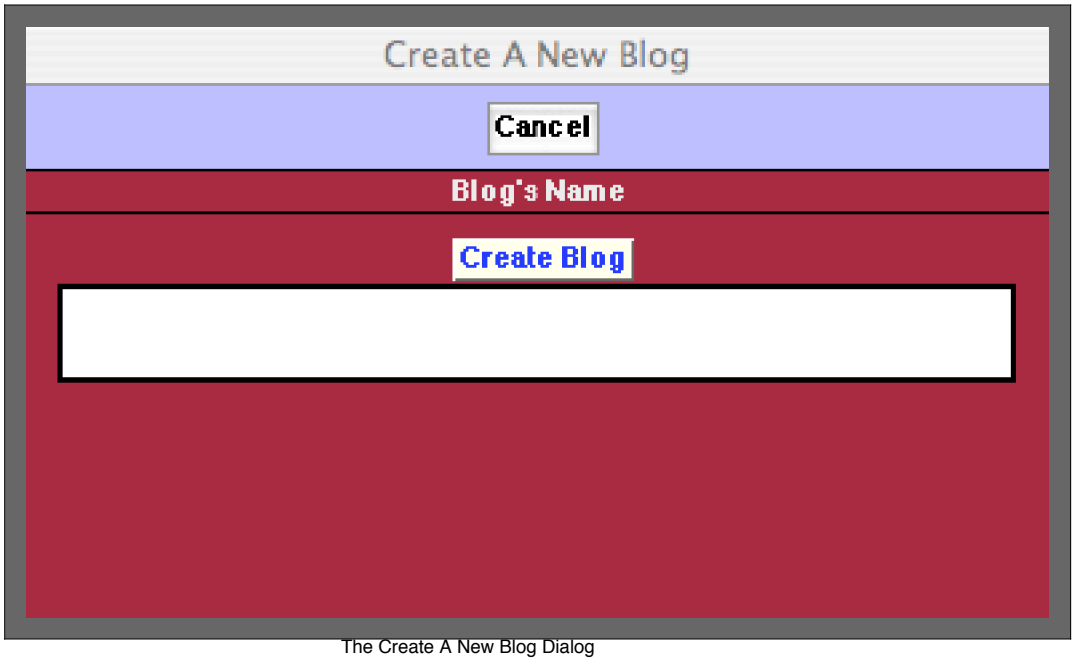
In this example there are two templates to choose from, the default BlogTemplate.html and a user-supplied Initial-Singularity.html. To apply a template to a given Blog, the insertion point must me within a Blog entry that resides in a

Diary.

> *Blog entries exclusively are placed in Diaries. They are not placed in other kinds of notebooks.*

- **New Blog**

This button opens up a dialog for creating a **New Blog**. This dialog looks like:

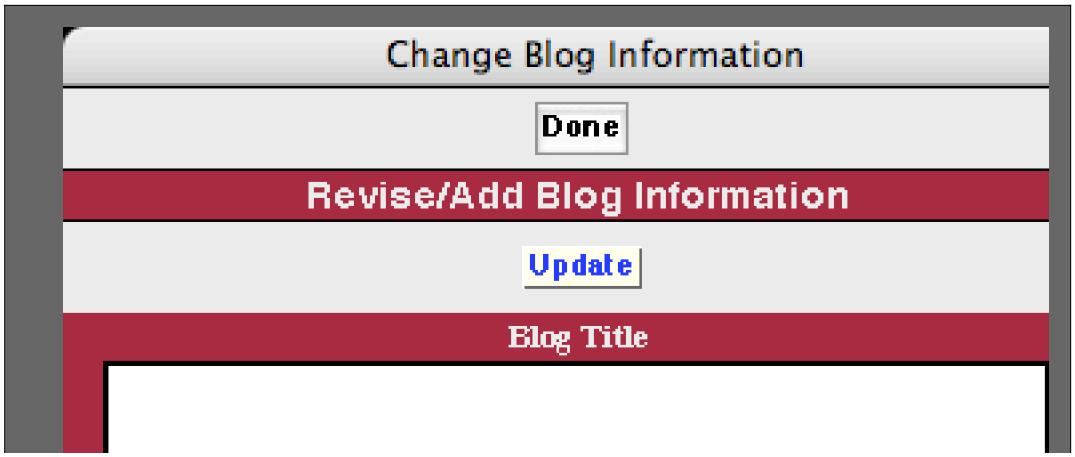


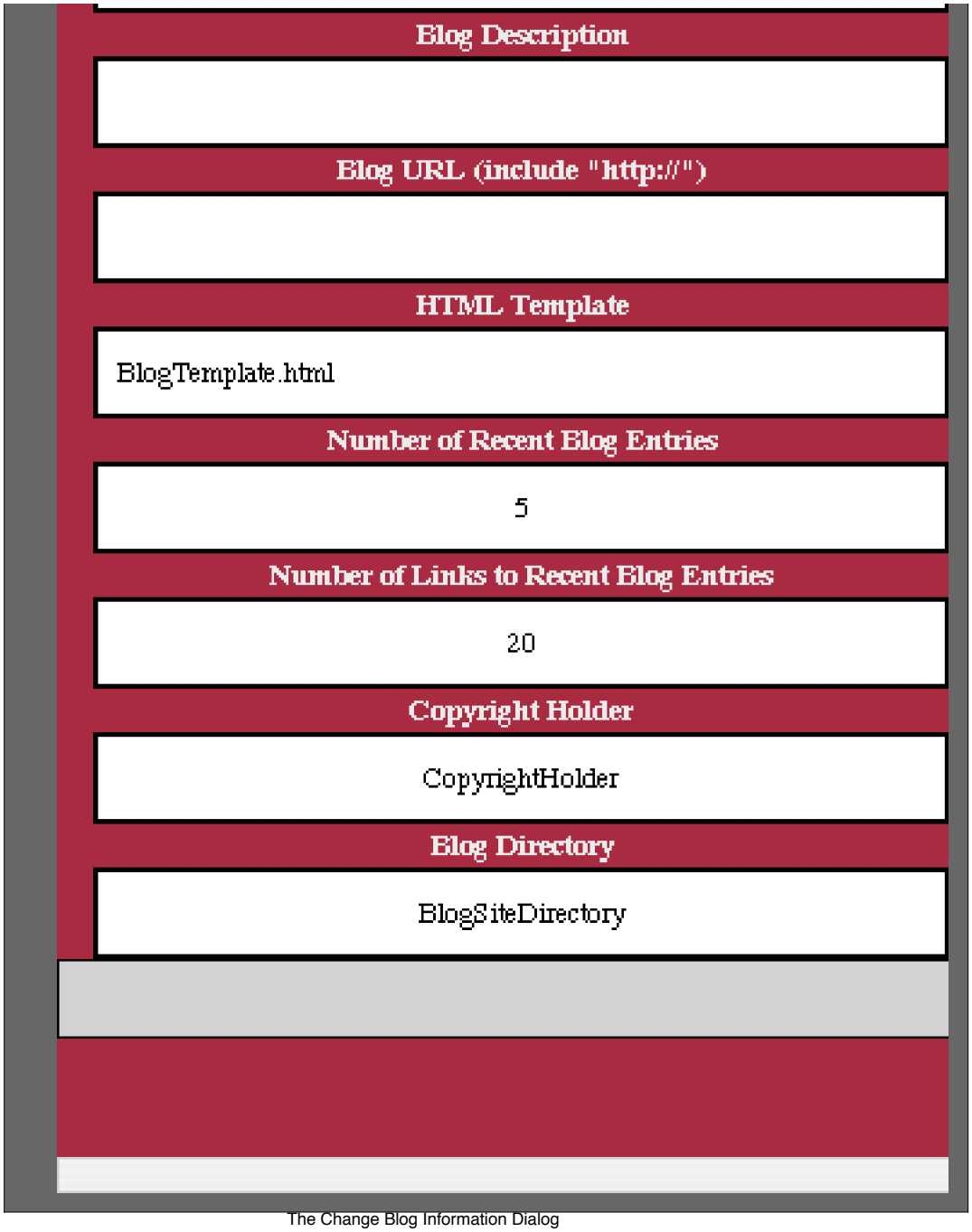
The Create A New Blog Dialog

In this dialog you place the name of the Blog.

> *The name of the Blog is different from the Blog's title. The Blog's name is used to refer to the blog within this package. The Blog's title is what will be used on the Blogs' web page. The Blog's title will be entered in the next step.*

When the **Create Blog** button is clicked the blog is created and a second dialog is opened to add information specific to the newly created blog. This dialog looks like:





The Change Blog Information Dialog

This dialog has eight input fields:

1. **BlogTitle**: The title of the blog. This is what will be used on the blogs web page.
2. **Blog Description**: A short description of the blog for use in RSS Feeds and other places.
3. **Blog URL**: The home URL of the blog for the creation of hyperlinks back to the front page of the blog.
4. **HTML Template**: The HTML Template that will be used for the blog and its entries. This must be one of the templates from the list given from executing:

```
BlogTemplates[]
```

Number of Recent Blog Entries: This is the number of links to recent blog entries that will be displayed on the Blog's front page. It must be an integer.

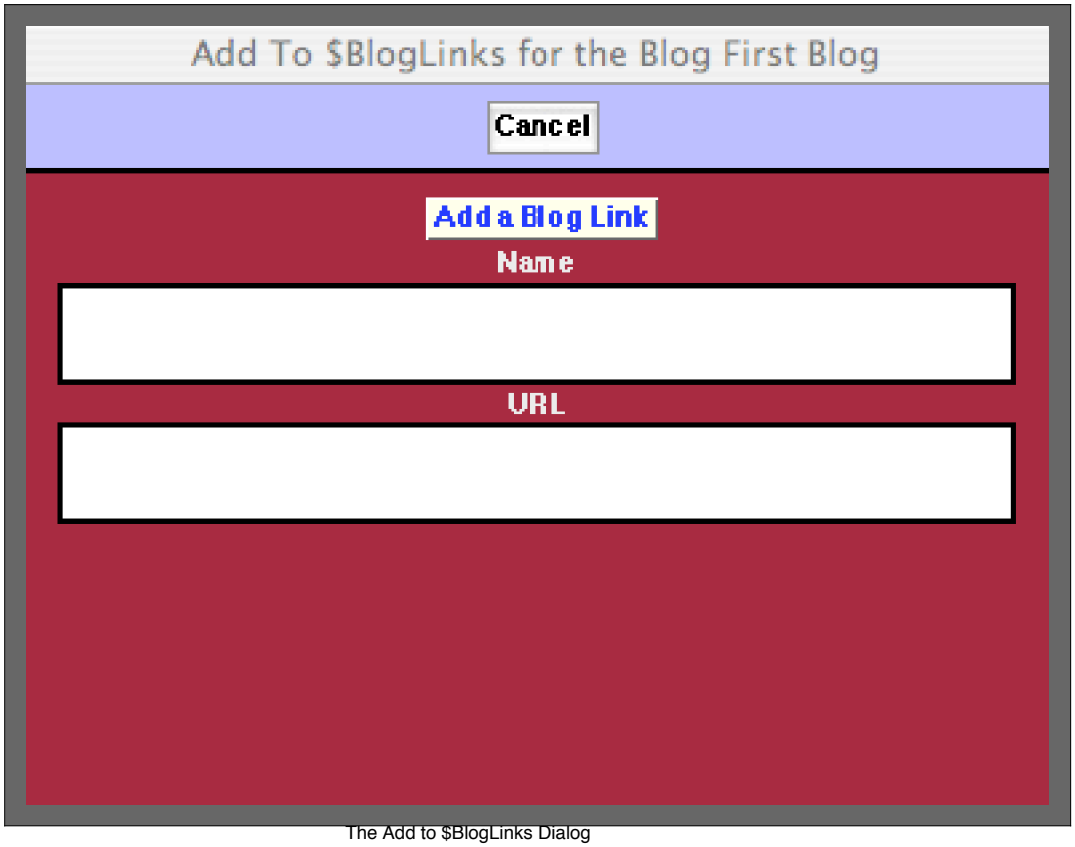
6. **Number of Links to Recent Blog Entries**: This is the number of recent blog entries that will be displayed on the Blog's front page. It must be an integer. All earlier links will appear on the Blog's Archive page.
7. **Copyright Holder**: This is the name of the entity that holds the copyright to the blog. It may be left blank.
8. **Blog Directory**: This is the full path to the local directory on your computer where the html source for your blog resides. When a blog entry is "published" it is then automatically copied to this directory. When this happens the original directory is backed up first. This directory must already exist. The published version of the Blog also exists in the Blog's directory.

Opens a dialog to create a **New Blog Entry** in the current Diary notebook. The dialog allows you to choose which Blog the new entry will be assigned to and to give the entry a title. The new blog entry will be placed at the end of the current Diary notebook.

The button **Update Blog Info** first opens a dialog allowing you to choose a Blog. When you click on the Blog's button in this dialog a second dialog opens allowing you to change the Blog's information as in the Change Blog Information dialog for the **New Blog** button above.

Each blog has a list of links that it contains and displays on a side bar. To add a link to a blogs list of links the **Add to Links** button opens a dialog that allows you to do this. The first dialog that opens up allows you to choose a Blog. Then, when the Blog is chosen, a second dialog opens that permits you to add a link and to name it.

This second dialog looks like:



The Add to $BlogLinks Dialog

Create Hyperlink

Opens the *Mathematica* dialog to create a hyperlink for the selected text in the Blog entry. A hyperlink to a web site or resource can be used or a link to a tagged item in the current Blog entry.

> *If you link to a tag elsewhere in the Diary in which the current Blog's entry exists that link will not function in the web html version of the Blog.*

Add Blog Cell

Opens and closes the **Add Blog Cell** sub-palette which includes the six buttons that follow.

Text

Adds a new **Text** cell below the current insertion point in the current Blog entry. If the insertion point is not within a Blog entry, then an error message is generated.

Adds a new **Input** cell below the current insertion point in the current Blog entry. If the insertion point is not within a Blog entry, then an error message is generated.

Adds a new **Subsection** cell below the current insertion point in the current Blog entry. If the insertion point is not within a Blog entry, then an error message is generated.

Adds a new **Subsubsection** cell below the current insertion point in the current Blog entry. If the insertion point is not within a Blog entry, then an error message is generated.

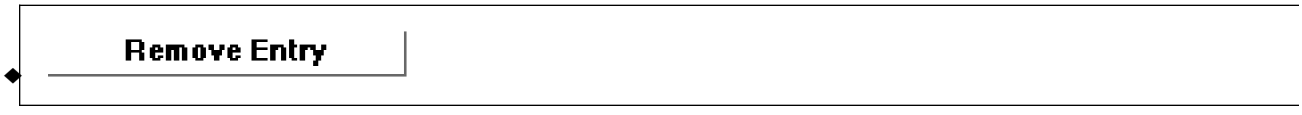
Adds a new **Notes** cell below the current insertion point in the current Blog entry. If the insertion point is not within a Blog entry, then an error message is generated. **Notes** cells are *not* included in the HTML version of the Blog entry.

Additional CellStyles can be added to the Blogging Palette at this location by using the function `AddCellStylesToBlogPalette`.

The **Place Image** button opens up a dialog that allows you to choose an image file on file system. This image file is read and its data is converted to a bitmap that is placed in the Blog entry.

Opens and closes the **More Log Functions** sub-palette which includes the two buttons that follow.

The **Create Notebook** button opens up a new notebook containing the contents of the Blog entry. This excludes the toolbar cells and any notes cells.



Removes the Blog entry. The insertion point must be within the Blog entry. Removed Blog entries are copied to the clipboard. The clipboard will be overwritten whenever something else is copied to it.

> *Note if you accidentally remove a Blog entry and wish to recover it you should* ***immediately*** *paste it back into the Diary. A number of functions in this package make use of the clipboard and may overwrite the temporarily copied Blog entry. Also the Mathematica* Edit ▹ Undo *Menu command (or its keyboard equivalent) will remove the Blog entry from the clipboard.*

Opens and closes the **Directories** sub-palette which includes the three buttons that follow.

Opens up the **Blogs Directory** which is the subdirectory of the current Diary's directory that contains Blogs that are associated with it.

Opens a dialog to choose a Blog from a list of Blogs associated with the current Diary. When one of these buttons is clicked the associated **Blog's Directory** is opened.

Opens a dialog to choose a Blog from a list of Blogs associated with the current Diary. When one of these buttons is clicked the **HTML Directory** associated with this Blog is opened.

Opens the **Formatting** Palette. This can also be opened programmatically by executing:

```
FormattingPalette[];
```

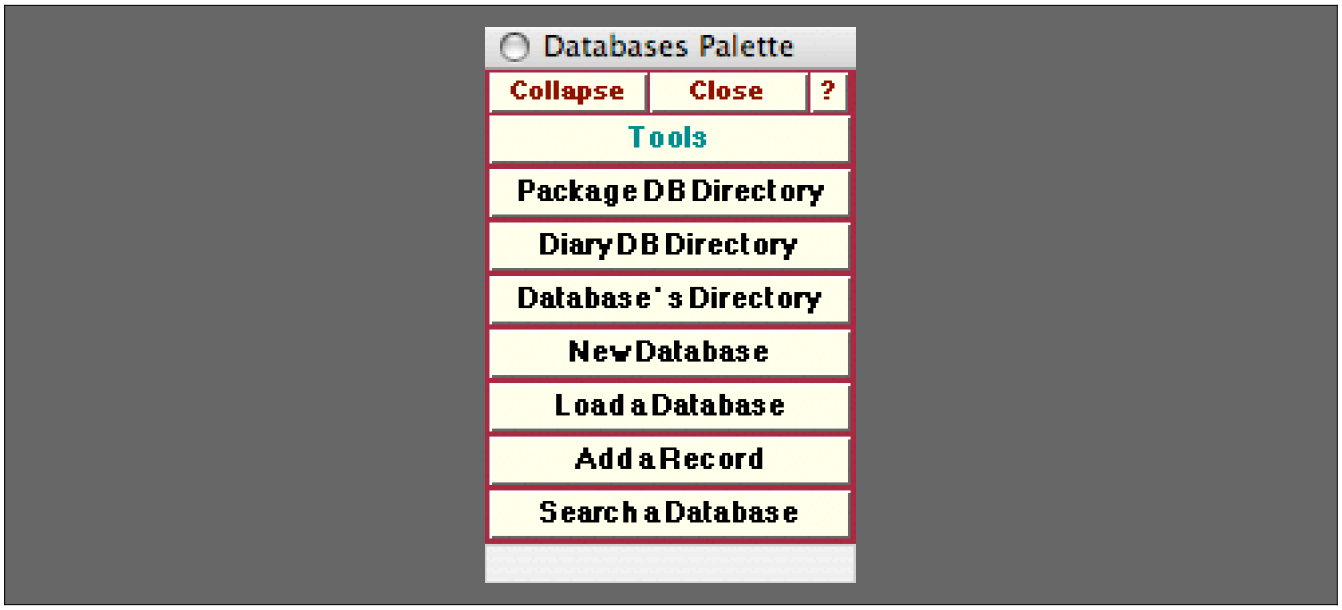
Databases Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Databases** button on the All Palettes Palette, you will open the Databases Palette. You can also open the Databases Palette by executing::

```
DatabasesPalette[];
```



The Databases Palette

The **Databases** Palette provides administrative buttons for managing Databases in the WorkLife FrameWork™ Package.

In this Palette each of the buttons below the **Databases** button, when clicked on, loads the indicated Database.

Although each Databases on this Palette is associated with a specific Diary (or group of Diaries that live in the same directory) the Databases listed in the **Databases** *Palette are ones that are known to the* WorkLife FrameWork™ *Package. These are contained in the list* `$Databases`. *If a Database is not listed on this Palette it can still be loaded using the function* `LoadDatabase`.

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Package DB Directory

The WorkLife FrameWork™ Package has a global directory where Databases useful for the function of the package are located—the **Package DB Directory** button opens up this directory.

Diary DB Directory

The **Diary DB Directory** button opens the `Databases` subdirectory of the current Diary's directory that contains the directories of individual Diaries.

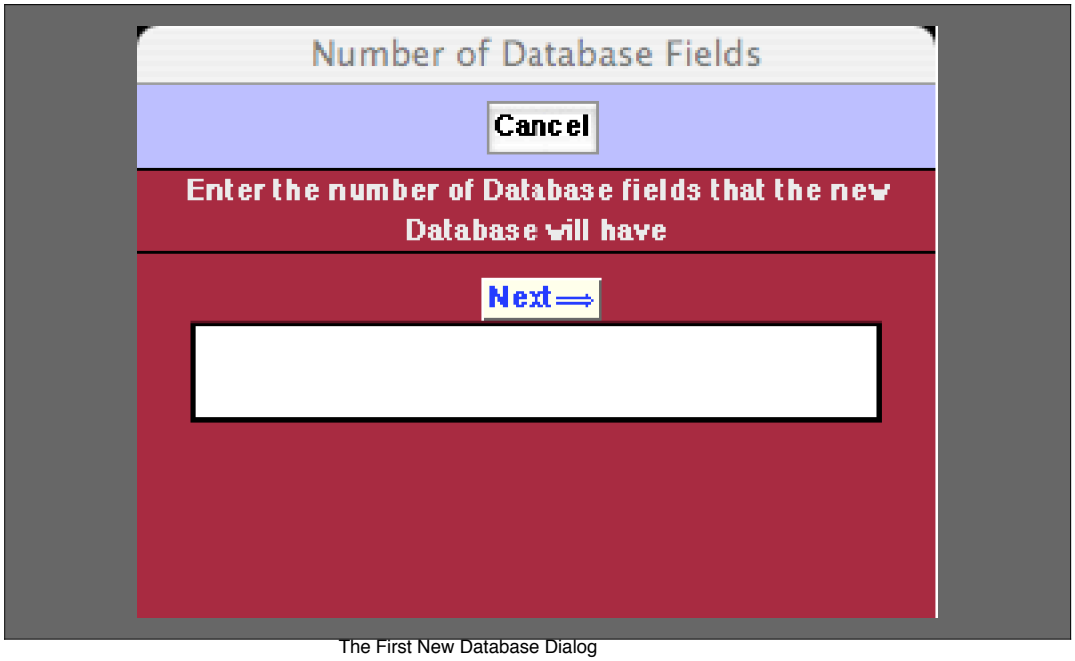
Database's Directory

This button opens a popup menu containing buttons listing the known databases that have been created. Clicking on one of these buttons will open the directory that that Database resides in.

New Database

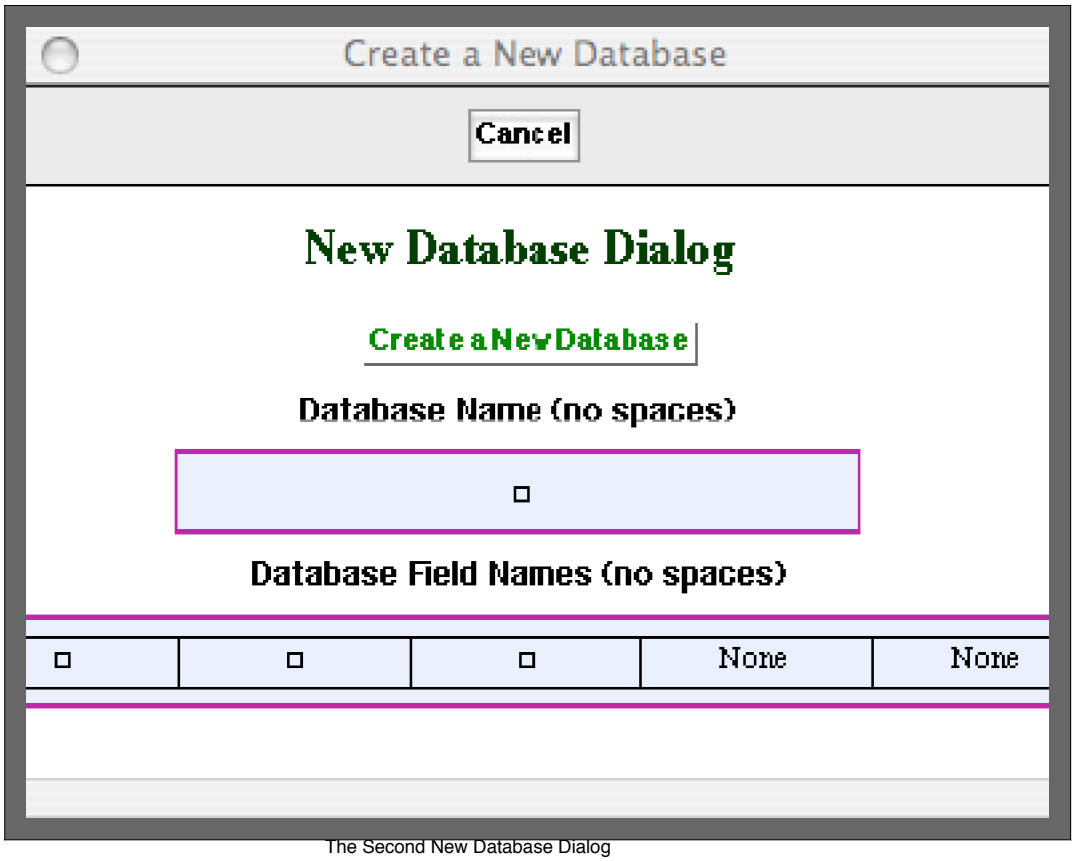
This button opens a dialog for creating a **New Database**.

This dialog looks like:



The First New Database Dialog

After the number of Database fields is entered into this dialog and the Next⟹ button is pressed a second dialog allow you to name the Database and enter the field names. In the following illustration of this second dialog the number of Database fields has been chosen to be 3.



The Second New Database Dialog

In this dialog you supply a name for the database and Field Names for the indicated fields.

> *The boxes with* "None" *are just placeholders and generally should not be modified, though doing so will have no effect on the creation of the Database.*

When the **Create a New Database** button is clicked the new database will be created in the Databases subdirectory of the current Diary's directory and the **Databases** Palette will be updated to list the new database.

This button opens a popup menu containing buttons listing the known databases that have been created. Clicking on one of these buttons will load that database.

This button opens a popup menu containing buttons listing the known databases that have been created. Clicking on one of these buttons will open a dialog Notebook that allows you enter a new record to add to the chosen database. When a button in the popup is clicked on the database will first be loaded if it has not been loaded yet.

This button opens a popup menu containing buttons listing the known databases that have been created. Clicking on one of these buttons will open a dialog Notebook that allows you enter information to base a search on for the chosen database. When a button in the popup is clicked on the database will first be loaded if it has not been loaded yet.

Diary Entries Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Diary Entries** button on the All Palettes Palette, you will open the Diary Entries Palette. You can also open the Diary Entries Palette by executing

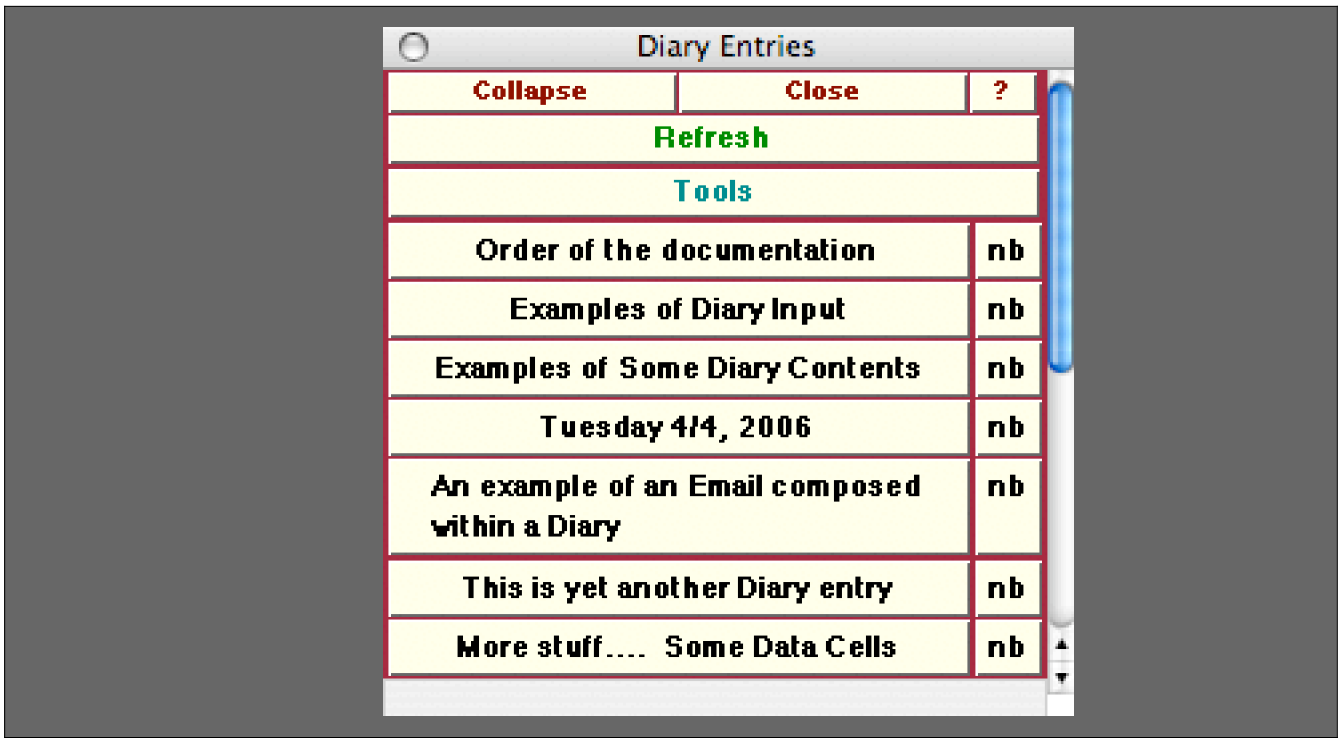
```
DiaryEntriesPalette[];
```

The **Diary Entries** Palette provides sets of buttons that give simple navigational access to individual entries in the current diary. An individual entry in a diary is the sets of cells that are underneath a `"Section"` (or Heading) cell.

In its standard version the **Diary Entries** Palette gives a pair of buttons for each entry in the diary, sorted chronologically

according to the dates of the entries.

An example of this Palette looks like:



An Example of the Diary Entries Palette

In the example of this picture, the Diary has four entries corresponding to the dates shown. The dates were the text in the section headings of the entries. If any of the section headings had material other than a date, then that would be shown in the respective button.

The Palette Buttons

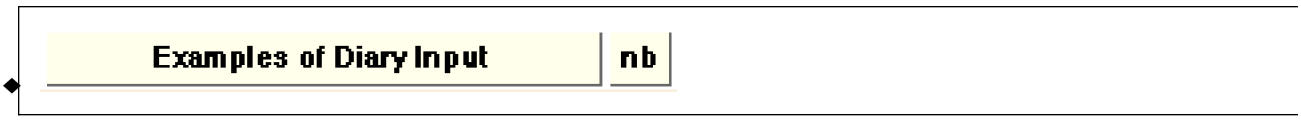
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Order of the documentation **nb**

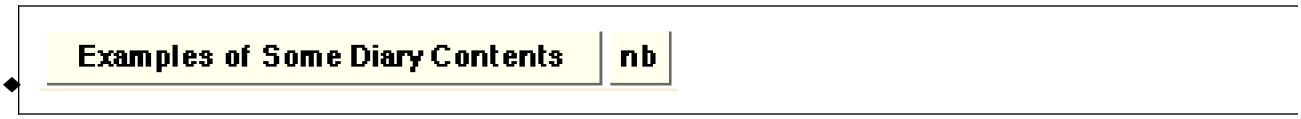
In this example, when the button with the date (**Order of the doucmentation**) is clicked on, the cells corresponding to that entry are all selected and any closed cell groups are opened. All other cell groups in the Diary are closed (collapsed) and the Diary window is scrolled to the top of the selected entry.

When the button **nb** is clicked on, a new notebook is opened up containing only those cells corresponding to the given

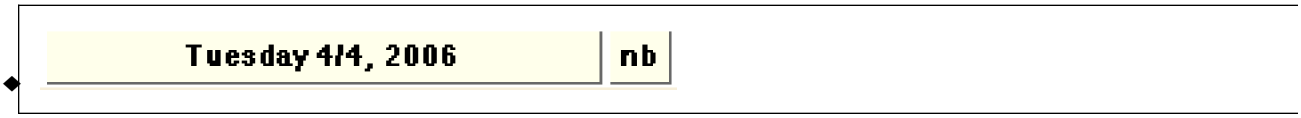
entry.



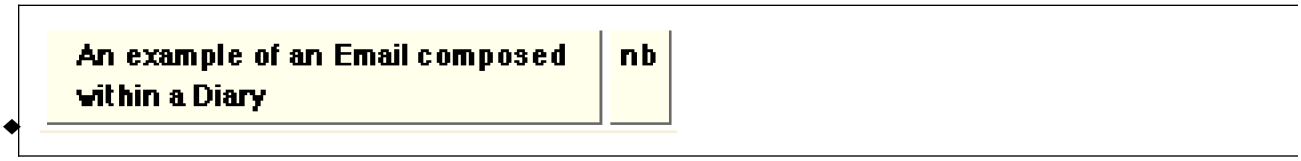
A different entry in the Diary created after than the preceding one.



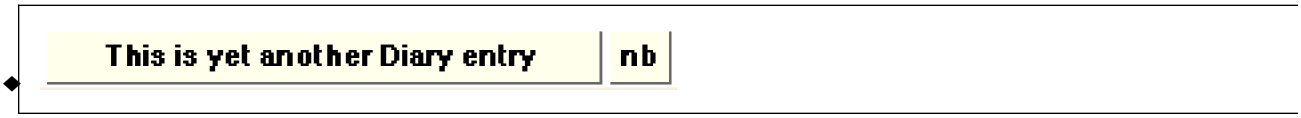
A different entry in the Diary created after than the preceding one.



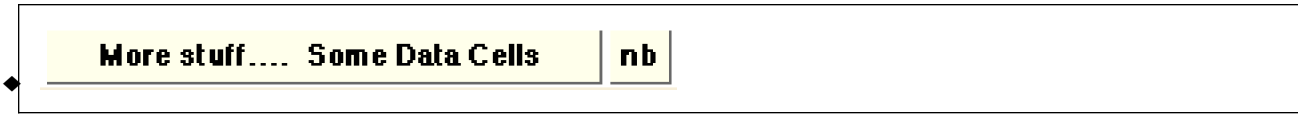
A different entry in the Diary created after than the preceding one with the default heading that consitutes the date the entry was created..



One more entry in the Diary created after than the preceding one.



One more entry in the Diary created after than the preceding one.



The final entry in the Diary created after than the preceding one.

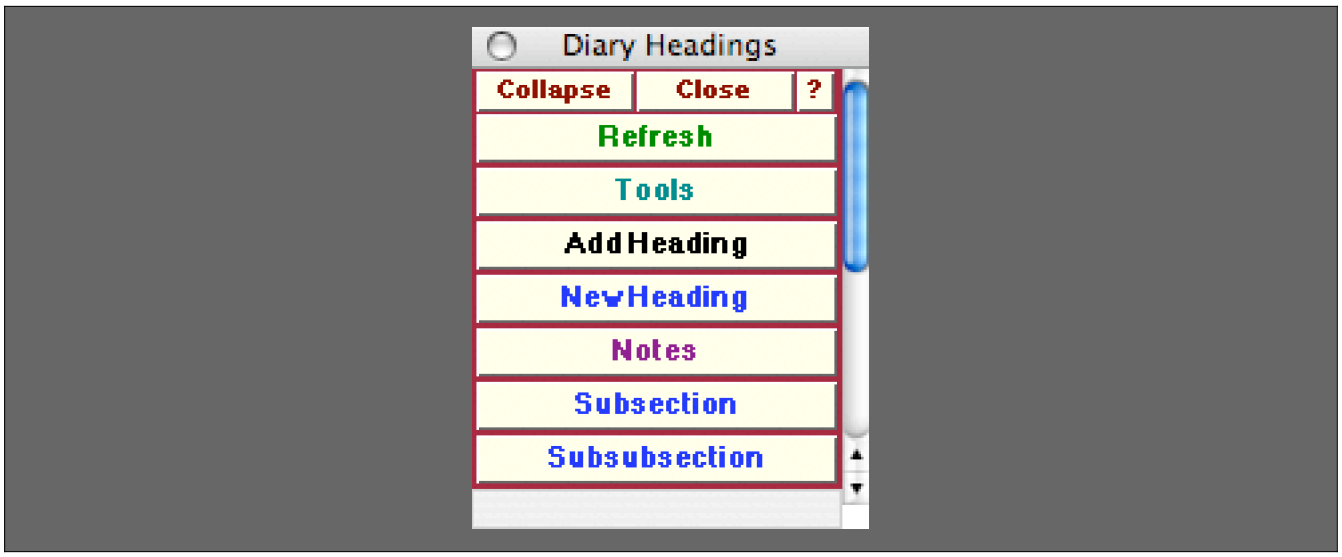
Diary Headings Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Diary Headings** button on the All Palettes Palette, you will open the Diary Headings Palette. You can also open the Diary Headings Palette by executing

```
DiaryHeadingsPalette[];
```



The Diary Headings Palette with the default headings shown as well as the user-added custom heading, "Notes."

The **Diary Headings** Palette allows you to add items to the current Diary with headings that are other than the default ones. Generally, when for example the **New Heading** button is clicked—either in this Palette or in the **Diary Entry** Palette—a new Diary entry is created with the current date as the contents of the initial `Section` Cell of the entry. However, using the **Add Heading** button on this Palette opens a dialog that allows you to create other headings that can be used in place of this default.

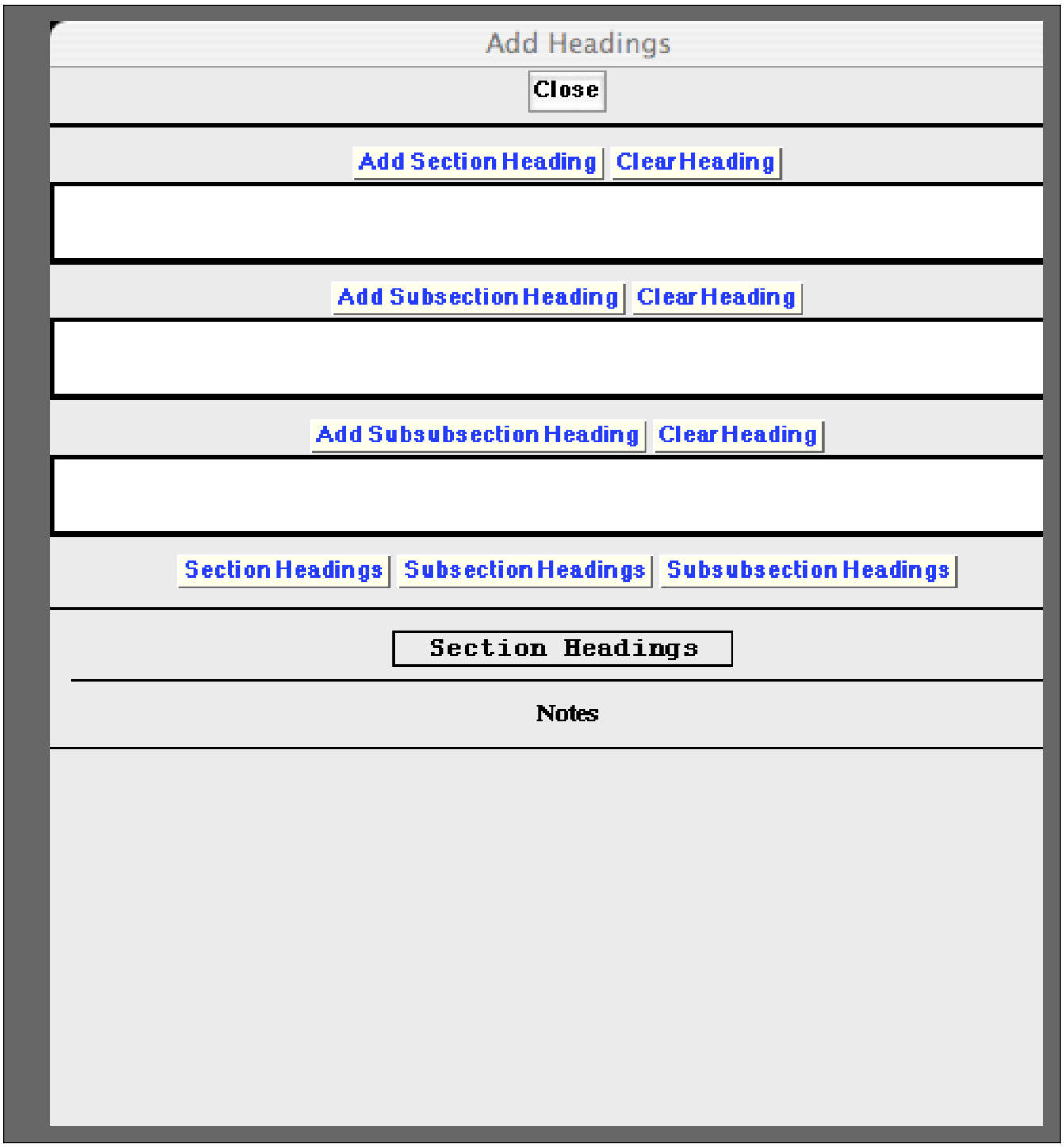
> *Of course the contents of any* **`Section`**, `Subsection`, *or* `Subsubsection` *cell can be edited by hand. This Palette give a way to create automated insertion of often-used material.*

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**

Add Heading

Opens a dialog to let the user add a custom heading to the **Diary Headings** palette. An example of this dialog is the following:



The Add Headings Dialog with the current custom Section headings shown (in this case just the custom heading "**Notes**").

Adds a **New Heading** in the current Diary Notebook at the end of the Diary.

In the example of the **Diary Headings** palette shown above there is a custom heading, **Notes**, that the user added via the **Add Heading** button and its associated dialog.

Adds a **New Subsection** in the current Diary Notebook at the end of the Diary under the current main heading. If no main heading has been created during the current calendar date then a new main heading is placed first with the default content of the current date.

Adds a **New Subsubsection** in the current Diary Notebook at the end of the Diary. If no main heading has been created during the current calendar date then a new main heading is placed first with the default content of the current date

Diary List Palette

The Palette

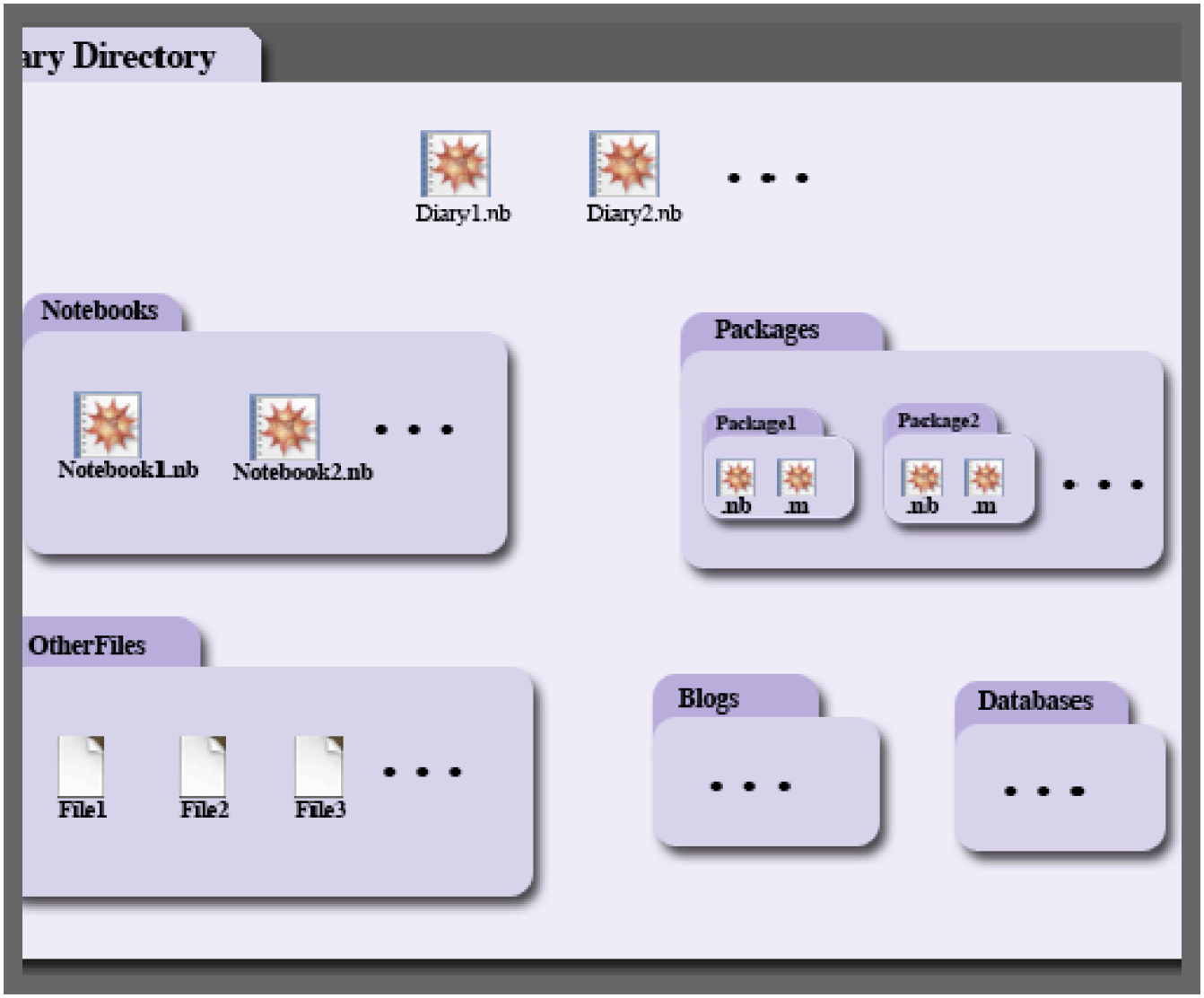
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Diary List** button on the All Palettes Palette, you will open the **Diary List** Palette. You can also open the Diary List Palette by executing

:

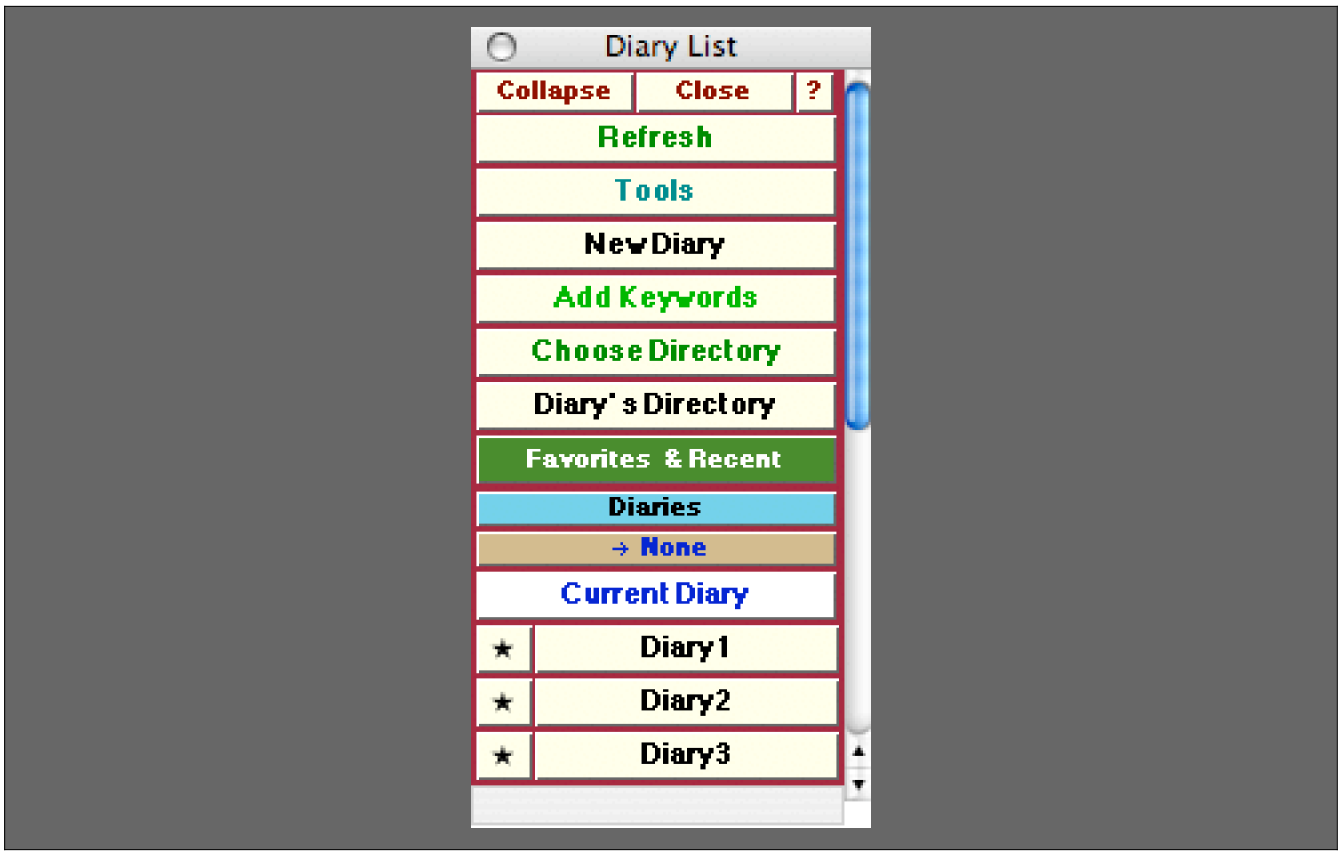
```
DiaryListPalette[];
```

The **Diary List** Palette, in addition to several administrative buttons, shows a list of buttons for opening the Diaries that are in the current Diary's directory. Recollect that a Diary's directory may have multiple Diaries as in this diagram:



The Directory Structure of a Diary or an Associated Group of Diaries

So the **Diary List** Palette will list the Current Diary and any other that coexist with it in the current Diary directory. In the example of the preceding diagram—if there are three Diaries— the **Diary List** Palette would look like:



An Example of the Diary List Palette

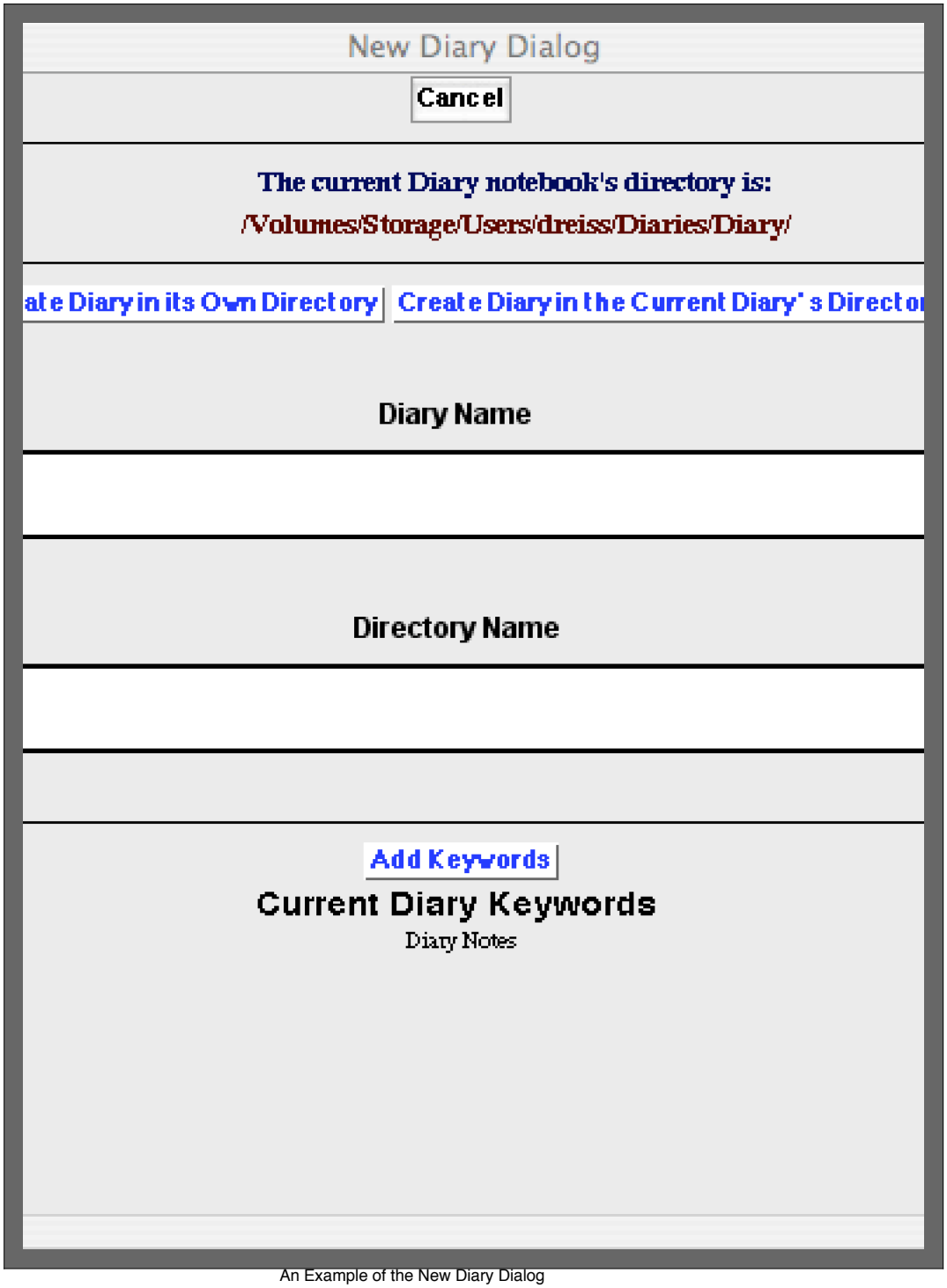
Note that the Diary List Palette generally lists ***all*** *of the Notebooks contained in the Current Diary Directory, whether or not they are, in fact, Diaries. Clicking on the button of a Notebook in the Diary List Palette that is not a Diary will elicit an error message that instructs you how to convert the given notebook into a Diary if you wish to. You can change the behavior of which Notebooks are displayed in the Diary List Palette through the use of Keywords. The functions that are relevant to this are* `AddDiaryKeywords` *and* `DeleteDiaryKeywords`. *The default behavior is to accept all Keywords through the wildcard* `"*"`.

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

New Diary

This opens up a dialog window to create a **New Diary**. The **New Diary** dialog looks like:



An Example of the New Diary Dialog

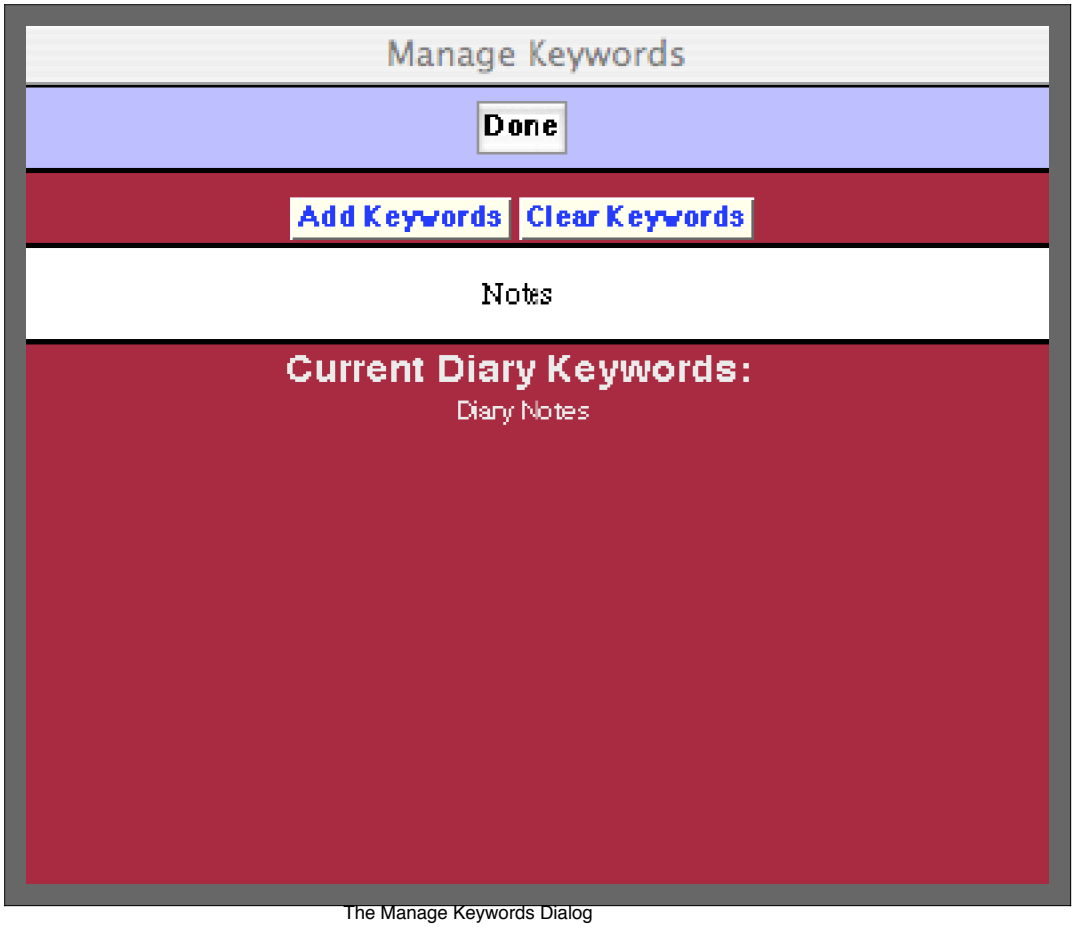
At the top of this dialog is information on the current Diary directory.

If you enter a name in the **Diary Name** field and click on **Create Diary in the Current Diary's Directory**, then a new Diary with that name will be created there and that Diary will open and become the current Diary. (In this case any entry into the **Directory Name** field will be ignored.)

If instead you enter a name in the **Diary Name** field and a directory name in the **Directory Name** field and click on **Create Diary in its own Directory**, then a new Diary with that name will be created in a subdirectory of the current Diary's directory. The new Diary will open and become the current Diary and the new directory will become the current Diary's directory. (If no name is entered in the **Directory Name** field an error message will be returned if you click on **Create Diary in its own Directory**.)

- **Add Keywords**

This opens the Manage **Keywords** dialog. This dialog looks like:



The Manage Keywords Dialog

In this example of the **Manage Keywords** dialog there are two keywords, "Diary" and "Notes," and "Notes" has just been added to the list of **Current Diary Keywords** by typing it into the text field and clicking on the **Add Keywords** button. "Notes" could be removed from the list by clicking on the **Clear Keywords** button.

> *Note that the Diaries that are listed below under the* **Diaries** *button are actually* **all** *of the Mathematica notebooks contained in the current Diary's directory that contain one of the Keywords in its name. In order to avoid having to specify many Keywords you can take one of two strategies: [1] Pick a few Keywords and always use one of them in a Diary's name or [2] allow All possible words to be Keywords and make sure that no non-Diary Mathematica notebooks are placed in a Diary directory. To allow all*

possible words to be Keywords execute `AddDiaryKeywords[All]`. *The default setting is to allow all possible words to be Keywords. In general, unless you wish, you do not need to change this behavior.*

This opens the Directory Browser dialog window. In this window you can navigate your computer's directories and choose the one that you wish to open a Diary from. The resulting directory is then the value of `$DefaultDiaryDirectory`.

This opens the current **Diary's Directory**.

Opens the **Favorites & Recent** Palette. This can also be opened programmatically by executing:

```
FavoritesAndRecentPalette[];
```

This button toggles open and closed the sub-palette containing the **Diaries** that are contained in the current diary's directory. When one of these buttons is clicked, the associated Diary is opened and specified as the current Diary.

It is generally important to open notebooks (and Diaries) with buttons from the WorkLife FrameWork™ *Package. This is because when they are opened in this way additional information is added to the notebook—and entries are made in* A WorkLife FrameWork's™ *internal databases—so that they can be tracked in a variety of ways by the* WorkLife FrameWork™ *Package.*

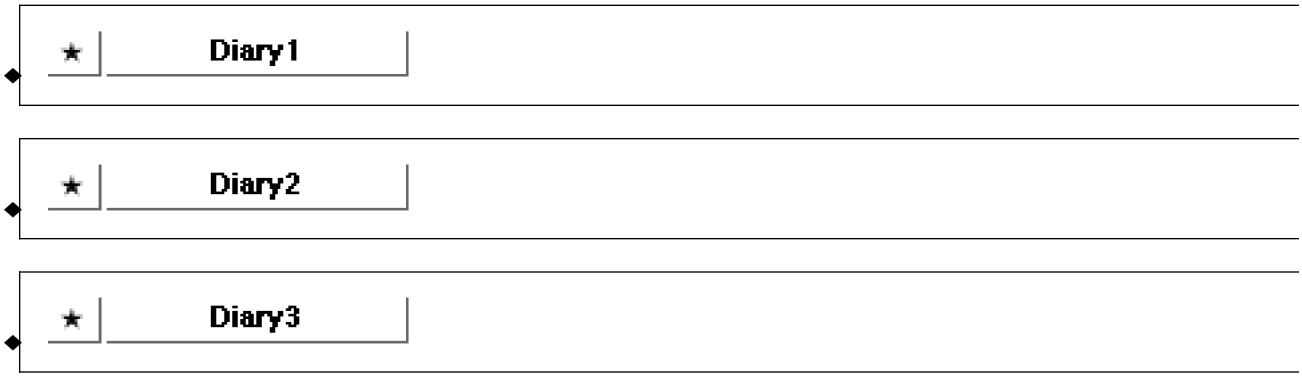
This button shows the name of the current Diary. It is refreshed whenever a Diary is made the current Diary. In this example it is the Diary named **Diary1**. If there is no current Diary then this button reads **None**. The current Diary can be opened by using the **Current Diary** button below.

This opens the **Current Diary** if it has been chosen. If not, an error message is generated.

If the current Diary is open then its window is brought to the front. If it is not opened then it is opened and any Default Code Cells are executed.

The following three sets of buttons show those Diaries that, when this document was being written, were in the current Diary's directory.

In each case the button with the Diary's name, when clicked on, opens that Diary and makes it the current Diary. The button with the ★ makes that Diary the current one but does not open it up.



Diary Templates Palette

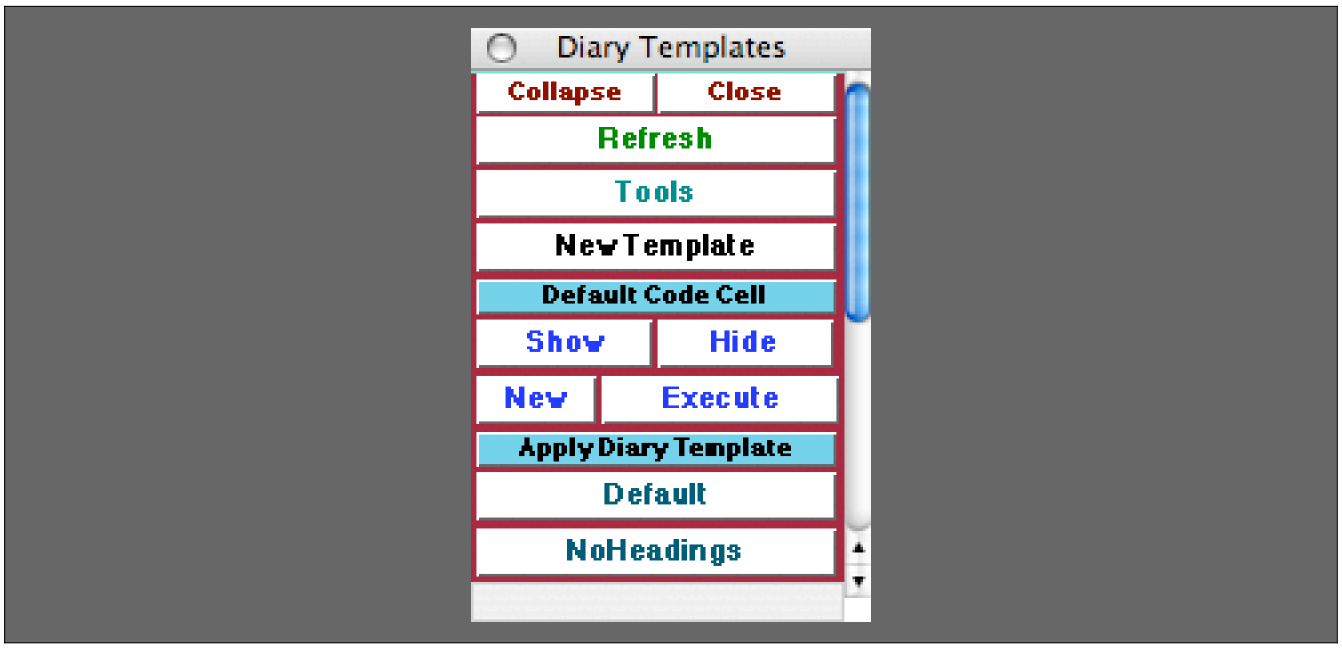
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Diary Templates** button on the All Palettes Palette, you will open the Diary Templates Palette. You can also open the Diary Templates Palette by executing:

```
DiaryTemplatesPalette[];
```

The **Diary Templates** Palette allows you to create Templates for Diaries that can be applied to a given Diary (as long as it is the current Diary). Each Diary has a set of Default Code Cells which generally are hidden. You can show these Default Code Cells for the current diary by clicking on the **Show** button, either in this Palette or in the **Diary Entry** Palette in its Defaults Code Cell. You can then place *Mathematica* code in the Default Code Cells that will be automatically executed whenever the Diary is opened (in addition to opening a diary from a button in a Palette, this also includes when the **Make Current** button is clicked on in either the Diary's toolbars or in the **Diary Access** Palette).



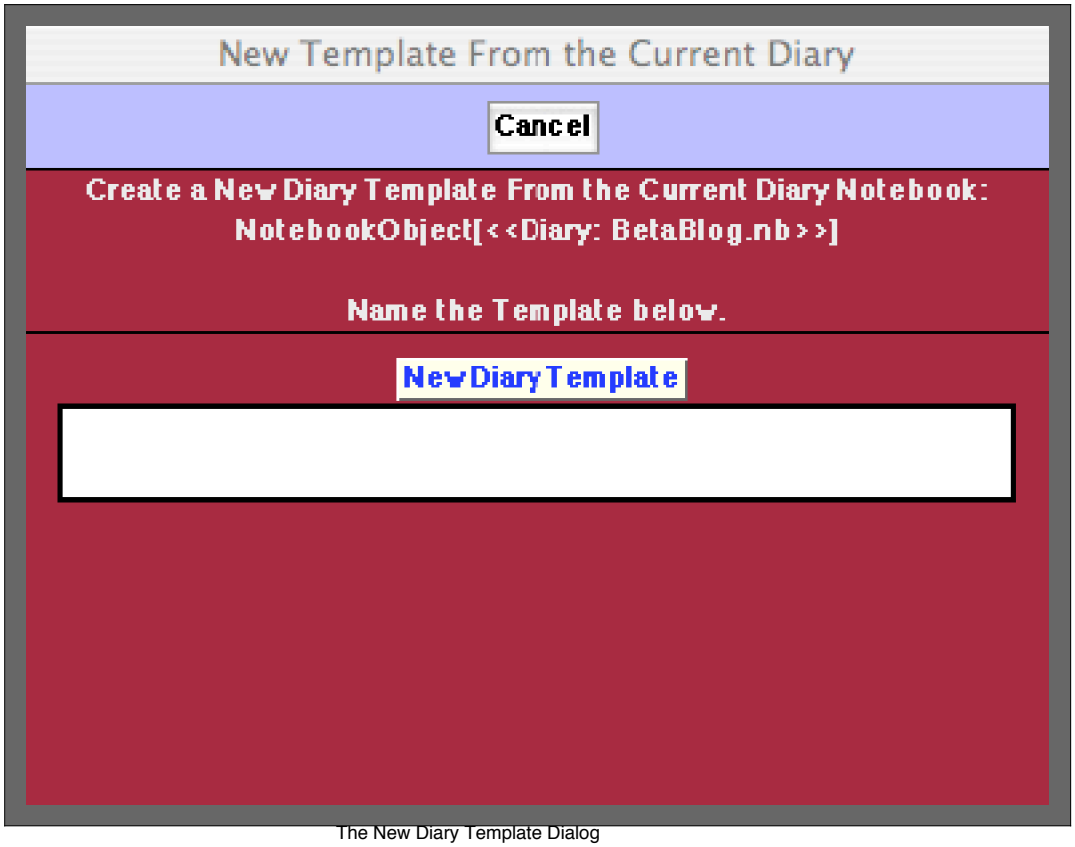
An Example of the Diary Templates Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

New Template

Opens the **New** Diary **Template** dialog.



The New Diary Template Dialog

By entering a name for the template in this dialog and clicking on the **New Diary Template** button, you will add the template to the list of template buttons below under **Apply Diary Template**.

- **Default Code Cell**

This button toggles open and closed the sub-palette containing the following two pairs of buttons concerning the default code cells of the current Diary.

- **Show** | **Hide**

The **Show** button scrolls to and opens the Default Code Cells in the current Diary notebook. The **Hide** button hides the Default Code Cells in the current Diary notebook.

- **New** | **Execute**

The **New** Button creates a new Default Code Cell in the current Diary notebook and places it below the current Default Code Cells while showing them all. The **Execute** button executes all of the Default Code Cells in the current Diary notebook.

Apply Diary Template

This button toggles open and closed the **Apply Diary Template** sub-palette containing buttons for applying saved Diary Templates to the current Diary.

Default

The **Default** button causes the Diary's Default Code Cells to revert to their default state: including no *Mathematica* code. The **Default** button always appears as the first button under **Apply Diary Template** button in this Palette. Its behavior is slightly different than the user-added template buttons that follow, in that the **Default** button removes all current Default Code Cells from the current Diary. In contrast to this, the other buttons that follow do not remove any of the Default Code Cells: they simply append the set of Default Code Cells corresponding to the template to the set of current Default Code Cells of the current Diary.

NoHeadings

This is an example of a Diary Template. In this particular case, when the NoHeadings button is clicked the following default code cell is appended to the set of current Default Code Cells of the current Diary.

↓ Place Default code in this cell ↓

```
SetAutoSectionHeading[False];
```

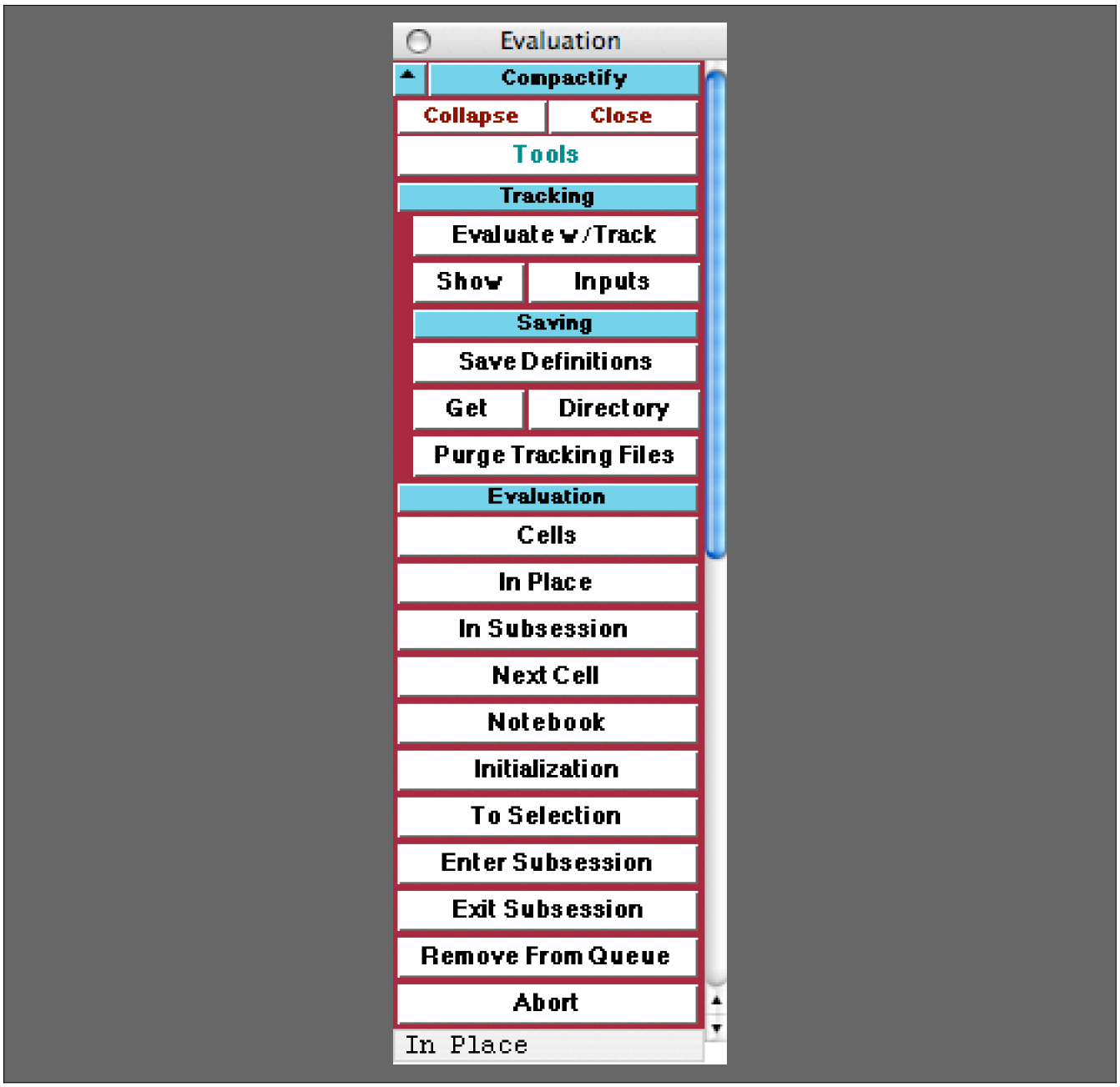
Evaluation Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: Load WorkLife FrameWork™

By clicking on the on the **Evaluation** button on the All Palettes Palette, you will open the Evaluation Palette. You can also open the Evaluation Palette by executing:

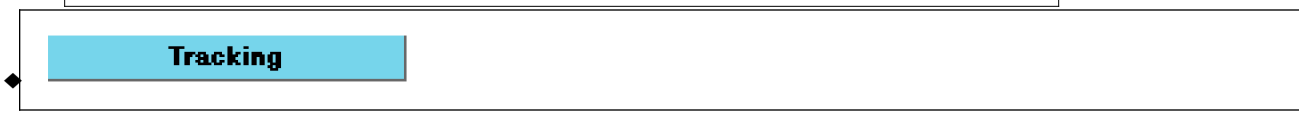
```
EvaluationPalette[];
```



The Evaluation Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**



Opens and closes the **Tracking** sub-palette which includes the two sets of buttons that follow.

The **Evaluate w/ Track** button is used to execute one or more executable cells (such as **`Input`** cells) in the current InputNotebook. The cells that are executed are appended into an Evaluation Tracking Notebook along with information on when the evaluation took place and which notebook the evaluation took place in. If a cell is executed using another evaluation approach (for example by pressing the `enter` key) then that evaluation is *not* placed in the Evaluation Tracking Notebook.

The **Show** button shows the current Evaluation Tracking Notebook. The **Inputs** button opens up a new notebook with just the inputs that have been recorded in the Evaluation Tracking Notebook.

Opens and closes the **Saving** sub-palette which includes the three sets of buttons that follow.

The **Save Definitions** button saves the current state of the `Global`` variables in the current *Mathematica* session.

Clicking the **Save Definitions** button will return an error message unless the parameter `$SaveEvaluatedToFile` has the value `True`. Its default value is `False`.

- *Note that if a variable has the Attribute* `ReadProtected` *or* `Locked`*, then its definition will not be saved.*

- *The contexts of the variables that are saved using this button are determined by the parameter* `$EvaluationTrackingContexts`*, which has the default value* `{"Global`"}`.

- *The notebooks and other files associated with Evaluation Tracking and Saving Definitions are located in the directory given by the value of the parameter* `$EvaluationTrackingDirectory`.

The **Get** button reads in the state of the `Global`` variables as of the last time the **Save Definitions** button was clicked in the current *Mathematica* session.

- *Variables that had the Attribute* `ReadProtected` *or* `Locked`*, will not be recovered from the state of Mathematica at the time that the* **Save Definitions** *button was clicked.*

The **Directory** button opens up the directory where the notebooks and other files associated with Evaluation Tracking and Saving Definitions are located.

The **Purge Tracking Files** button removes all of the notebooks and other files associated with Evaluation Tracking and Saving Definitions except for the current ones.

Opens and closes the **Evaluation** sub-palette which includes the buttons that follow.

- *None of the following buttons provide the Evaluation Tracking that was described above.*

Evaluates the currently selected **Cells** in the current InputNotebook.

Evaluates the currently selected material in the InputNotebook **In Place** and replaces the selection with the result of the evaluation.

Evaluates the currently selected **Cells** in the current InputNotebook in a subsession of the current *Mathematica* session.

Evaluates the **Next Cell** in the current InputNotebook.

Evaluates all of the cells in the current input **Notebook**.

Evaluates all of the Cells that are **Initialization** cells in the current InputNotebook.

Evaluates all of the Cells that in the current InputNotebook starting at the beginning of the notebook up until the current selection.

Causes the any current evaluations to be suspended and **Enter** a **Subsession** for evaluation.

Exits any currently active *Mathematica* **Subsession** and returns to any evaluations that were active when the subsession was entered.

Removes any cells marked for evaluation **From** the evaluation **Queue**.

Aborts the current evaluation (if possible).

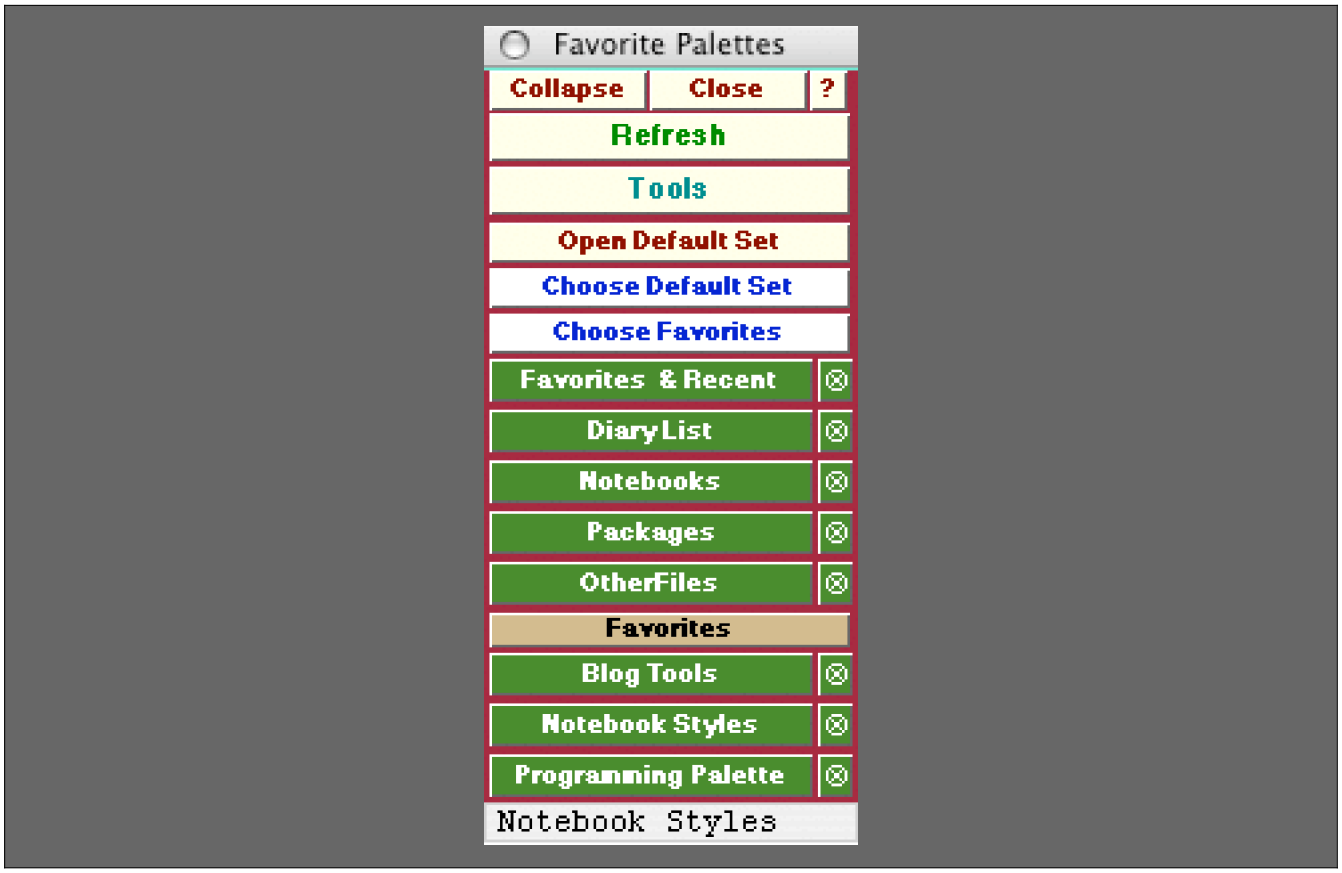
Favorites Palettes Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Favorite** Palette**s** button on the All Palettes Palette, you will open the Favorite Palettes Palette. You can also open the Favorite Palettes Palette by executing:

```
FavoritePalettesPalette[];
```



The Favorite Palettes Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

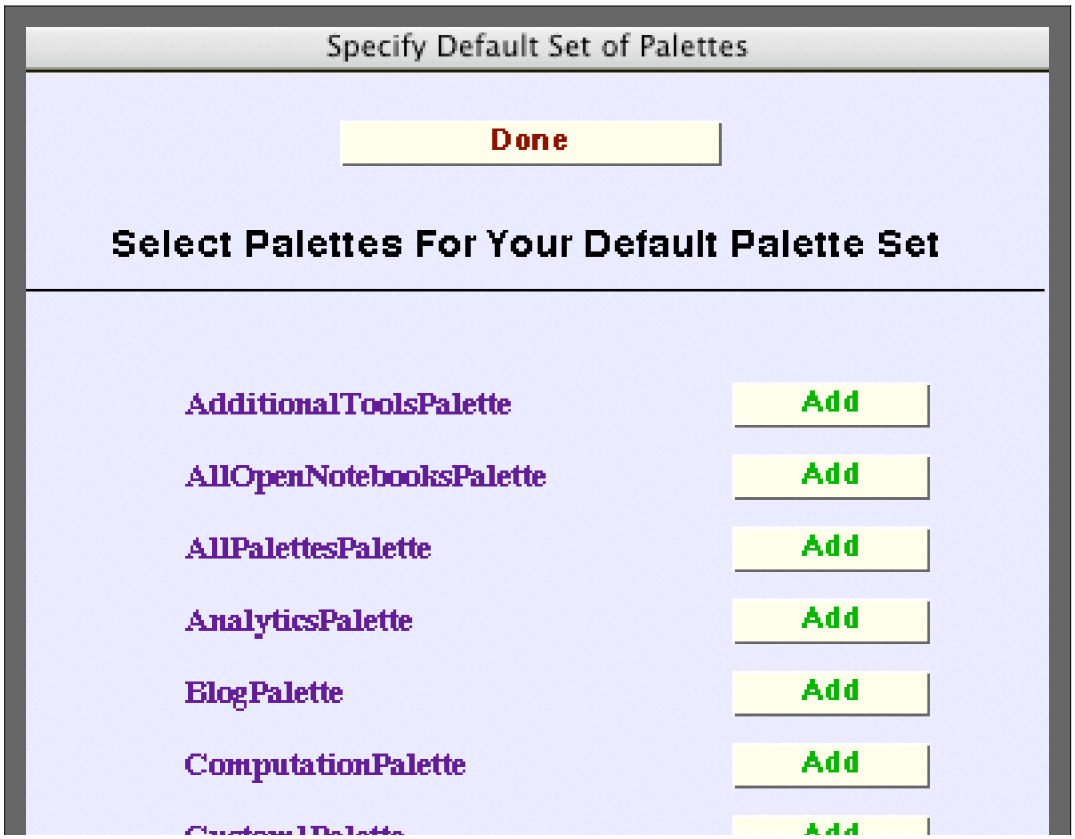
- **Open Default Set**

When the **Choose Default Set** button is clicked on the Palettes in the Default Set are opened and any Palettes that are not in the Default Set are closed. This button has the same effect as the **Default** Button near the top of the **WorkLife Tools** Palette.

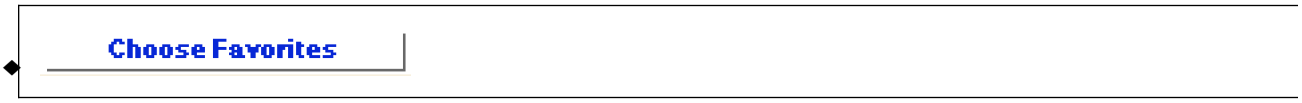
- **Choose Default Set**

The **Choose Default Set** Button opens up a dialog that allows you to add or delete fromt he list of Palettes that are specified as your default set.

The dialog looks like:



The Specify Default Set Dialog



The **Choose Favorites** button opens up a dialog that allows you to add or delete **Favorites** from the **Favorite Palettes** Palette.

The dialog looks like:

The Organize Favorite Palettes Dialog

Opens the **Favorites & Recent** Palette. This can also be opened programmatically by executing:

```
FavoritesAndRecentPalette[];
```

The ⊗ button closes the Palette if it is currently open.

> *This pair of buttons and the four pairs that follow are, by default, always present at the top of the* **Favorite Palettes** *Palette.*

Opens the **Diary List** Palette. This can also be opened programmatically by executing:

```
DiaryListPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Notebooks** Palette. This can also be opened programmatically by executing:

```
NotebooksPalette[];
```

The ⊗ button closes the Palette if it is currently open.

Opens the **Packages** Palette. This can also be opened programmatically by executing:

```
PackagesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

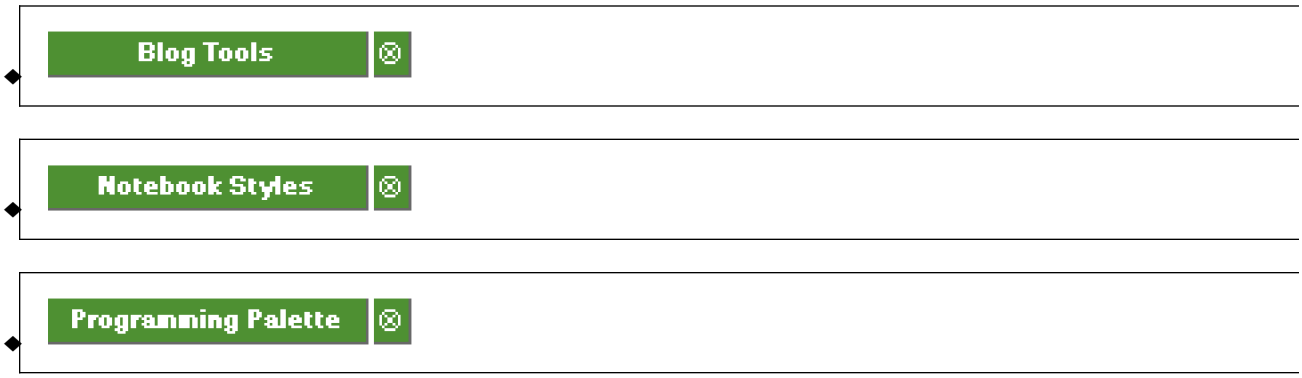
Opens the **OtherFiles** Palette. This can also be opened programmatically by executing:

```
OtherFilesPalette[];
```

The ⊗ button closes the Palette if it is currently open.

The buttons that follow this **Favorites** button are added or deleted to this Palette via the dialog that is opened via the

Organize button at the top of the Palette. In the case of this example there are three favorite Palettes in listed in the **Favorite Palettes** Palette.



Favorites & Recent Palette

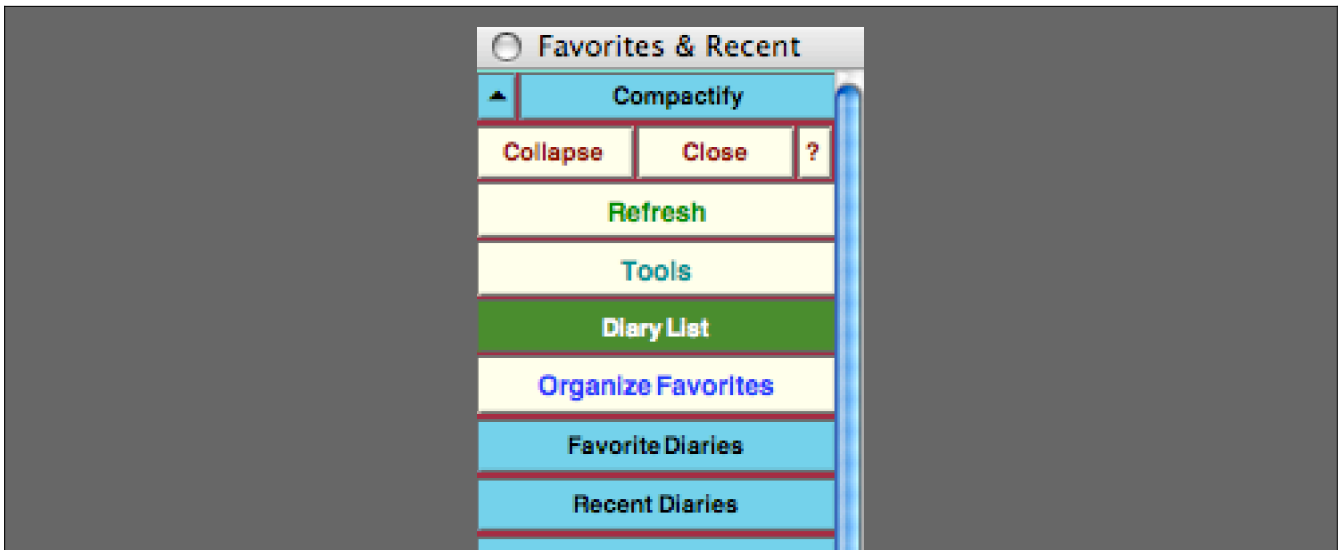
The Palette

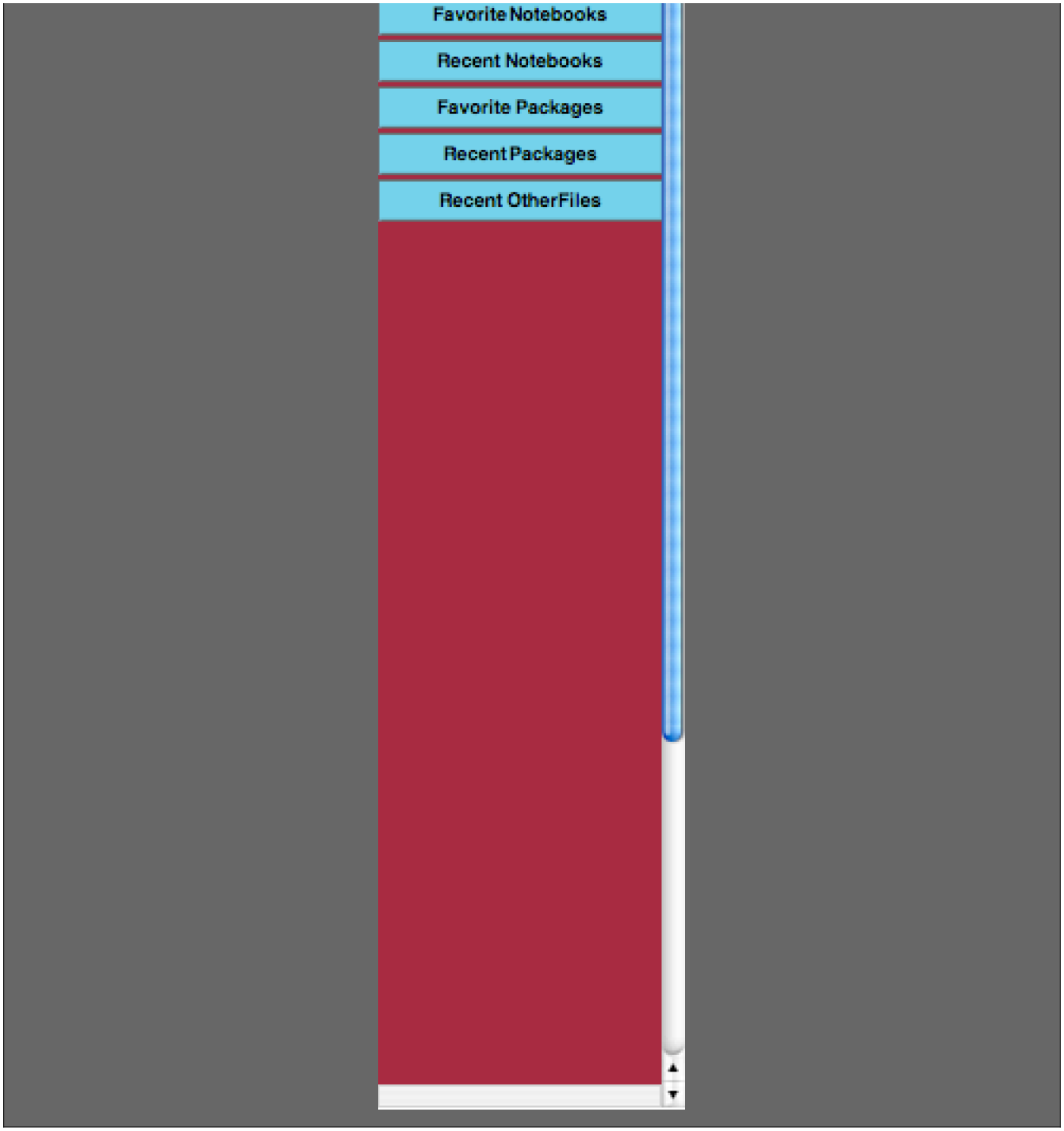
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Favorites & Recent** button on the All Palettes Palette, you will open the **Favorites & Recent** Palette. You can also open the Favorites & Recent Palette by executing:

```
FavoritesAndRecentPalette[];
```

The **Favorites & Recent** Palette keeps track of your favorite and recent Diaries, Notebooks, and Packages.

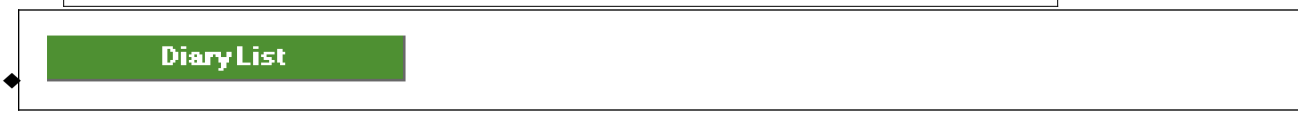




The Favorites & Recent Palette With It's Sub-palettes Compactified

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**



This button opens up the **Diary List** Palette. This can also be opened programmatically by executing:

```
DiaryListPalette[];
```

The Diaries listed in the **Diary List** Palette are those in the Default Diary Directory. This directory can be specified using the **Choose Directory** button from the **Diary List** Palette.

Opens the **Organized Favorites and Recent** dialog. This dialog allows you to delete items that are currently listed in the **Favorites & Recent** Palette as well as to specify what the maximum number of entries can be in the list for each category.

This button toggles open and closed the sub-palette containing the list of user-specified **Favorite Diaries** along with **Add** and **Delete** buttons for modifying the list. When one of these buttons is clicked, the associated Diary is opened and specified as the current Diary.

It is generally important to open notebooks (and Diaries) with buttons from the WorkLife FrameWork™ *Package. This is because when they are opened in this way additional information is added to the notebook—and entries are made in* A WorkLife FrameWork's™ *internal databases—so that they can be tracked in a variety of ways by the* WorkLife FrameWork™ *Package.*

The **Add** button adds the current Diary to this **Favorites Diaries** sub-palette. The **Delete** button removes the current Diary from this Palette if it is currently listed in it.

The following three buttons show the three Diaries that the user had assigned as Favorites when this document was being written.

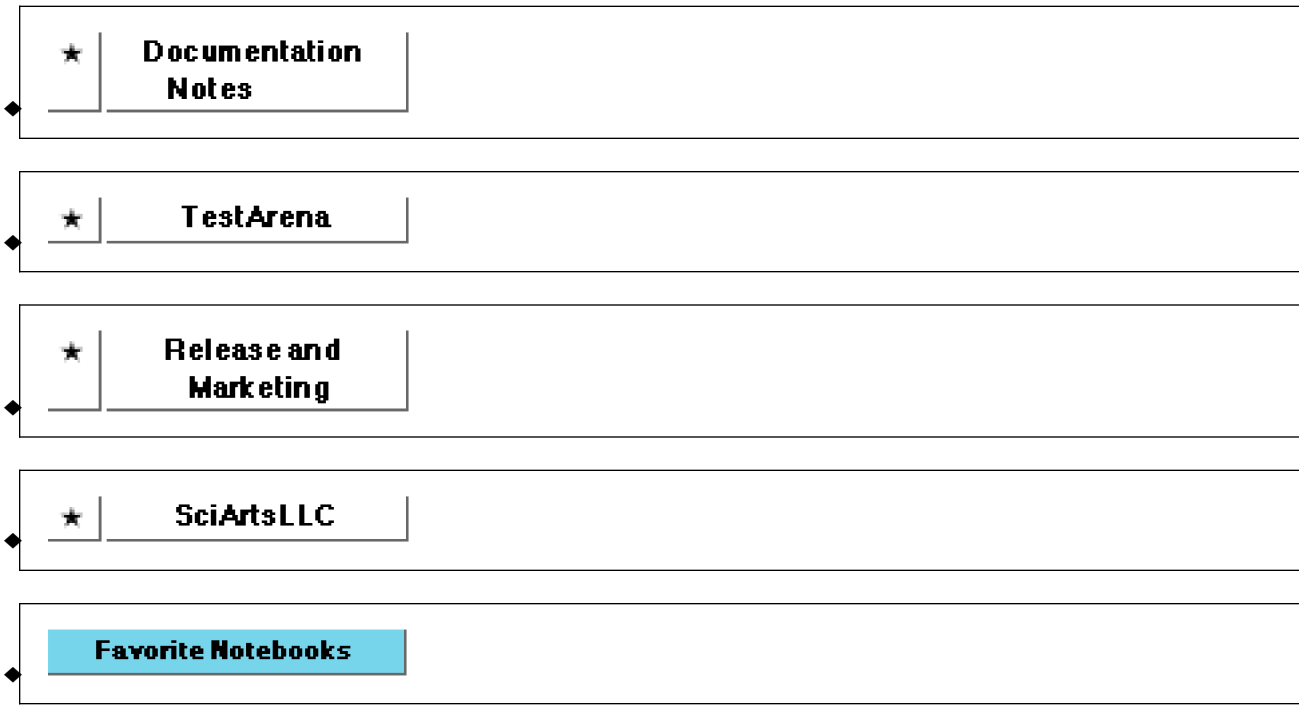
In each case the button with the Diary's name, when clicked on, opens that Diary and makes it the current Diary. The

button with the ★ makes that Diary the current one but does not open it up.

This button toggles open and closed the sub-palette containing the list **Recent Diaries**: they are listed in chronological order—with the most recently opened Diary first. When one of these buttons is clicked, the associated Diary is opened and specified as the current Diary.

The following four buttons show the four most recent Diaries that the user had opened when this document was being written.

In each case the button with the Diary's name, when clicked on, opens that Diary and makes it the current Diary. The button with the ★ makes that Diary the current one but does not open it up.



This button toggles open and closed the sub-palette containing the list of user-specified **Favorite Notebooks** along with

Add and **Delete** buttons for modifying the list. When one of these buttons is clicked, the associated Notebook is opened.

The **Add** button adds the current input Notebook to this **Favorites Notebooks** sub-palette. The **Delete** button removes the current input Notebook from this Palette if it is currently listed in it.

The following button shows the one Notebook that the user had assigned as a Favorite when this document was being written.

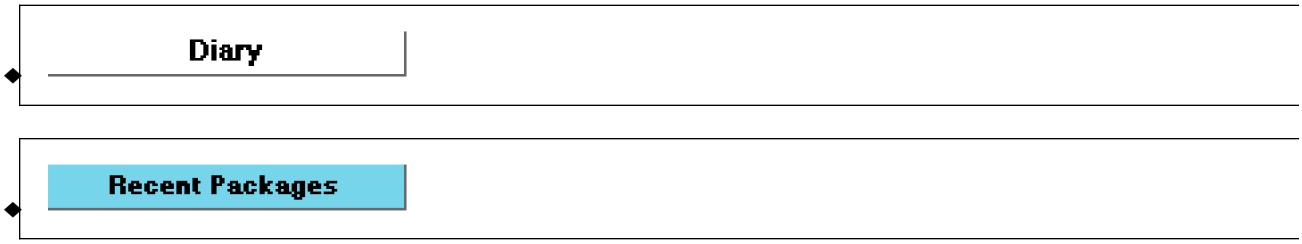
This button toggles open and closed the sub-palette containing the list **Recent Notebooks**: they are listed in chronological order—with the most recently opened Notebook first. When one of these buttons is clicked, the associated Notebook is opened.

The following four buttons show the four most recent Notebooks that the user had opened when this document was being written.

This button toggles open and closed the sub-palette containing the list of user-specified **Favorite Packages** along with **Add** and **Delete** buttons for modifying the list. When one of these buttons is clicked, the associated Package is opened.

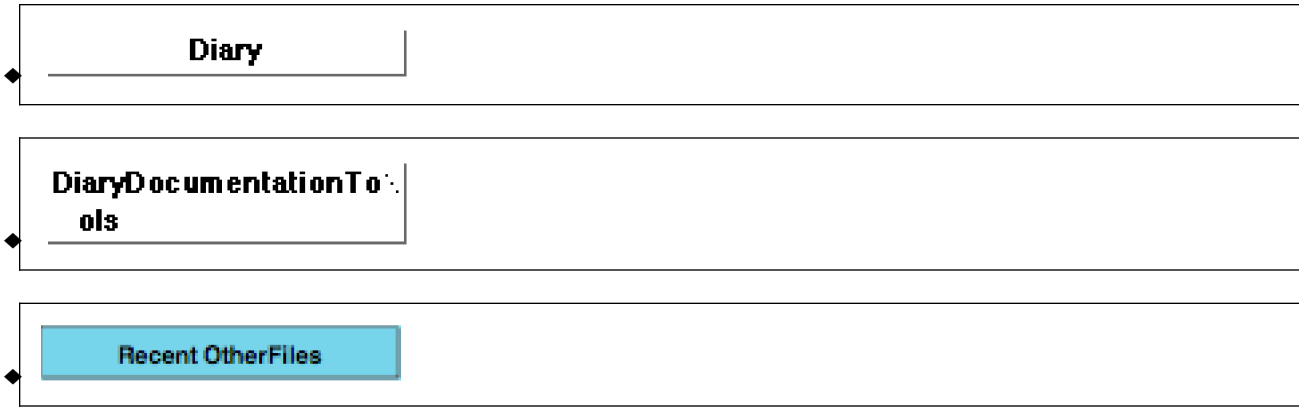
The **Add** button adds the current input Package to this **Favorites Packages** sub-palette. The **Delete** button removes the current input Package from this Palette if it is currently listed in it.

The following buttons show the one Package that the user had assigned as a Favorite when this document was being written. Clicking on that button would open the respective Package.



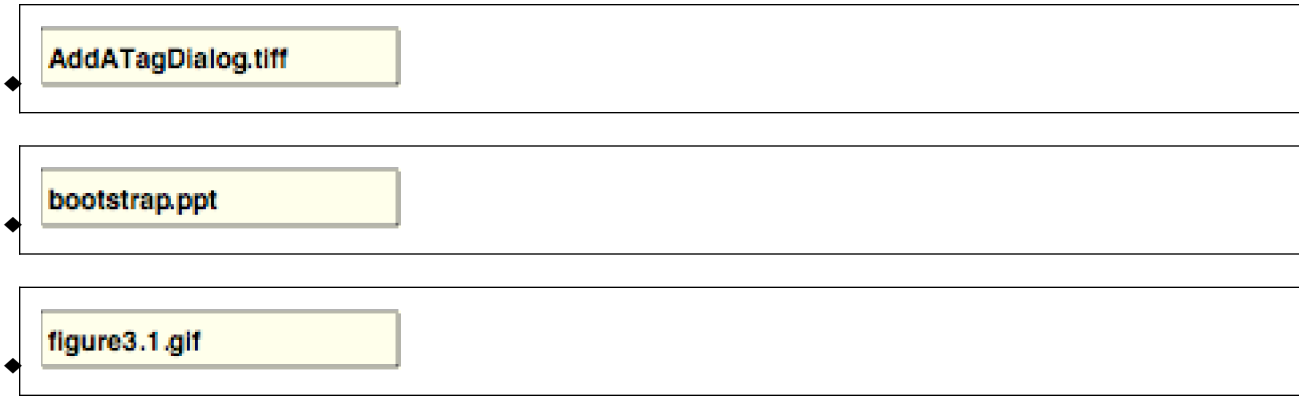
This button toggles open and closed the sub-palette containing the list **Recent Packages**: they are listed in chronological order—with the most recently opened Package first. When one of these buttons is clicked, the associated Package is opened.

The following two buttons show the two most recent Packages that the user had opened when this document was being written.



This button toggles open and closed the sub-palette containing the list **Recent OtherFiles**: they are listed in alphabetical order. When one of these buttons is clicked, the associated file is opened in its default application.

The following two buttons show the five most recent OtherFiles that the user had opened when this document was being written.



Report no 4 _Data.xls

stars.txt

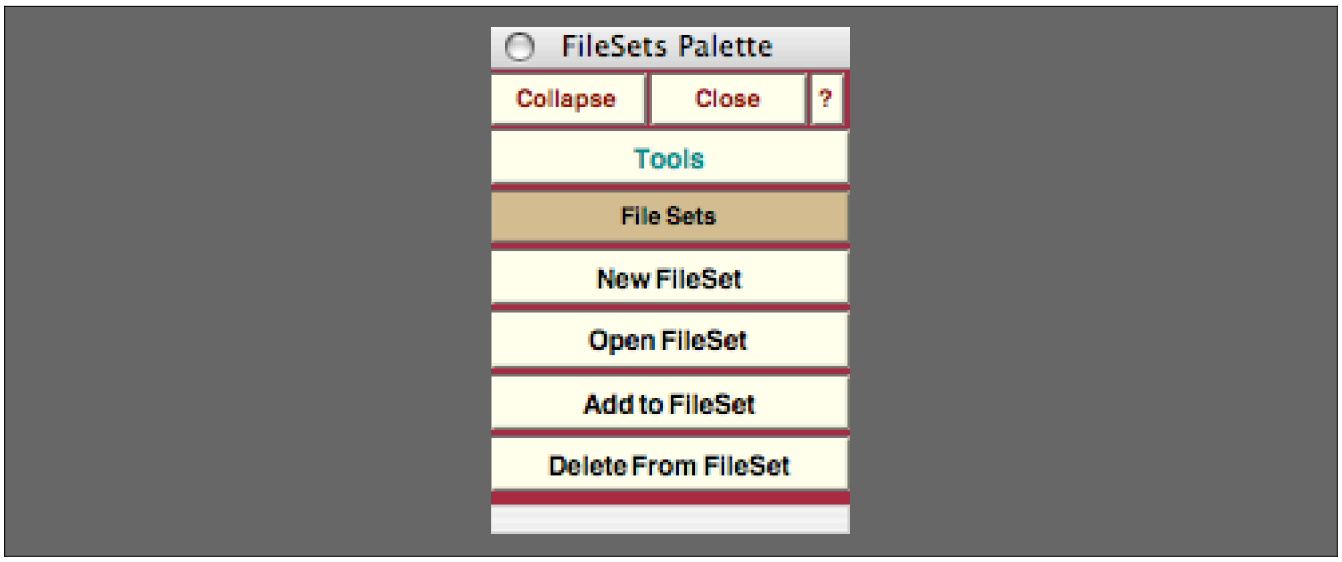
File Sets Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **File Sets** button on the All Palettes Palette, you will open the FileSets Palette. You can also open the FileSets Palette by executing::

```
FileSetsPalette[];
```



The FileSets Palette

The **FileSets** Palette provides administrative buttons for managing FileSets in the WorkLife FrameWork™ Package.

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

File Sets

New FileSet

This button opens a dialog that allows you to name and create a new FileSet. Use the buttons **Add to FileSet** and **Delete From FileSet** buttons below to add and deelete entries from a FileSet.

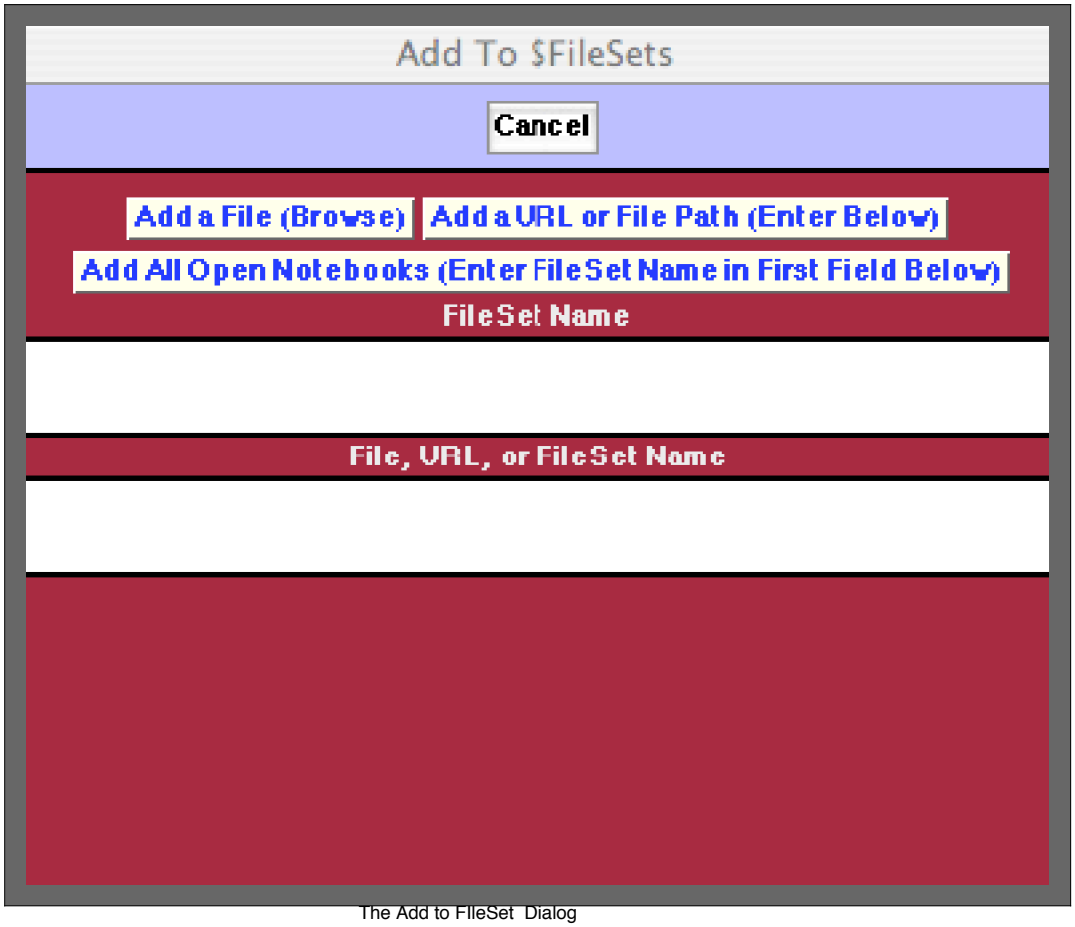
Open FileSet

This button opens a dialog with buttons that allow you to open one of your FileSets—opening the files in the FileSet in their default applications. A FileSet is a set of files (including URLs) that you add to and delete from using the following two buttons.

Add to FileSet

This button opens a dialog that allows you to add a file or URL to one of your FileSets. It also allows you to add all of your currently open Notebooks to an existing FileSet (to create a new FileSet use the **New FileSet** button above).

This dialog looks like:

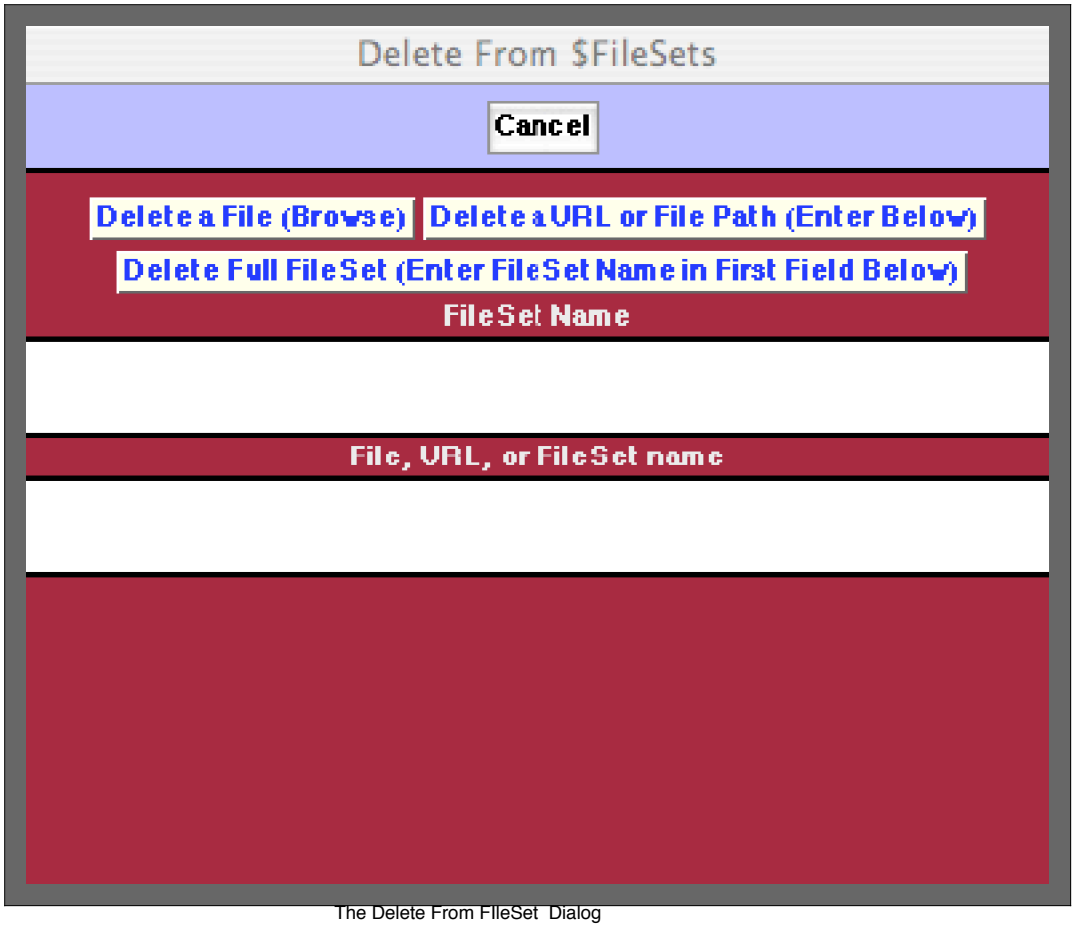


The Add to FIleSet Dialog

Delete From FileSet

This button opens a dialog that allows you to delete a file from one of your FileSets. It also allows you to delete an entire existing FileSet.

This dialog looks like:



The Delete From FIleSet Dialog

Form Templates Palette

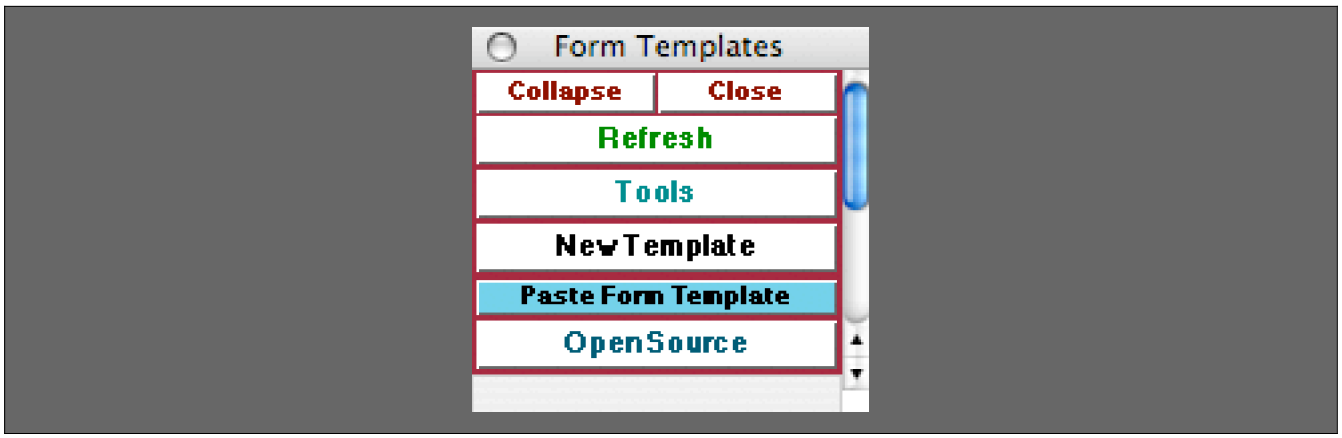
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Form Templates** button on the All Palettes Palette, you will open the Form Templates Palette. You can also open the Form Templates Palette by executing:

```
FormTemplatesPalette[];
```

The **Form Templates** Palette allows you to create Templates that can be easily reused to paste into Notebooks. The way that a template is created is to use the **New Template** button to name a template which is created from the current selection in the current InputNotebook.



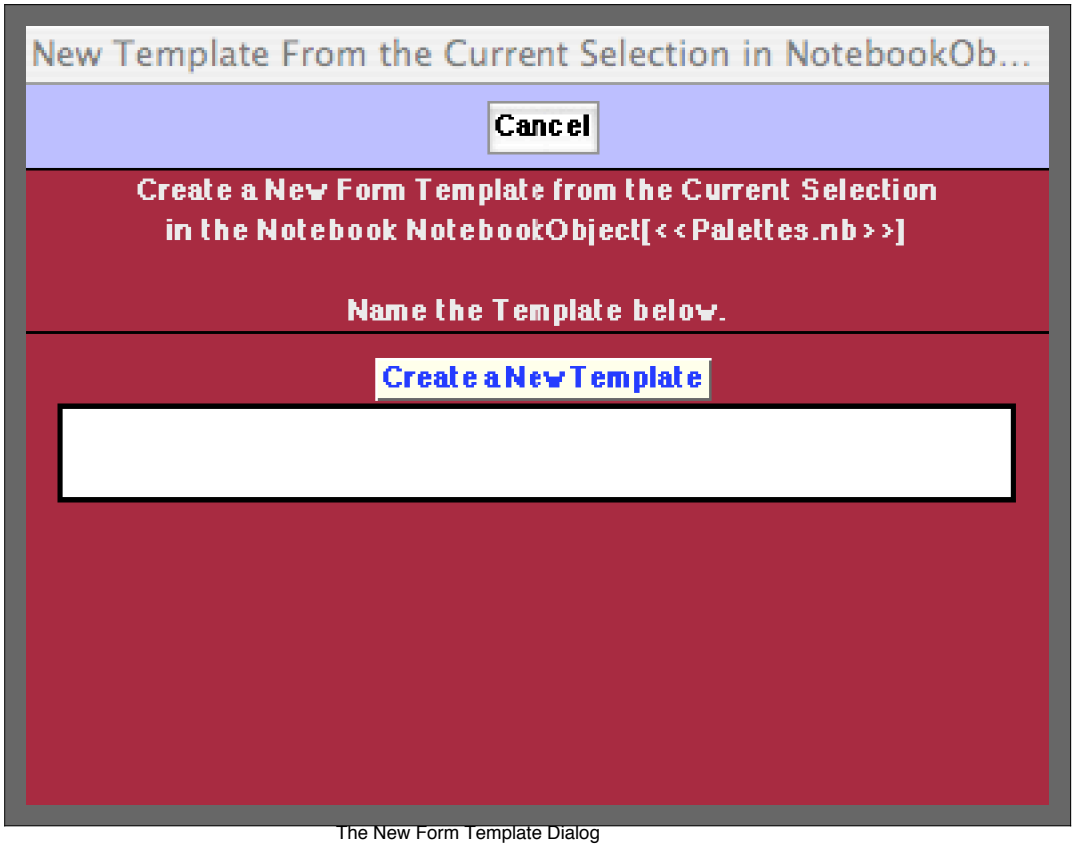
The Form Templates Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

New Template

The New Template button opens the following dialog which is used to name the template that will be created from the current selection in the current InputNotebook.



The New Form Template Dialog

Paste Form Template

Opens and closes the **Paste Form Template** sub-palette that contains buttons corresponding to all of the templates that the user has created..

PaletteDoc

In this example there is a single template that has been named **PaletteDoc**. This corresponds to a set of cells that will be pasted into the InputNotebook when the **PaletteDoc** button is clicked on.

Notebooks Palette

The Palette

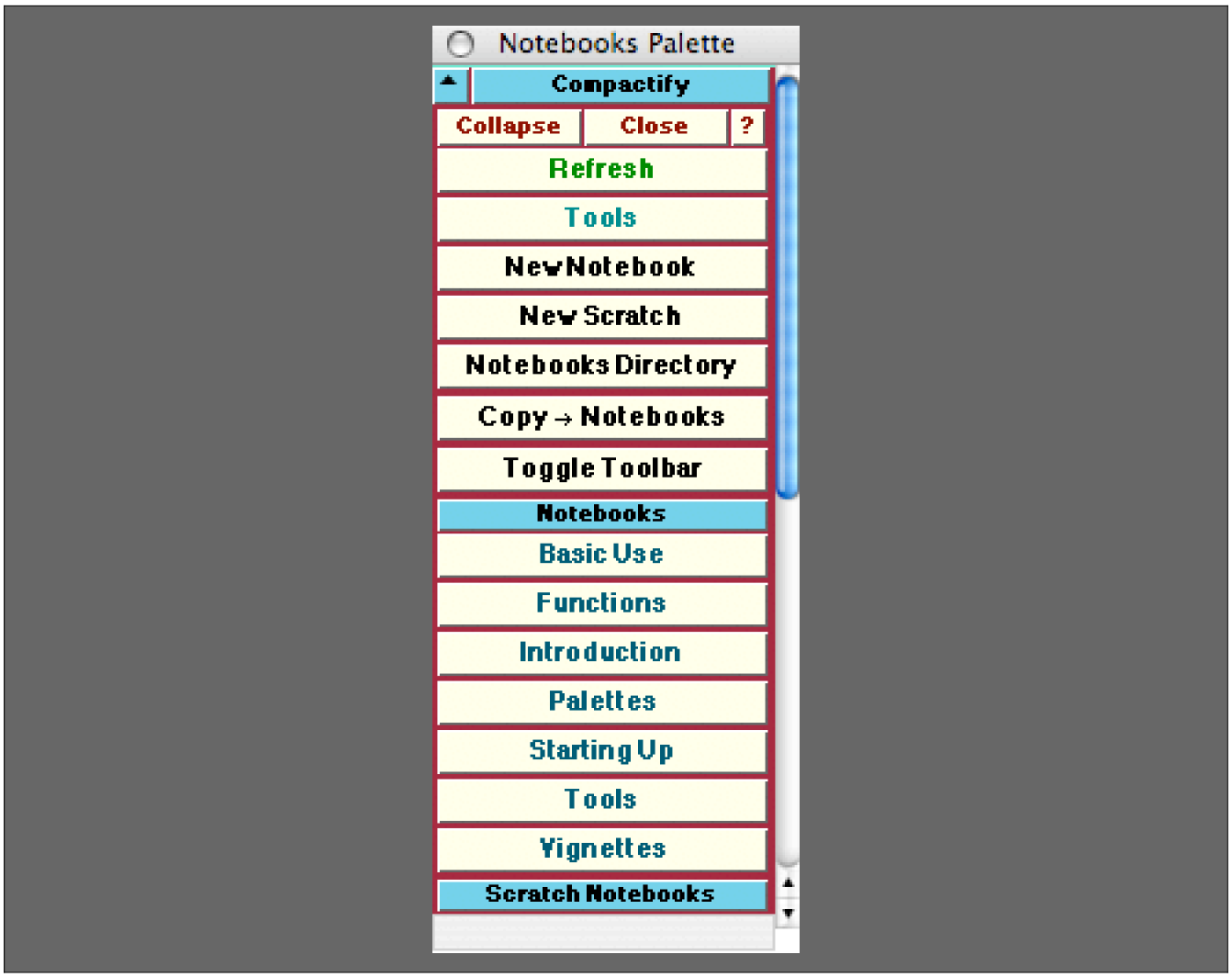
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Notebooks** button on the All Palettes Palette, you will open the **Notebooks** Palette. You can also open the Notebooks Palette by executing:

```
NotebooksPalette[];
```

The Notebooks Palette displays the notebooks that reside in the Notebooks subdirectory of the current Diary notebook. In addition to this, the Palette also has the administrative buttons for creating a new notebook or moving a chosen notebook into the current Diary's Notebooks directory.

Whenever a Diary is opened this Palette is refreshed to reflect the contents of the new Diary's Notebooks subdirectory.



The Notebooks Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

NewNotebook

The **New Notebook** button opens up a dialog box for creating a new notebook. This dialog has a field for entering the name that you wish to give the new notebook. When the notebook is created it will be placed in the Notebooks

subdirectory that is within the Current Diary's directory.

To see what the current Diary is you can click on the **Current Diary** button in the **Diary Access** Palette.

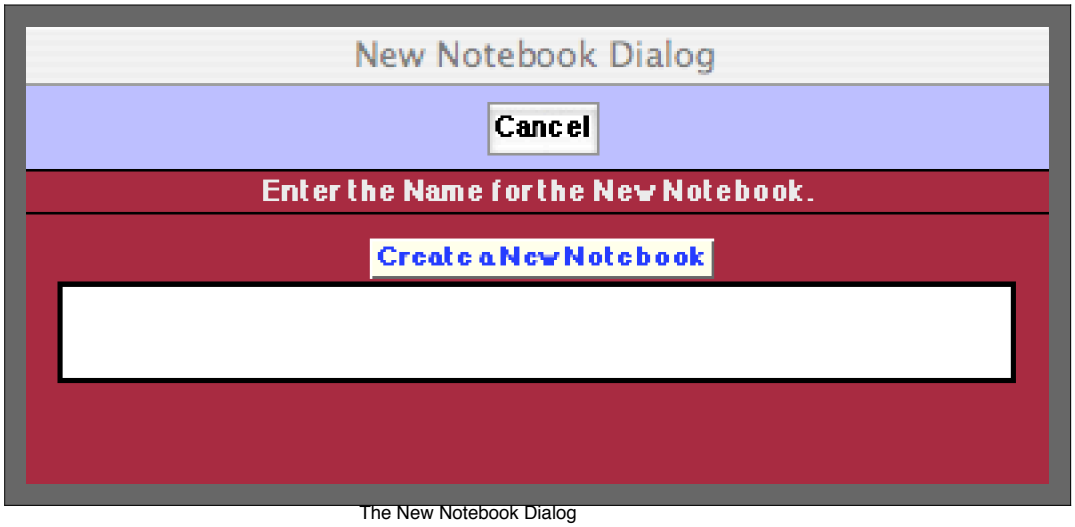
The current Diary's directory is given by

```
CurrentDiaryNotebookDirectory[]
```

And the Notebooks subdirectory is given by

```
DiaryNotebooksDirectory[]
```

The **New Notebook** dialog looks like this:



The New Notebook Dialog

When a Notebook is created it has a ToolBar at its top along with the name that you assigned to it in a Title cell just below the Toolbar. The Title cell can, of course, be removed or edited, while the Toolbar cannot be deleted directly. The Toolbar can however be either toggled open/closed through the function `ToggleSaveBackupToolbarCell[nb]` where nb is the NotebookObject of the Notebook. Also the functions `DeleteSaveBackupToolbarCell[nb]` and `AddSaveBackupToolbarCell[nb]` can be used to completely delete the `SaveBackupToolbarCell` or to replace it if it has been removed. The `SaveBackupToolbarCell` looks like:

The SaveBackupToolbarCell

The **New Scratch** button opens up a dialog box for creating a new scratch notebook. This dialog has a field for entering the name that you wish to give the new scratch notebook. When the scratch notebook is created it will be placed in the Scratch subdirectory of the Notebooks subdirectory that is within the Current Diary's directory.

A scratch notebook is a convenient place to put *Mathematica* calculations that are free form and unstructured.

To see what the current Diary is, you can click on the **Current Diary** button in the **Diary Access** Palette.

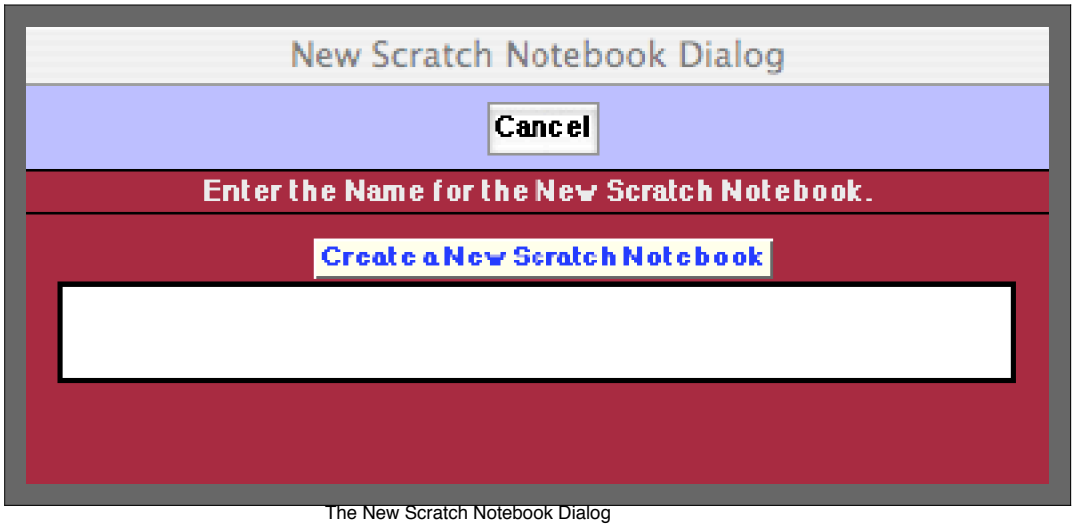
The current Diary's directory is given by:

```
CurrentDiaryNotebookDirectory[]
```

And the Scratch subdirectory of Notebooks subdirectory is given by:

DiaryScratchNotebooksDirectory[]

The **New Scratch** dialog looks like this:



The New Scratch Notebook Dialog

When creating a scratch notebook you can repeatedly use the same name for the notebook. Each time you do this a new scratch notebook will be created with the name that you have given along with an integer appended to its end. The integer starts with 1 and then increments by 1 each time you create a new scratch notebook with the given base name.

Notebooks Directory

The **Notebooks Directory** button opens up the Notebooks subdirectory that is within the Current Diary's directory.

Copy → Notebooks

The **Copy→Notebooks** button opens up a browsing window that allows you to select a notebook. A copy of this notebook will then placed in the Notebooks subdirectory that is within the Current Diary's directory.

Toggle Toolbar

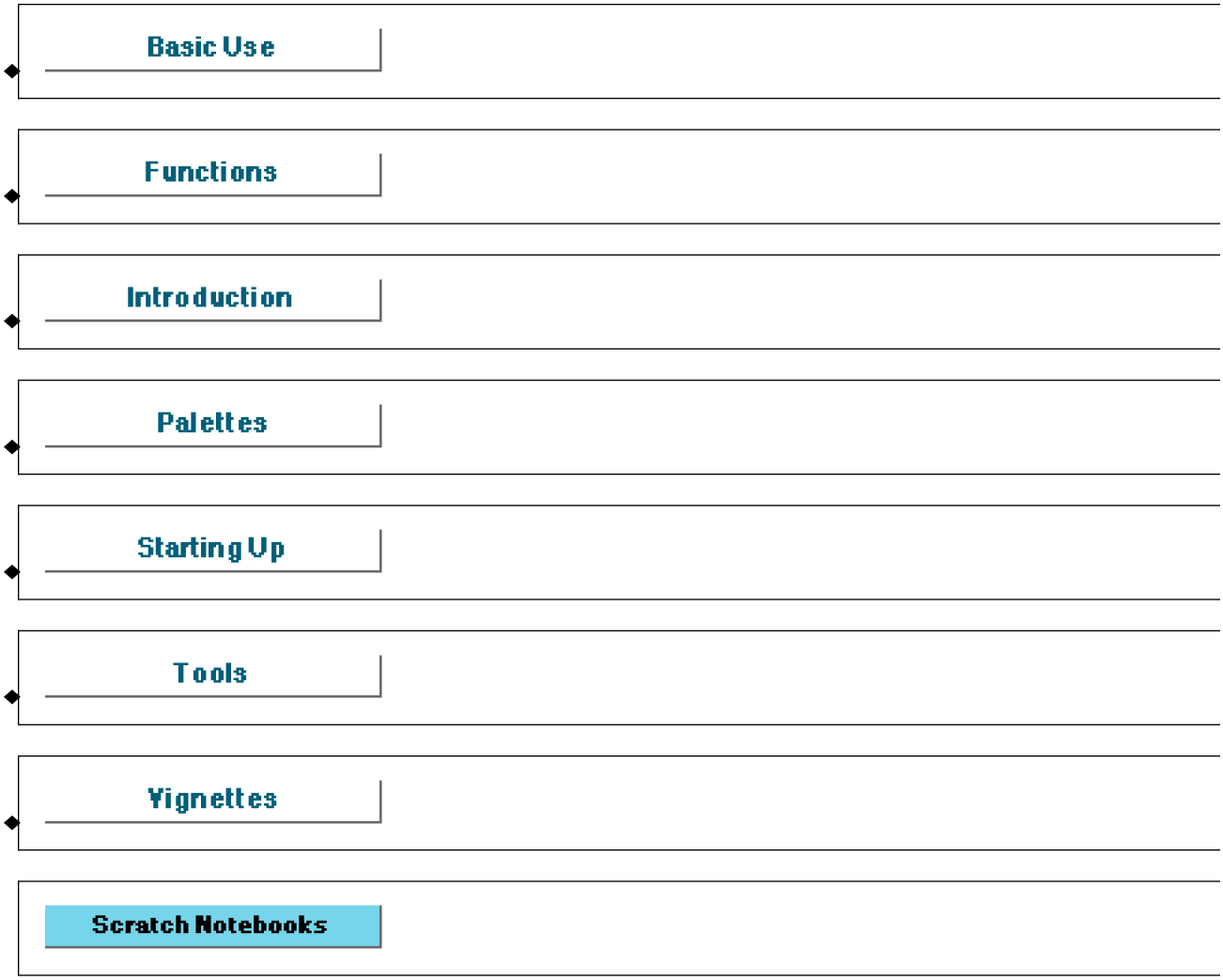
This toggles Opened and Closed the `SaveBackupToolbarCell` at the top of a Notebook. It has this effect on the current `InputNotebook`. It has no effect on Diaries.

Notebooks

This button toggles open and closed the sub-palette containing the list of notebooks that are in the Notebooks subdirectory of the Current Diary's directory. When one of these buttons is clicked, the associated notebook is opened.

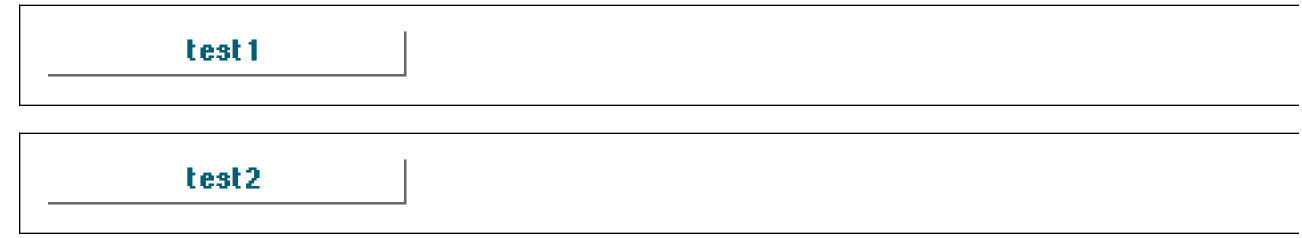
The following seven buttons show the seven files that were in the current Diary's Notebooks directory at the time that this document was being written.

> *It is generally important to open notebooks (and Diaries) with buttons from the* WorkLife FrameWork™ *Package. This is because when they are opened in this way additional information is added to the notebook so that it can be tracked in a variety of ways by the* WorkLife FrameWork™ *Package.*



This button toggles open and closed the sub-palette containing the list of scratch notebooks that are in the Scratch subdirectory of the Notebooks subdirectory of the Current Diary's directory. When one of these buttons is clicked, the associated scratch notebook is opened.

In this case there are three scratch Notebooks with the base name `test`. Each was created by entering `test` into the **New Scratch** dialog. The Scratch Notebook **test3** is the one that was most recently created.



test3

Notebook Styles Palette

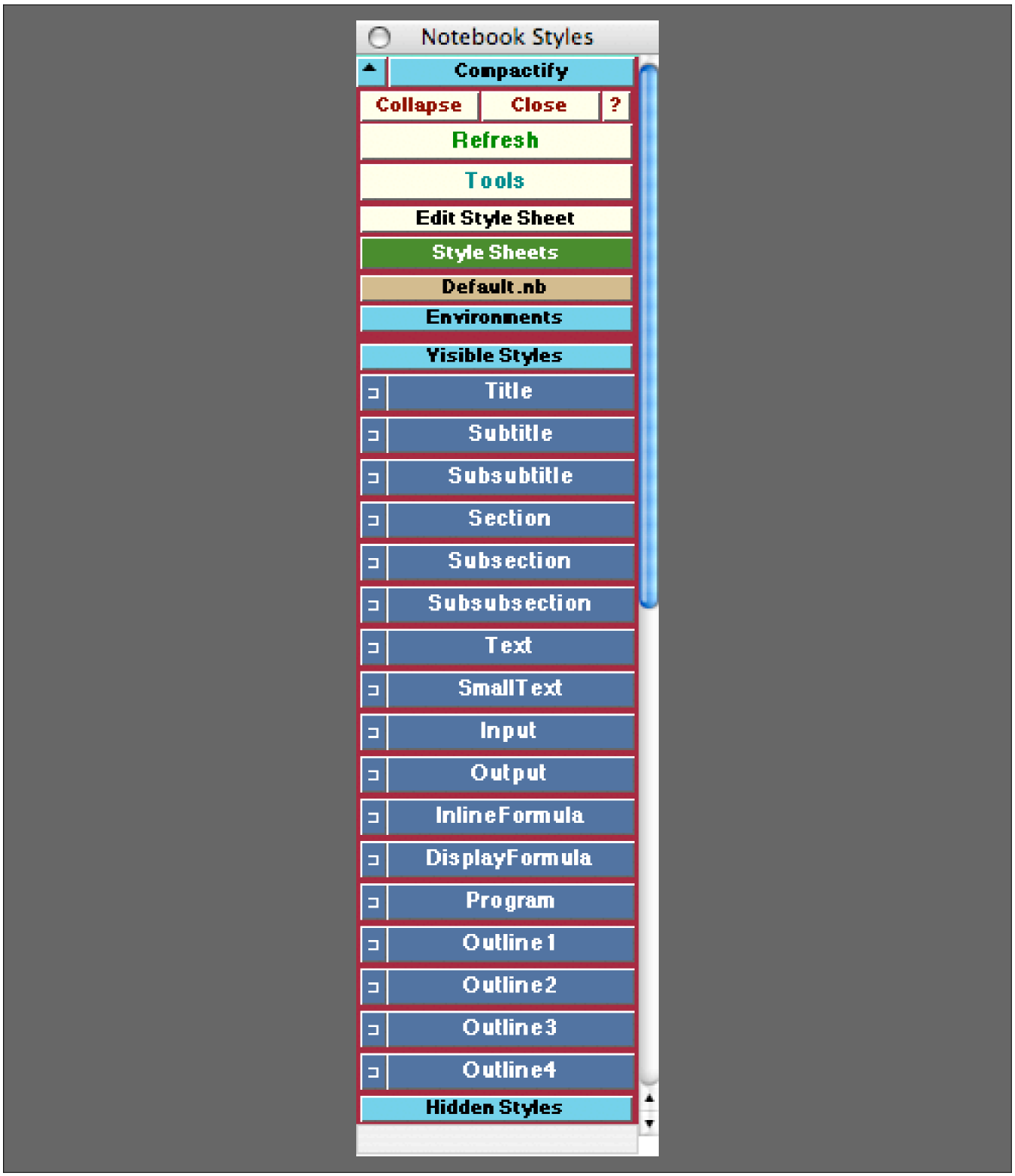
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Notebook Styles** button on the All Palettes Palette, you will open the Notebook Styles Palette. You can also open the Notebook Styles Palette by executing:

```
NotebookStylesPalette[];
```

This Palette provides access to Style Definitions for the Style Sheet of the notebook that you are currently working in.



The Notebook Styles Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Edit Style Sheet

Opens the style sheet of the InputNotebook so that you can edit it.

Style Sheets

Opens the **StyleSheets** Palette to allow you to change the style sheet of the InputNotebook.

Default.nb

Indicates the name of the Style Sheet that the style names in the Palette are obtained from. To change the Style sheet to that of the current InputNotebook, click the Refresh button at the top of the Palette.

Environments

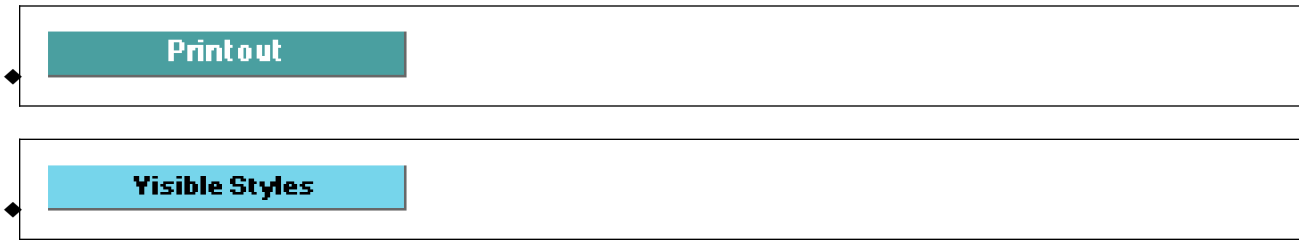
This button toggles open and closed the sub-palette containing the list of Screen Style **Environments** which follow. Clicking on any of the Buttons in this sub-palette will change the Screen Style Environment to the one indicated.

Working

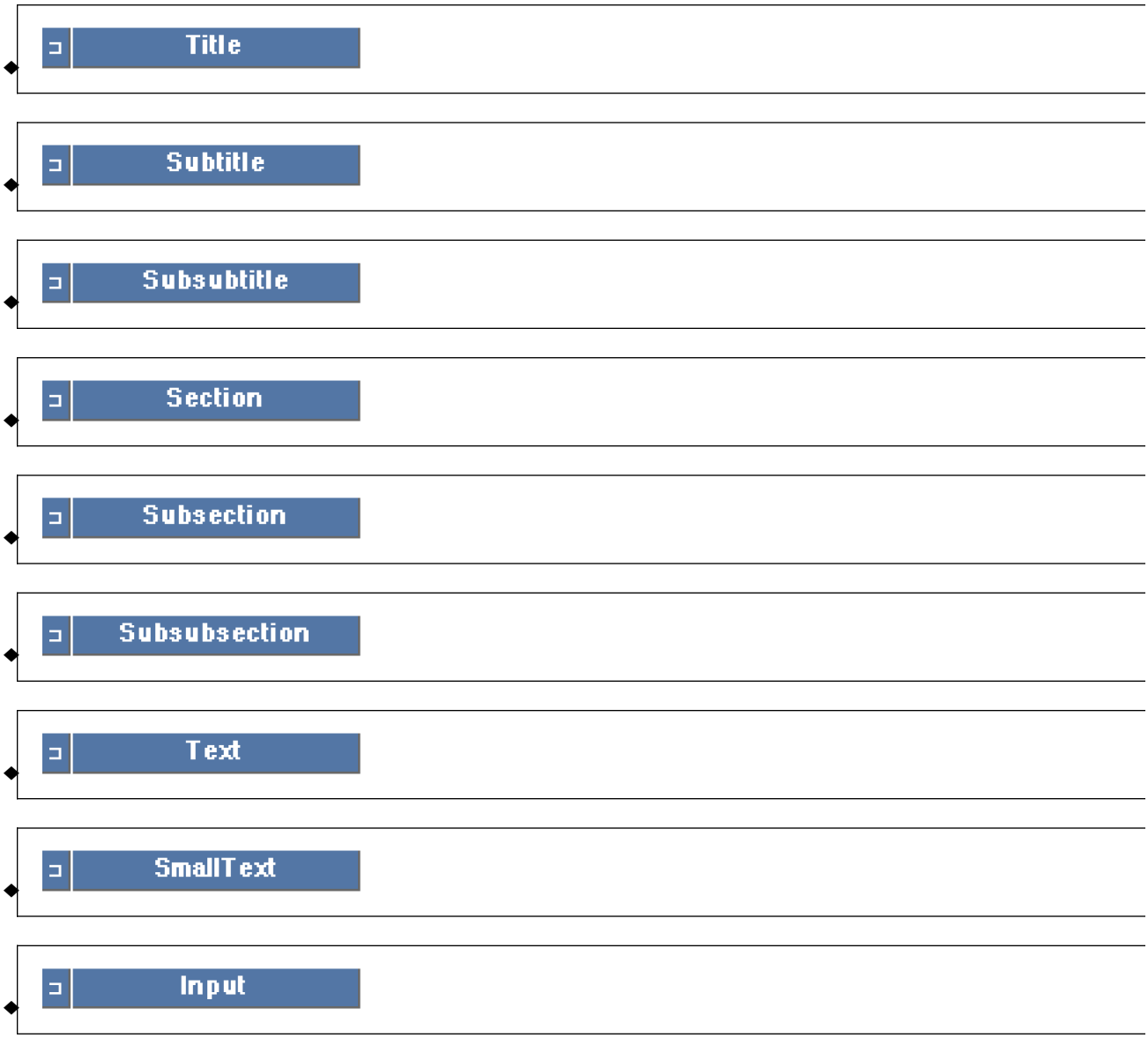
Presentation

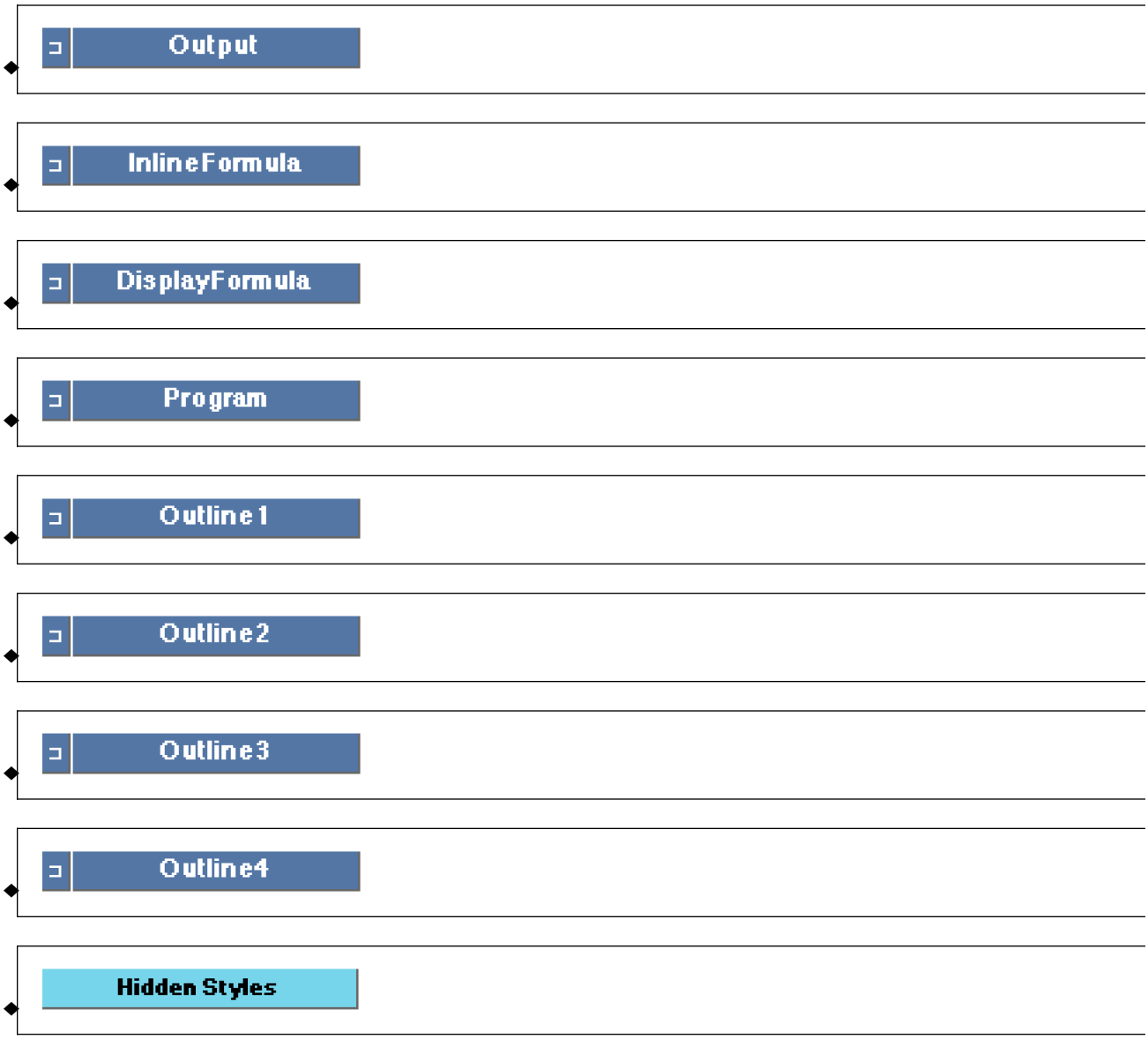
Condensed

SlideShow



This button toggles open and closed the sub-palette containing the list of **Visible Styles** which follow. These are the style that are listed in the Format ▹ Style menu. Clicking on any of the Buttons in this sub-palette with a named style will change the current selection in the InputNotebook to that style. Clicking on the smaller button to it's left with the symbol ⊐ will create a new Cell with the indicated Style below the current selection in the InputNotebook.





This button toggles open and closed the sub-palette containing the list of **Hidden Styles** which follow it (but which are suppressed in this documentation. These are the style that are *not* listed in the Format ▹ Style menu. Clicking on any of the Buttons in this sub-palette with a named style will change the current selection in the InputNotebook to that style. Clicking on the smaller button to it's right with the symbol ⊐ will create a new Cell with the indicated Style below the current selection in the InputNotebook.

Organizations Palette

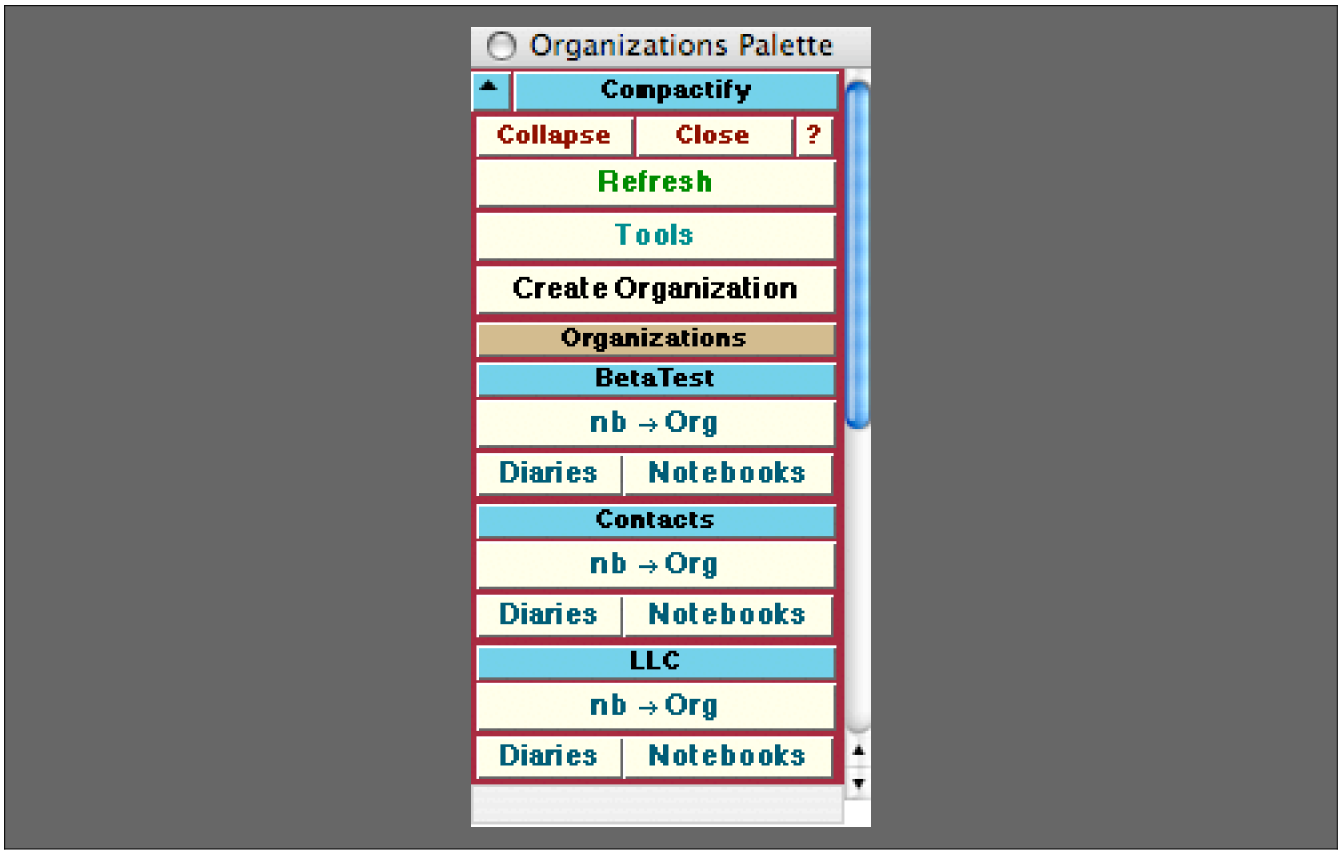
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Organizations** button on the All Palettes Palette, you will open the Organizations Palette. You can also open the Organizations Palette by executing:

```
OrganizationsPalette[];
```

This Palette provides access to those Organizations that you have created. Organizations can be created through the function CreateOrganization.



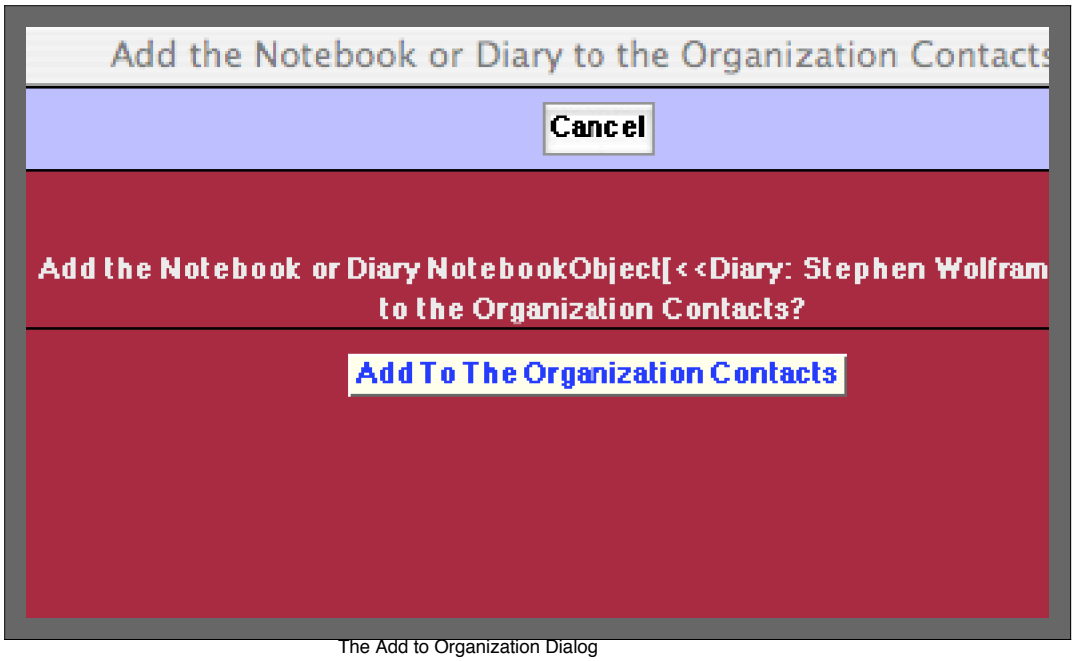
The Organizations Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

- **Create Organization**

Opens up a dialog to allow you to **Create** a new **Organization**. Once created buttons corresponding to that organization will appear on the palette. The following is and example of this dialog:

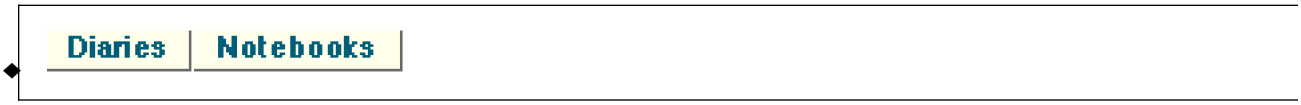


The Add to Organization Dialog

- **Organizations**
- **BetaTest**

This opens and closes the sub-palette containing the following buttons that manage the indicated Organization, in this case the **BetaTest** Organization. For this and the remaining two sets of buttons following them, the user has previously created the organizations **BetaTest**, **Contacts**, and **LLC**.

This button adds the current InputNotebook (**nb**, which can be a diary or other notebook) to the given Organization (in this case, **BetaTest**). When the button is clicked on a dialog opens asking whether you do indeed want to add the notebook to that Organization.



The **Diaries** button opens a Popup Palette listing the Diaries in the Organization (in this case, **BetaTest**). The **Notebooks** button opens a Popup Palette listing the Notebooks in the Organization.

The following buttons are the corresponding ones for the additional Organizations in the example of the Organizations palette above..

OtherFiles Palette

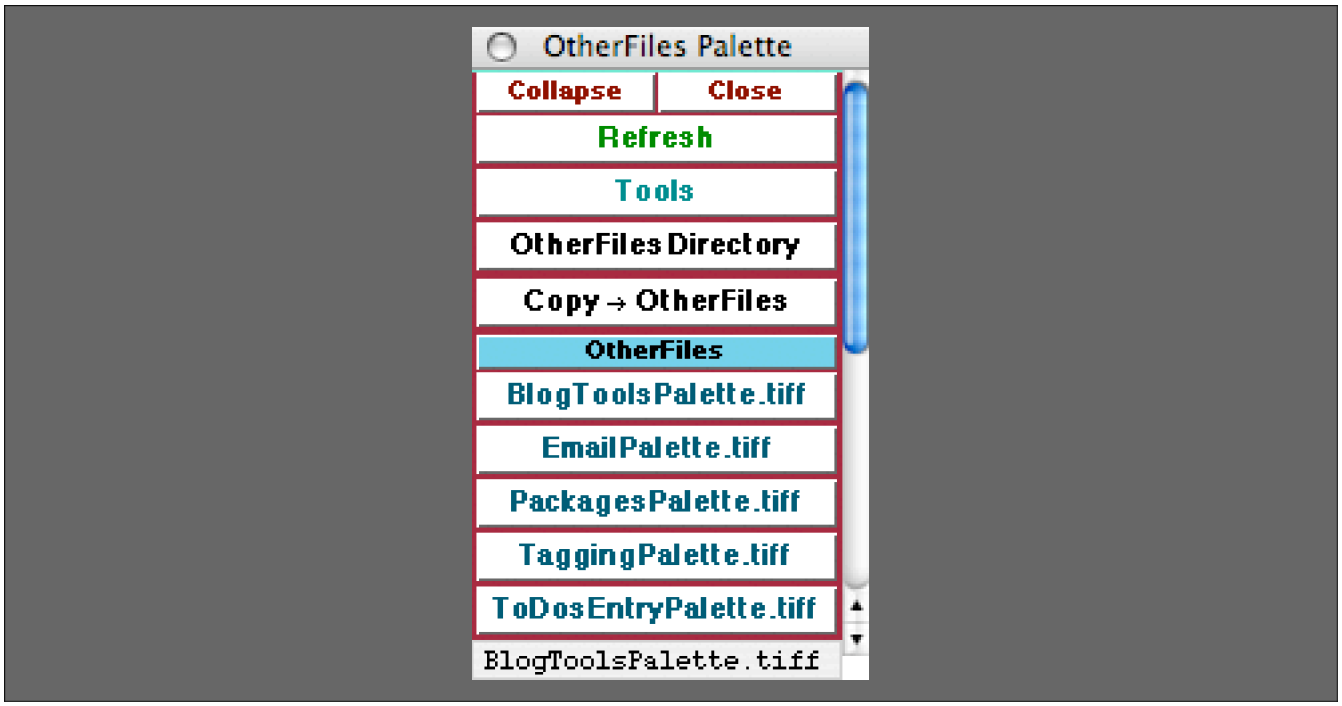
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **OtherFiles** button on the All Palettes Palette, you will open the **OtherFiles** Palette. You can also open the OtherFiles Palette by executing:

```
OtherFilesPalette[];
```

The **OtherFiles** Palette lists the files that are in the current Diary's OtherFiles subdirectory. This Palette is refreshed when a different Diary is made the current Diary.



The OtherFiles Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

OtherFiles Directory

Opens the current Diary's OtherFiles subdirectory

Copy → OtherFiles

The **Copy→OtherFiles** button opens up a browsing window that allows you to select a file. A copy of this file will then placed in the OtherFiles subdirectory that is within the Current Diary's directory.

OtherFiles

This button toggles open and closed the **OtherFiles** sub-palette containing the list of other files that are in the OtherFiles subdirectory of the Current Diary's directory. When one of these buttons is clicked, the associated file is opened in its default application. The buttons below show the three files in the current Diary's directory. Because they are `.tiff` files they will open up in the computer's default image editing program.

BlogToolsPalette.tiff

An example of a button that would open the file BlogToolsPalette.tiff which resides in OtherFiles subdirectory of the current Diary's directory.

EmailPalette.tiff

An example of a button that would open the file EmailPalette.tiff which resides in OtherFiles subdirectory of the current Diary's directory.

OtherFilesPalette.tiff

An example of a button that would open the file OtherFilesPalette.tiff which resides in OtherFiles subdirectory of the current Diary's directory.

Package Programming Palette

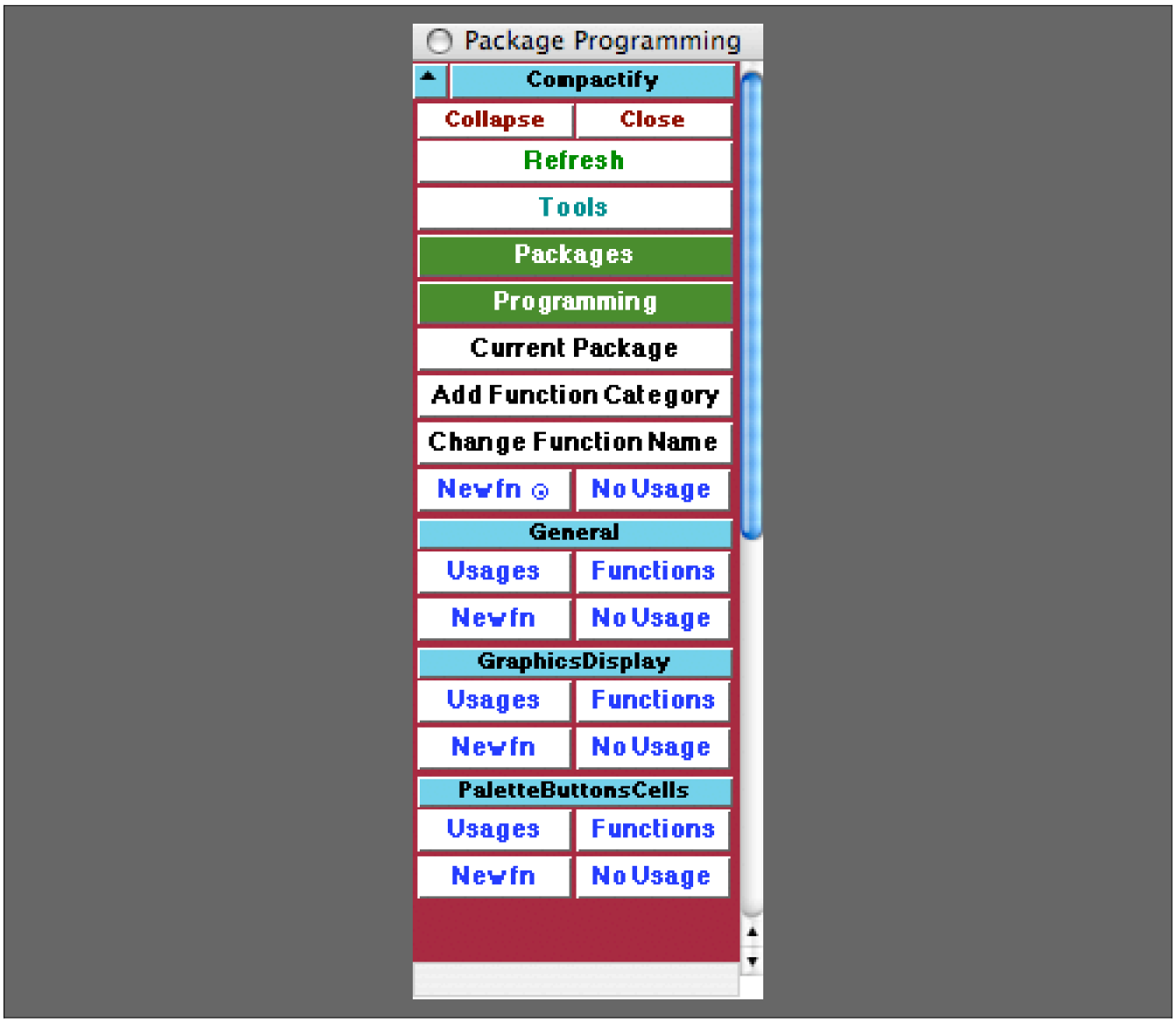
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Package Programming** button on the All Palettes Palette, you will open the Package Programming Palette. You can also open the Package Programming Palette for the `$CurrentPackageNotebook`, if it is open, by executing:

```
PackageProgrammingPalette[];
```

The Package Programming Palette has various administrative buttons for creating function templates in the current package. Functions are generally placed into categories by the user. This is a means of organizing one's code and navigating back and forth within a package's programming notebook. In the illustrative example in the picture below there are three categories: **General**, **GraphicsDisplay**, and **PaletteButtonCells**. Every package notebook has a General category. In this illustration the user has created the two additional categories.



The Package Programming Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Packages

Opens the **Packages** Palette. This Palette can also be opened by executing:

```
PackagesPalette[];
```

Programming

Opens the **Programming** Palette. This Palette can also be opened by executing:

```
ProgrammingPalette[];
```

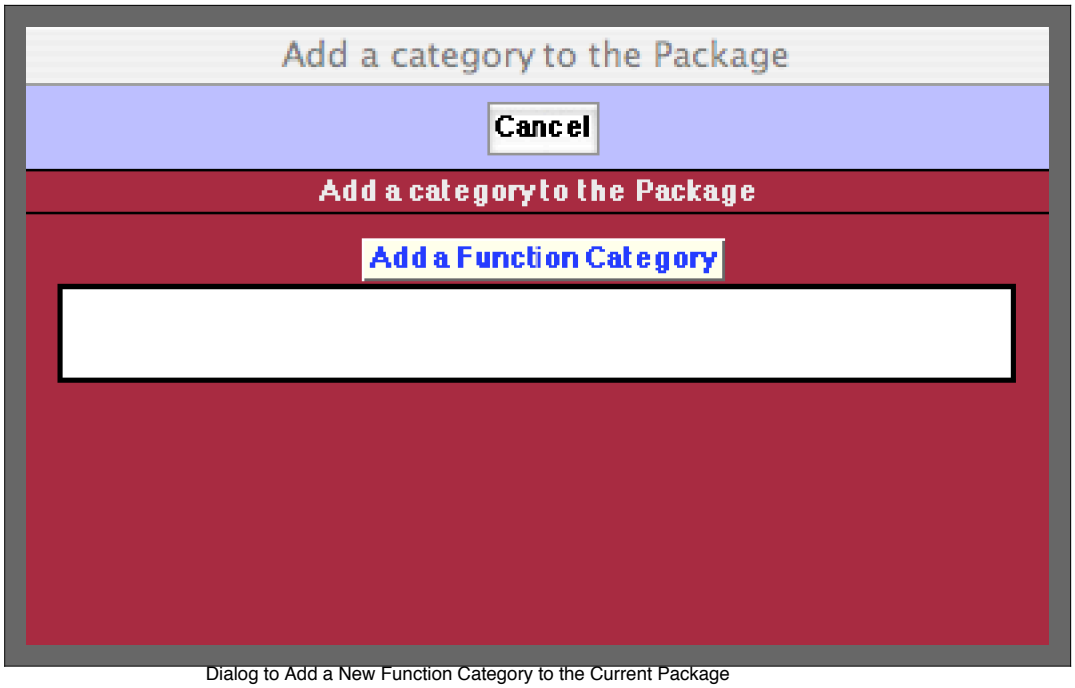
Current Package

Opens up the **Current Package** if there is one. The current package programming notebook file is given by `$CurrentPackageNotebookFile`. If the current package programming notebook is open its notebook object is given by `$CurrentPackageNotebook`.

Add Function Category

Opens up a dialog to add a new function category to the current package.

The dialog that is opened looks like:

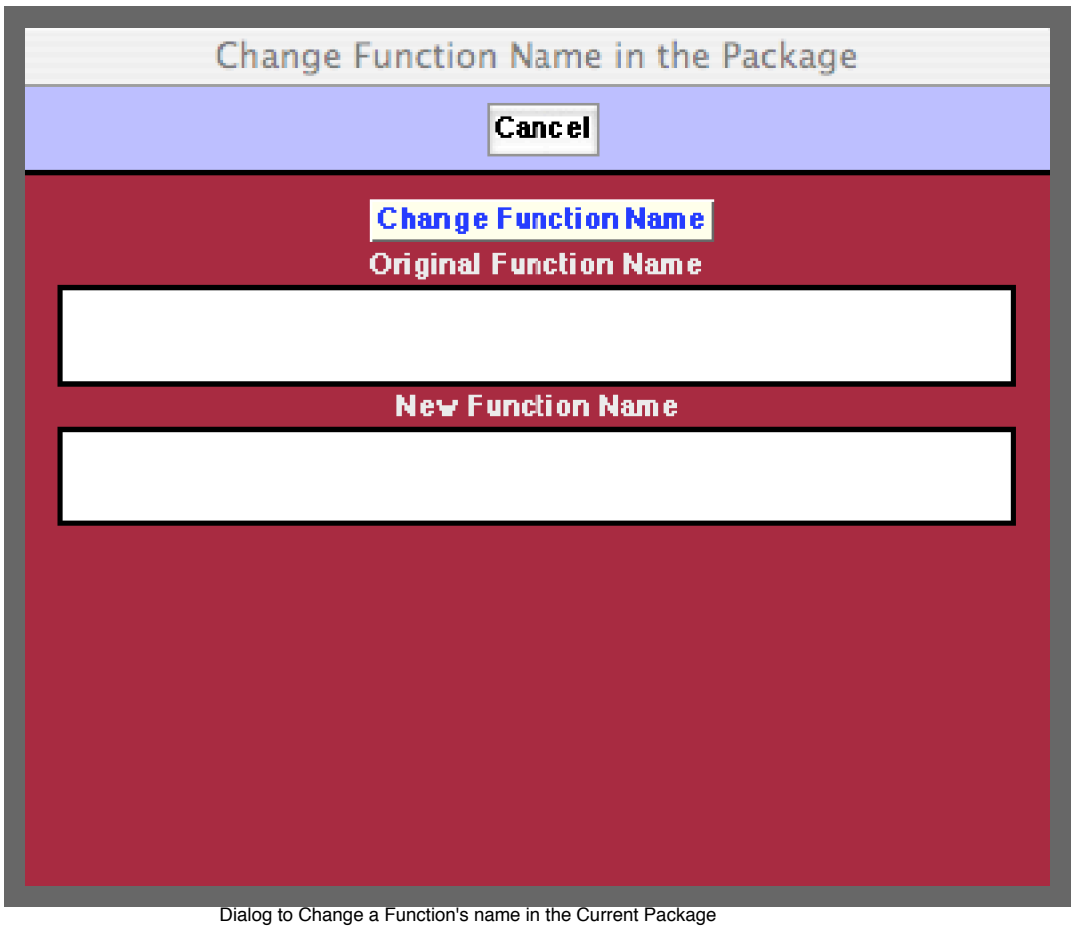


Dialog to Add a New Function Category to the Current Package

Change Function Name

The **Change Function Name** button opens a dialog that allows you to change a function's name in the current package. When the function's name has been changed via clicking the **Change Function Name** button in the dialog, a Find and Replace window opens to allow you to search and replace instances of the function name in the package.

The initial dialog that is opened looks like:

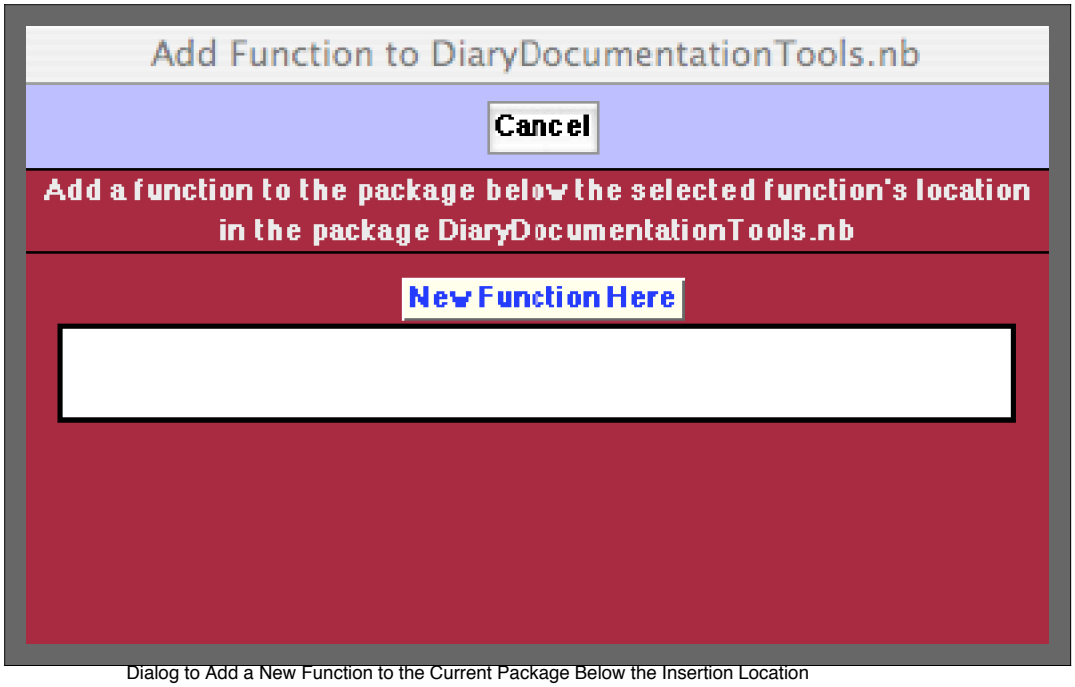


Dialog to Change a Function's name in the Current Package

Newfn ⊙ **No Usage**

The **Newfn ⊙** button opens a dialog to name and create a new function template below the function code that the cursor is currently located in. A template for a usage message is also created in the usage messages section for the category that the function has been created in so that the function will be exported from the package.

The dialog that is opened looks like:



Dialog to Add a New Function to the Current Package Below the Insertion Location

The **No Usage** button creates a function template but does not create a usage message template for that function. Therefore, in this case, the function is not exported by the package.

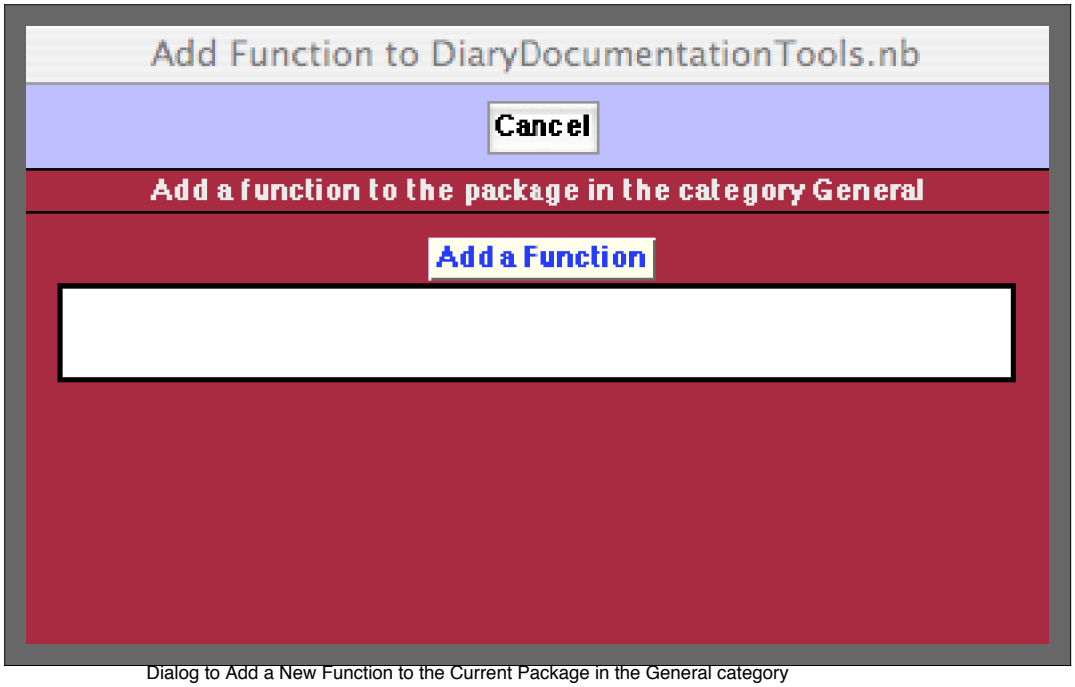
This button toggles open and closed the sub-palette containing the administrative buttons for navigating and creating functions that are in the **General** category of the package.

All packages at least have a General category of functions. The General category is created when the package is first created. Other categories can be added using the **Add Function Category** button above and its resulting dialog.

The **Usage** button moves the insertion point in the current package's notebook to the beginning of the Usage messages section for the General category. The Functions button moves the insertion point in the current package's notebook to the beginning of the **Functions** code section for the General category.

The **Newfn** and **No Usage** buttons here operate in a way that is similar to those described above. However the function and usage templates that are created are placed in the section of the programming notebook where the **General** category items are placed.

The dialog that is opened looks like:



Dialog to Add a New Function to the Current Package in the General category

- GraphicsDisplay

In this category, **GraphicsDisplay** (which, in this illustration, the user has created), the following buttons have the same meaning and use as in the General category above.

- Usages | Functions
- Newfn | No Usage
- PaletteButtonsCells

In this category, **PaletteButtonsCells** (which, in this illustration, the user has created), the following buttons have the same meaning and use as in the General category above.

- Usages | Functions
- Newfn | No Usage

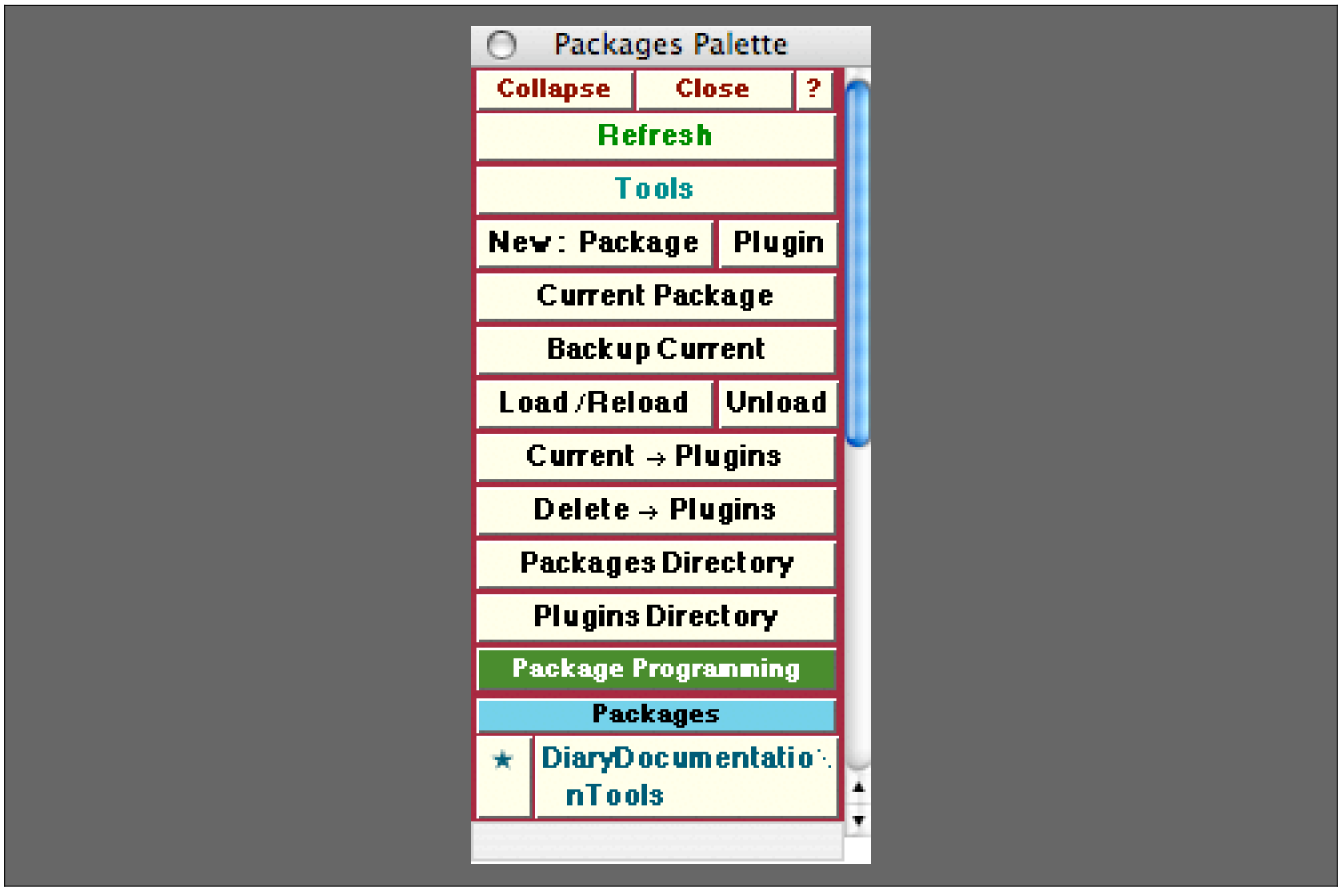
Packages Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Packages** button on the All Palettes Palette, you will open the **Packages** Palette. You can also open the Packages Palette by executing:

```
PackagesPalette[];
```



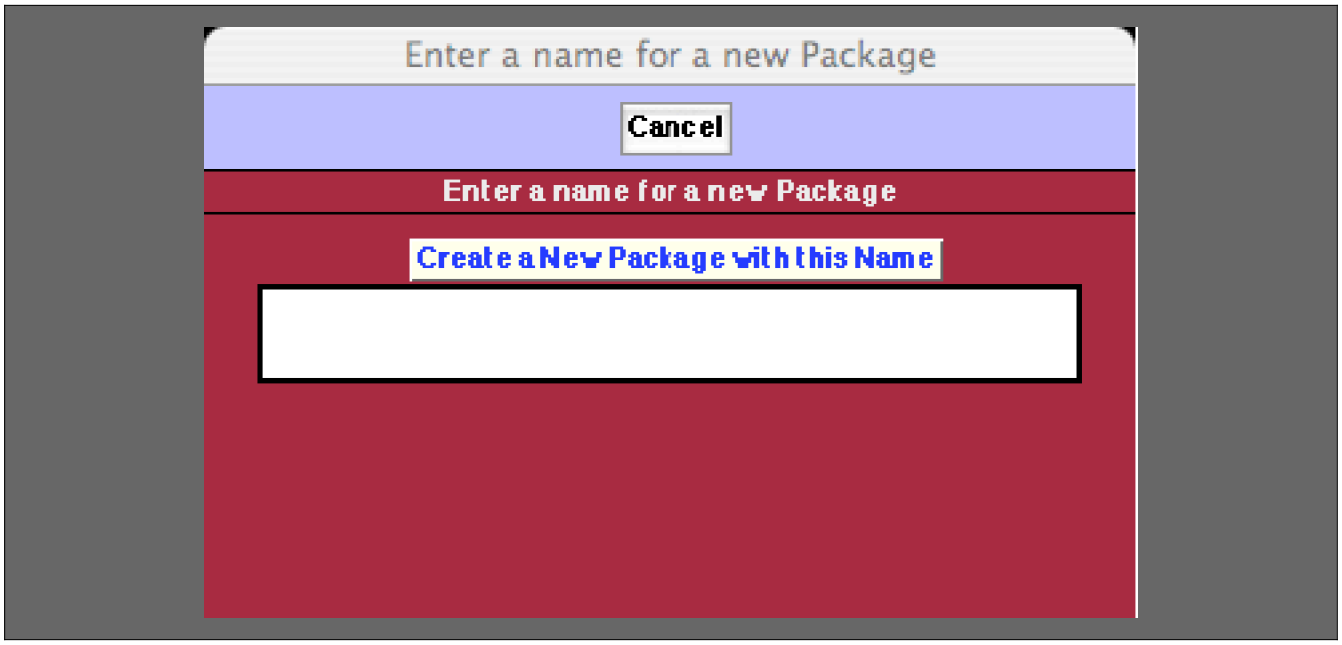
The Packages Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

New: Package **Plugin**

The **New: Package** button opens up a dialog to name and create a new package in the current Diary's Packages Directory.



Dialog to Create a New Package

The new package name must not have any WhiteSpace in it. **Clicking on the button Create A New Package with this Name** opens up a structured notebook in which to do the package programming. This notebook is saved in a directory within the current Diary's Packages directory which has the same name as the package.

The package's name is added to the **Packages** sub-menu at the bottom of this Palette and the path to the package is added to *Mathematica*'s `$Path` parameter so that the package can be found when it is loaded.

Similarly, the **Plugin** button opens up a dialog to name and create a new plugin in the current Diary's Packages Directory. The procedure and details are as just described for the **New: Package** button. A Plugin is a special case of a Package. The chief difference between a Plugin and a Package is that it automatically loads A WorkLife FrameWork™ when the Plugin is loaded (assuming that the system that it is loaded on has a validly licensed WorkLife FrameWork™ installed.

Opens up the **Current Package** (or Plugin) if there is one. The current package programming notebook file is given by `$CurrentPackageNotebookFile`. If the current package programming notebook is open, its notebook object is given by `$CurrentPackageNotebook`.

The **Backup Current** button backs up the current package. In doing so it creates a duplicate of the package's directory with the same name as the original but with a time stamp appended to it.

Load/Reload loads the current Package or unloads and **Reloads** it if it has already been loaded. The **Unload** button simply unloads the package.

Current→Plugins moves the current Package into A WorkLife FrameWork™'s Plugins directory so that it can be accessed by the WorkLife FrameWork™'*s* Plugin functionality and appear on the **Plugins Loading** Palette.

Delete→Plugins removes the current Package from A WorkLife FrameWork™'*s* Plugins directory so that it cannot be accessed by the WorkLife FrameWork™'s Plugin functionality and does not appear on the **Plugins Loading** Palette.

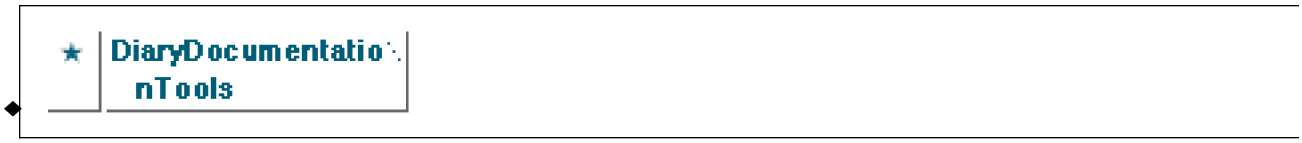
Opens the **Packages** subdirectory of the current Diary's directory.

Opens A WorkLife FrameWork™'s Diary Plugins directory

Opens the **Package Programming** Palette. This Palette can also be opened by executing:

```
PackageProgrammingPalette[];
```

This button toggles open and closed the sub-palette containing the list of **Packages** (and Plugins) that are in the Packages subdirectory of the Current Diary's directory. When one of these buttons with the name of the Package is clicked, the associated Package Notebook is opened and is assigned to be the Current Package. When the ★ button is clicked the associated Package Notebook is assigned to be the Current Package, but it is *not* opened. In the case illustrated in the Palette image above there is one package in the Current Diary's Packages subdirectory:



An example of Package buttons in the **Packages** sub-palette.

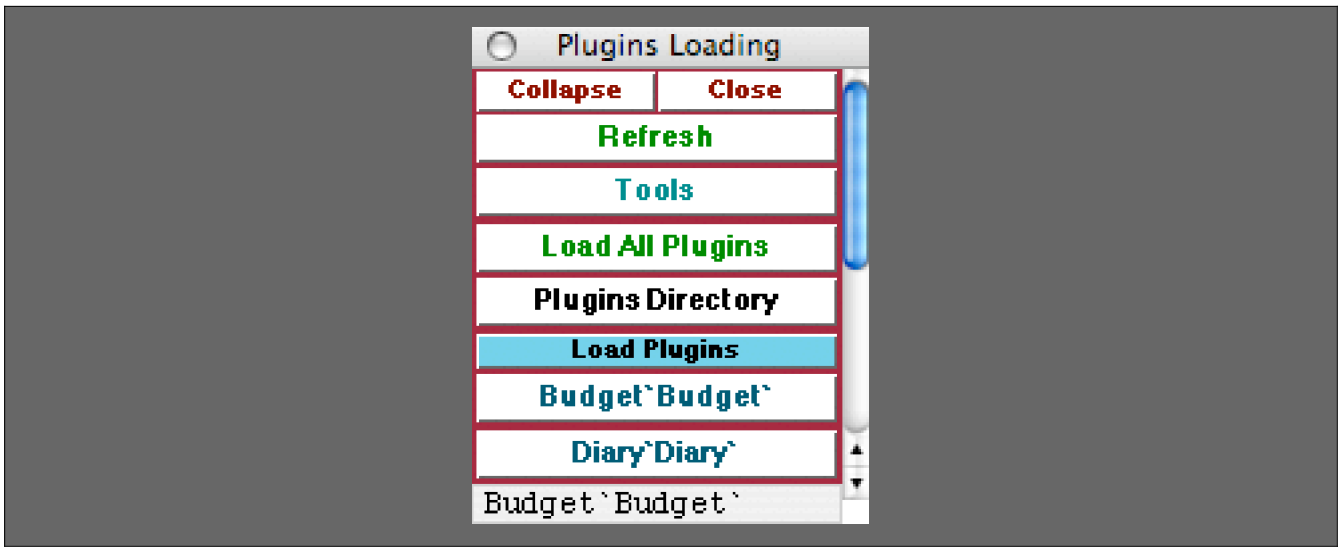
Plugins Loading Palette

The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Plugins Loading** button on the All Palettes Palette, you will open the Plugins Loading Palette. You can also open the Plugins Loading Palette by executing:

```
PluginsLoadingPalette[];
```



The Plugins Loading Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

- **Load All Plugins**

The **Load All Plugins** button loads all of the packages that are in the `$DiaryPluginsDirectory`. Each of these Plugins is listed individually under the **Load Plugins** sub-palette below. Each can be loaded independently by clicking on its button.

- **Plugins Directory**

Opens A WorkLife FrameWork™'s Diary Plugins directory

- **Load Plugins**

This button toggles open and closed the **Load Plugins** sub-palette containing all of the packages that are in the `$DiaryPluginsDirectory`. The packages are listed by their contexts. Clicking on one of these buttons will load the specified package. In this example there are two Plugins in the `$DiaryPluginsDirectory` with the contexts `Budget`Budget`` and `Diary`Diary``.

Budget`Budget`

This button would load the Budget`Budget` package from the $DiaryPluginsDirectory.

Diary`Diary`

This button would load the Diary`Diary` package from the $DiaryPluginsDirectory.

Programming Palette

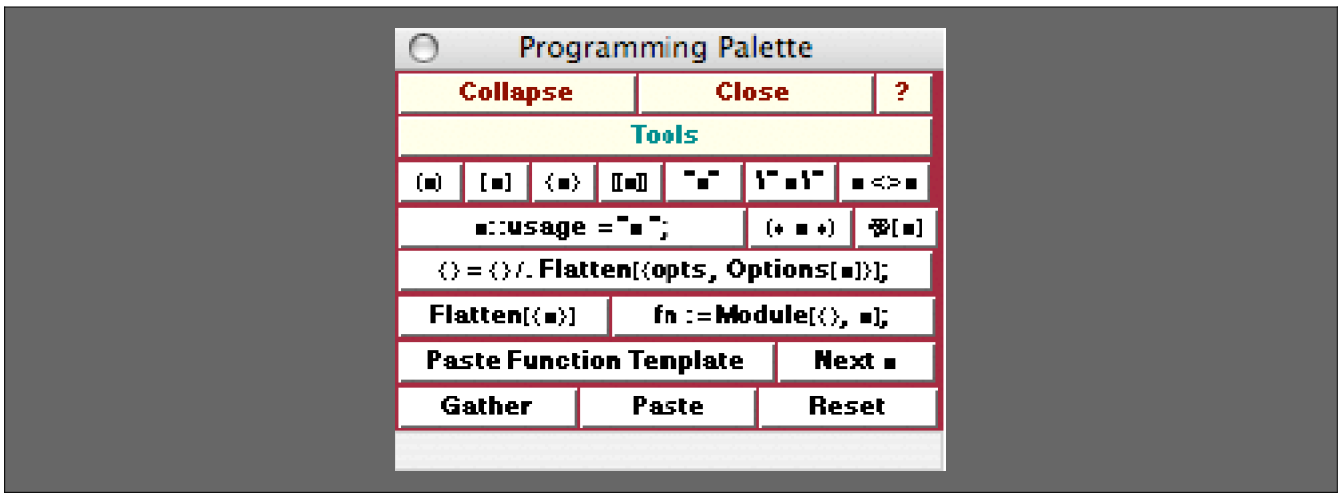
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Programming** button on the All Palettes Palette you will open the Programming Palette. You can also open the Programming Palette by executing:

```
ProgrammingPalette[];
```

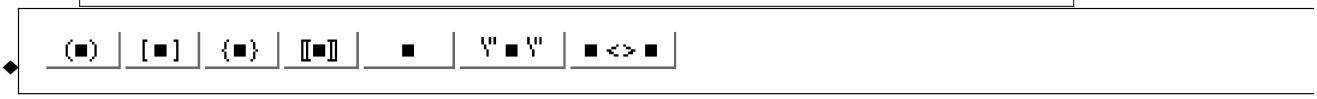
This Palette provides some useful buttons for writing *Mathematica* code.



The Programming Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**



This set of buttons places the selected material in the InputNotebook in the indicated template which mostly consists of bracketing elements.

As examples of each, assume that the text "parameter" is selected in an input cell:

parameter

Then, respectively clicking on one of each of the buttons will would produce,

(parameter)

[parameter]

{parameter}

[[parameter]]

"parameter"

\" parameter \"

parameter <> ■

This pair of buttons places the selected material in the InputNotebook in the indicated template: the first is the template for a usage message and the second one is to create a comment.

As examples of each assume that the text "parameter" is selected in an input cell:

parameter

Then, respectively clicking on one of each of the buttons will would produce,

```
parameter::usage = "■ ";
```

```
(* parameter *)
```

[parameter]

The last one is useful for accessing the data used by a Computation that was recently performed by the **ComputationPalette:**

> ✵["ComputationType"] holds the raw data of data cell of type "ComputationType" after ComputeDiaryNotebook has been executed. If ComputeDiaryNotebook is acting on multiple ComputationTypes then ✵[{"ComputationType1","ComputationType2",...}] holds that raw data.

◆ {} = {} /. Flatten[{opts, Options[■]}];

This button places the selected material in the InputNotebook into a template for passing an option into a function.

◆ Flatten[{■}] | fn := Module[{}, ■];

This pair of buttons places the selected material in the InputNotebook in the indicated template: the first `Flattens` the contents of a `List` and the second one is a template for a function definition using a `Module`.

◆ Paste Function Template | Next ■

The **Paste Function Template** button pastes a function template into the current InputNotebook at the insertion point.

The **Next ■** button moves the selection to the next instance of □ in the current InputNotebook.

◆ Gather | Paste | Reset

This set of three buttons provides an easy way for the user to accumulate a number of distinct selections from a notebook or notebooks into an internal variable.

The **Reset** button removes any previously accumulated items from the internal variable.

Each time the **Gather** button is clicked, the currently selected material in the current InputNotebook is appended to this internal list.

When the **Paste** button is clicked, the material in this internal list is pasted to the current insertion point in the current InputNotebook.

RSS Feeds Palette

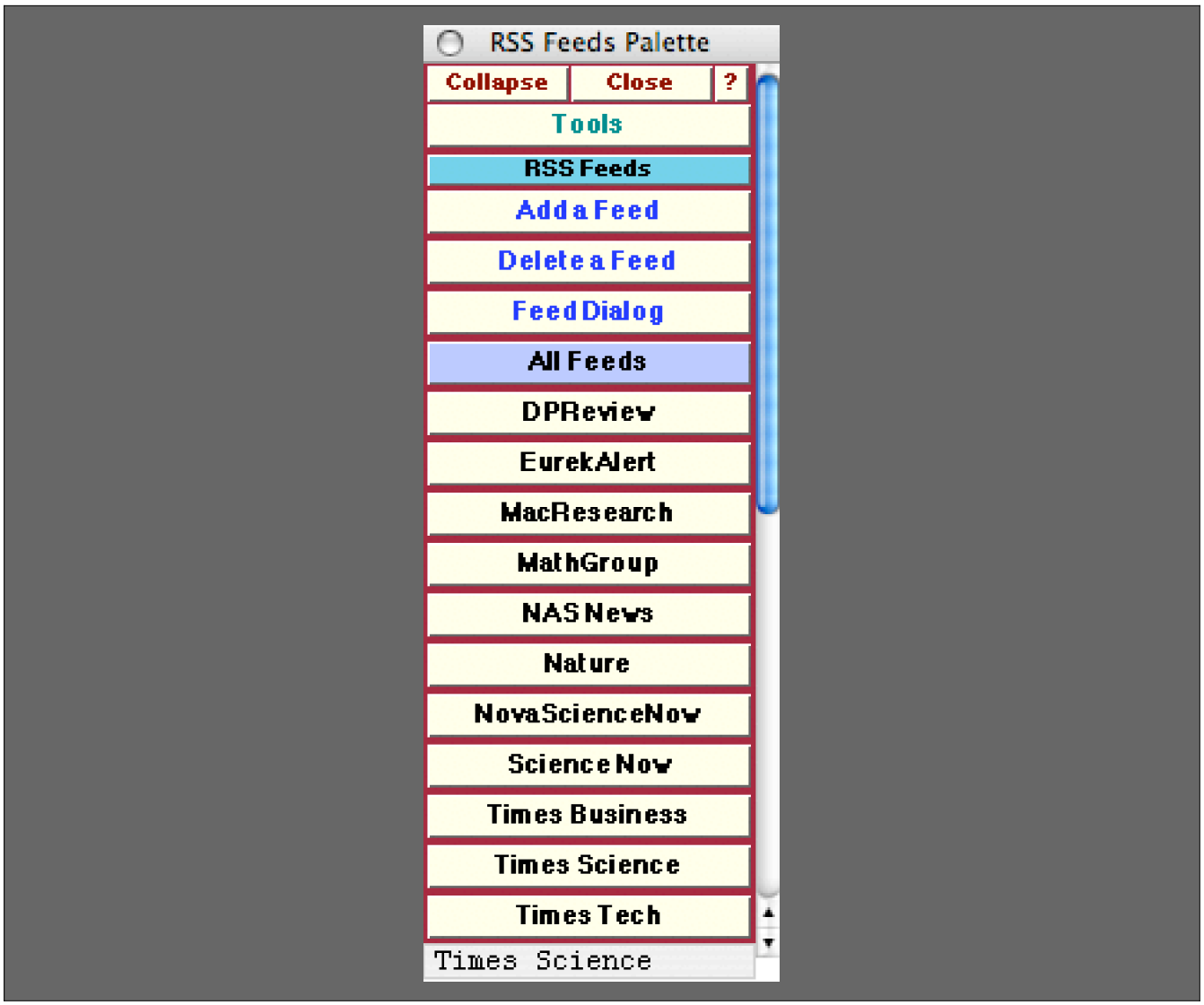
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **RSS Feeds** button on the All Palettes Palette, you will open the RSS Feeds Palette. You can also open the RSS Feeds Palette by executing:

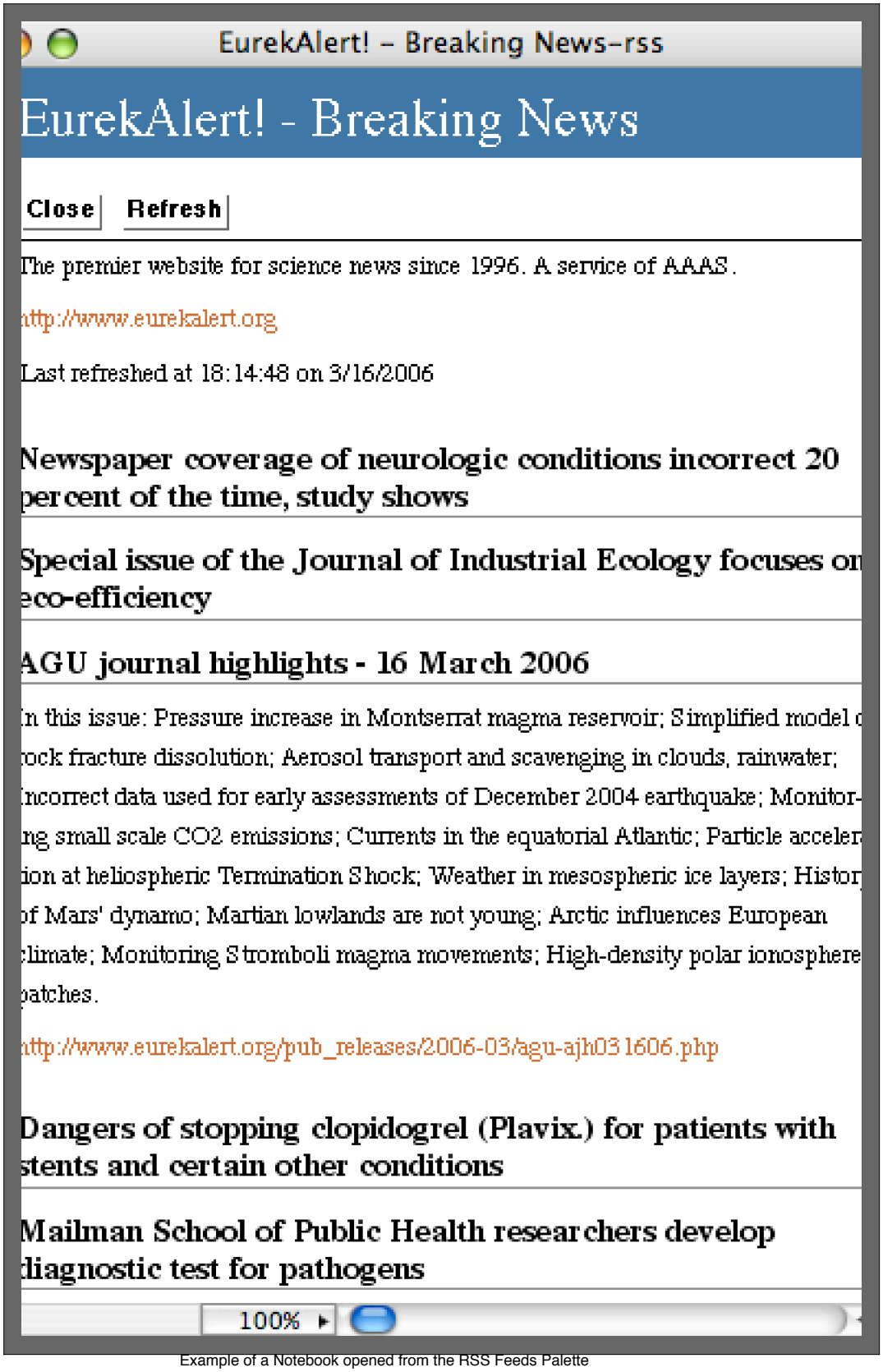
```
RSSFeedsPalette[];
```

This Palette provides an interface to accessing **RSS Feeds** from within *Mathematica* and displays RSS Feeds within *Mathematica* Notebooks.



The RSS Palette. In this Palette the buttons **DPReview** through **Times Tech** are all ones that have been added by the user. Upon opening this Palette for the first time none of these user-added buttons will be present.

When one of the buttons for an RSS Feed is clicked, the material from that RSS Feed is displayed in a *Mathematica* Notebook. For example the **EurekAlert** RSS Feed opened up the following notebook at the time this document was written:



Example of a Notebook opened from the RSS Feeds Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

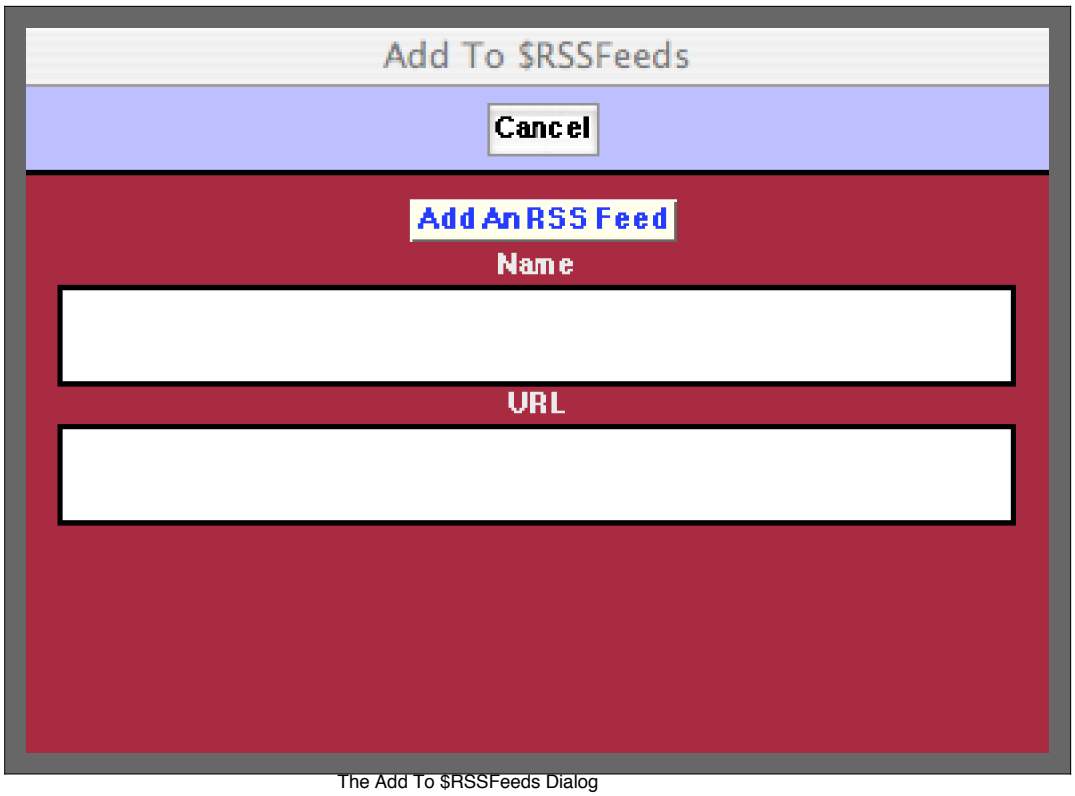
RSS Feeds

Opens and closes the **RSS Feeds** sub-palette which includes the buttons that follow.

Add a Feed

Add a Feed opens a dialog that allows you to add a and RSS Feed.

The dialog looks like:



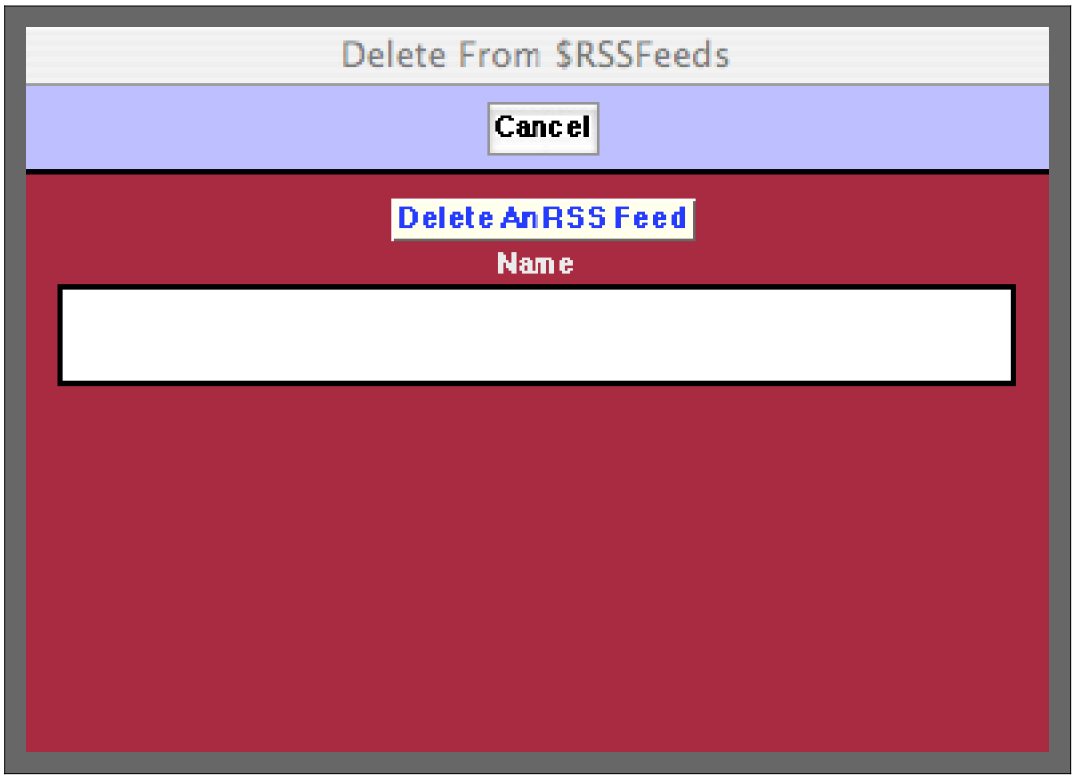
The Add To $RSSFeeds Dialog

In this dialog you give a name to your RSS feed and supply its URL (including the http:// prefix). When you click the **Add An RSS Feed** button the new RSS Feed is added to the Palette and to the `$RSSFeeds` parameter that it is based on.

◆ **Delete a Feed**

Delete a Feed opens a dialog that allows you to remove an and RSS Feed.

The dialog looks like:

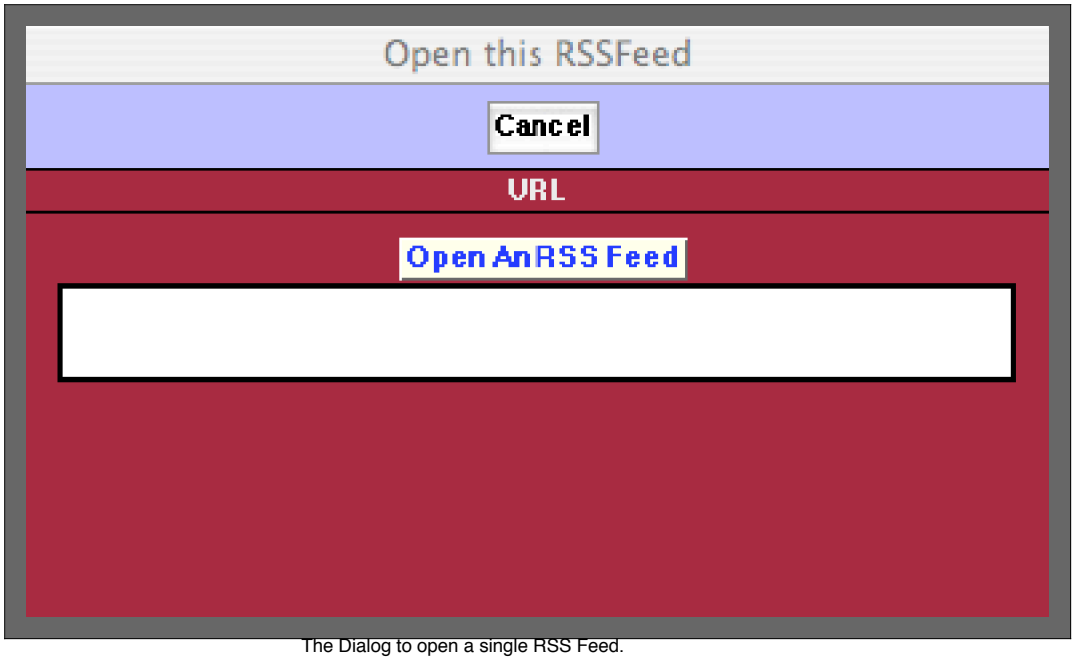


In this dialog you enter the name that you have previously given to your RSS feed (this is the name that appears in that RSS feed's button on the **RSS Feeds** Palette). When you click the **Delete An RSS Feed** button the RSS Feed is removed from the **RSS Feeds** Palette and from the `$RSSFeeds` parameter that it is based on.

◆ **Feed Dialog**

The **Feed Dialog** button opens a dialog that allows you to open an RSS Feed without including it in the RSS Feeds Palette.

This dialog looks like:



The Dialog to open a single RSS Feed.

All Feeds

The All Feeds Button opens all of the RSS Feeds listed on the Palette in a single *Mathematica* Notebook.

DPReview

This is an example of an RSS Feed button that was added to this Palette by the using the **Add a Feed** button to open the **Add To $RSSFeeds** Dialog. On clicking it, it will open the **DPReview** RSS Feed in a new *Mathematica* Notebook.

EurekAlert

This and the remaining buttons below are examples of RSS Feed buttons that were added to this Palette by the using the **Add a Feed** button to open the **Add To $RSSFeeds** Dialog.

MathGroup

NAS News

Nature

Style Sheets Palette

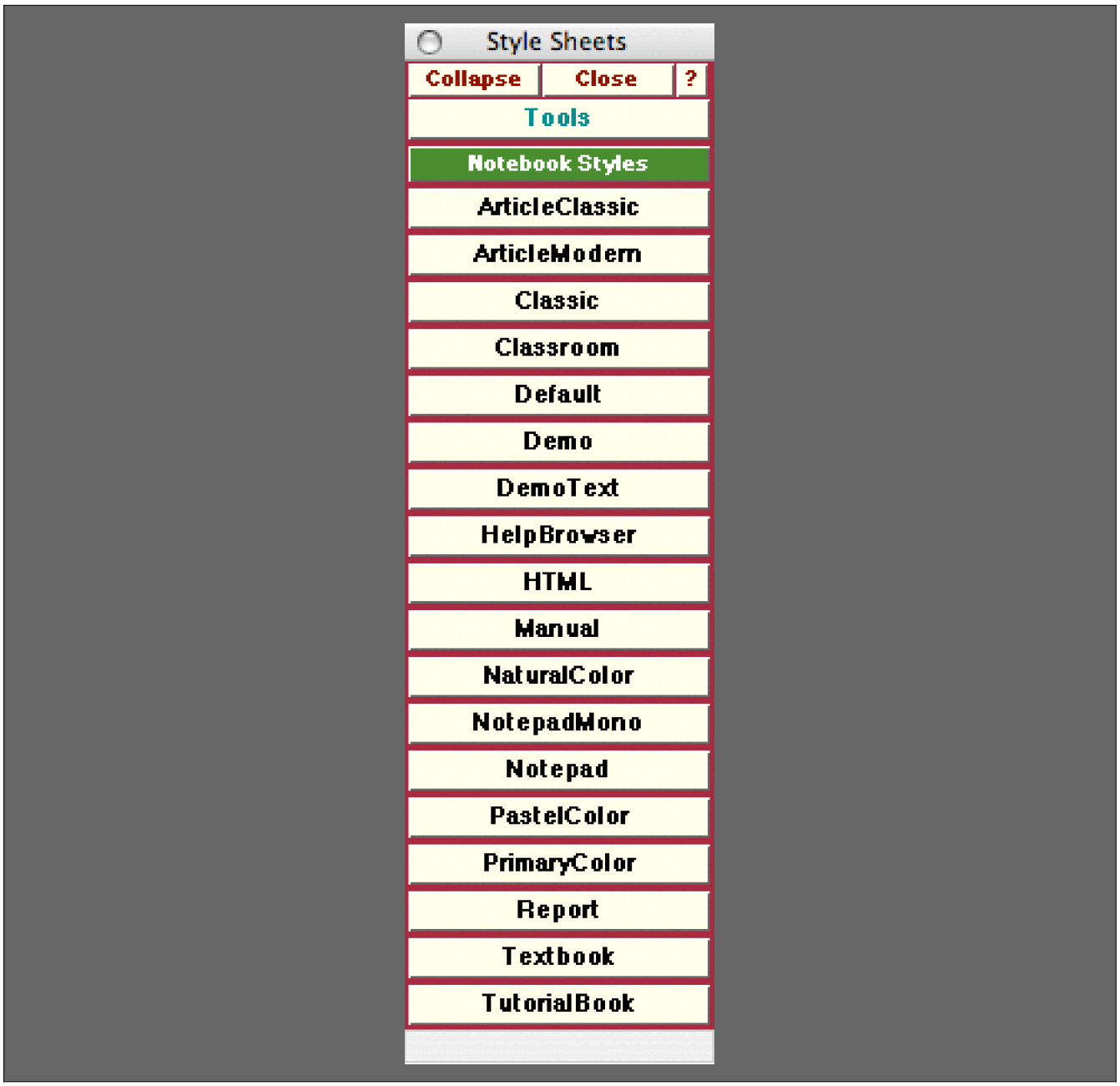
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Style Sheets** button on the All Palettes Palette, you will open the Style Sheets Palette. You can also open the Style Sheets Palette by executing:

```
StyleSheetsPalette[];
```

This Palette lists the style sheets that are contained in the default installation of *Mathematica*. It will also list any other style sheets that you have created or which are present in any *Mathematica* add-ons that you have installed.



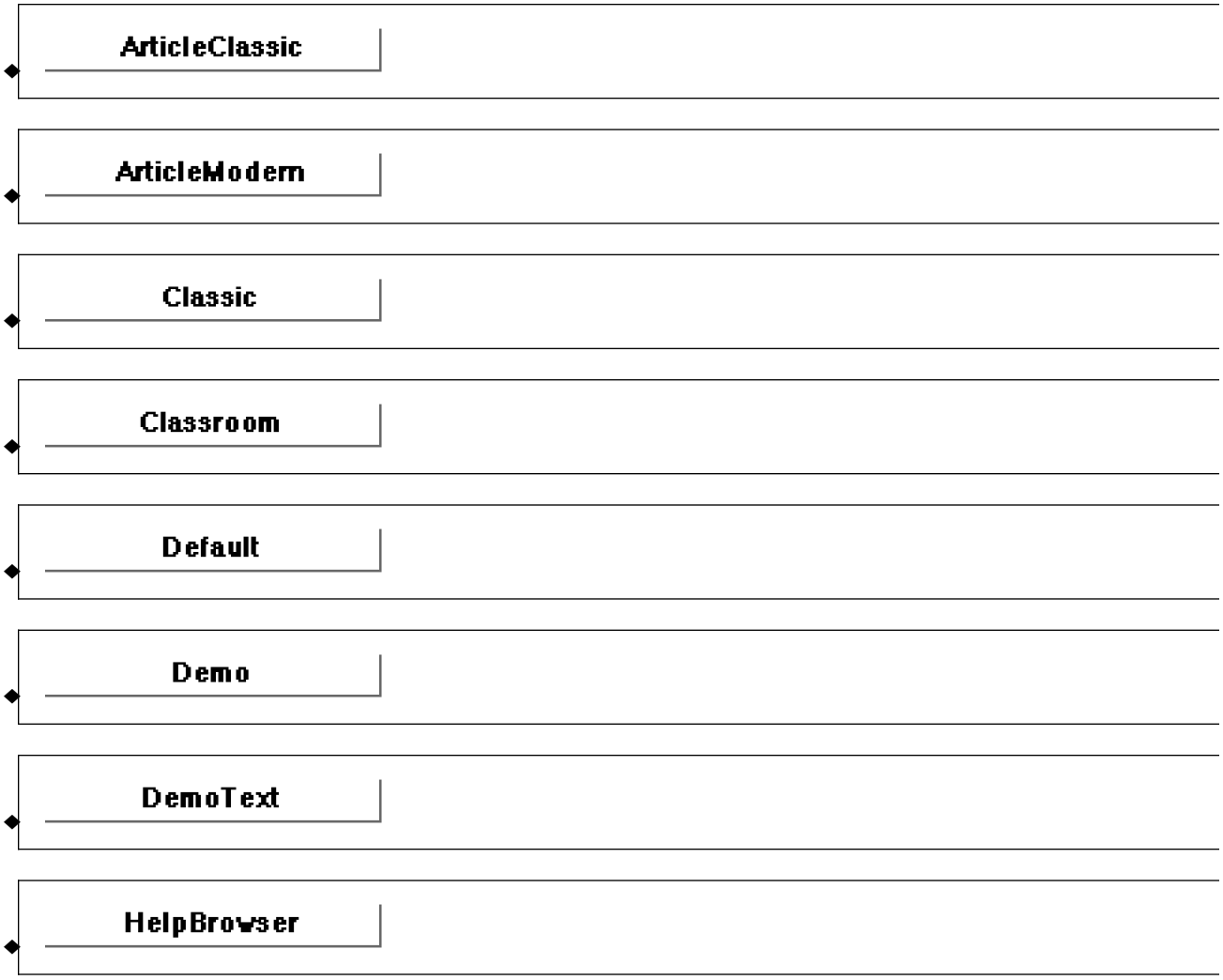
The Style Sheets Palette

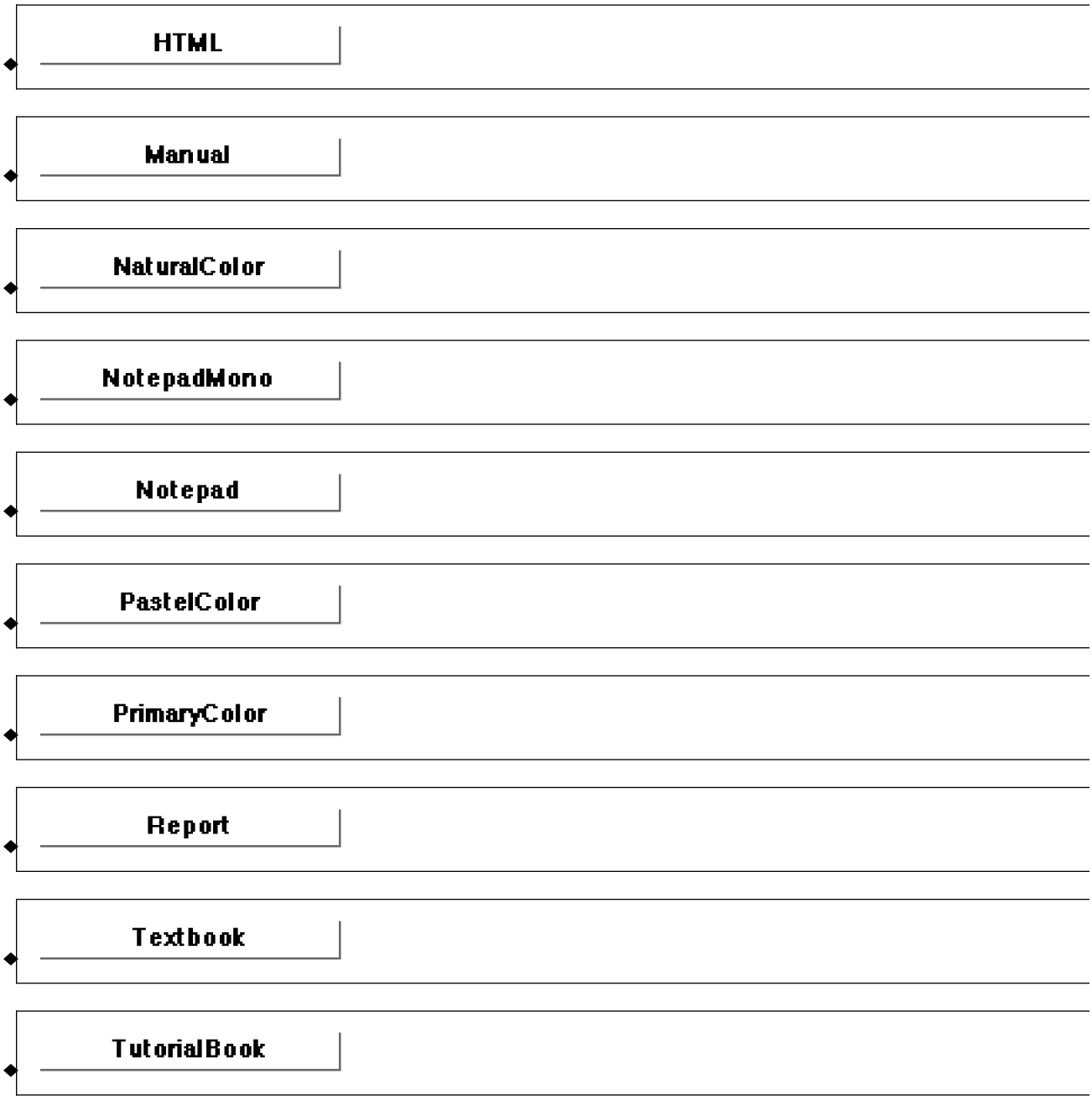
The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

This Palette lists the style sheets that are contained in the default installation of *Mathematica*. Clicking on one of the buttons will change the style sheet of the current input Notebook to the new style sheet.

If the **Styles** Palette is open, then it will be refreshed to reflect the styles that are available in the new style sheet for the current input Notebook.





Tagging Palette

The Palette

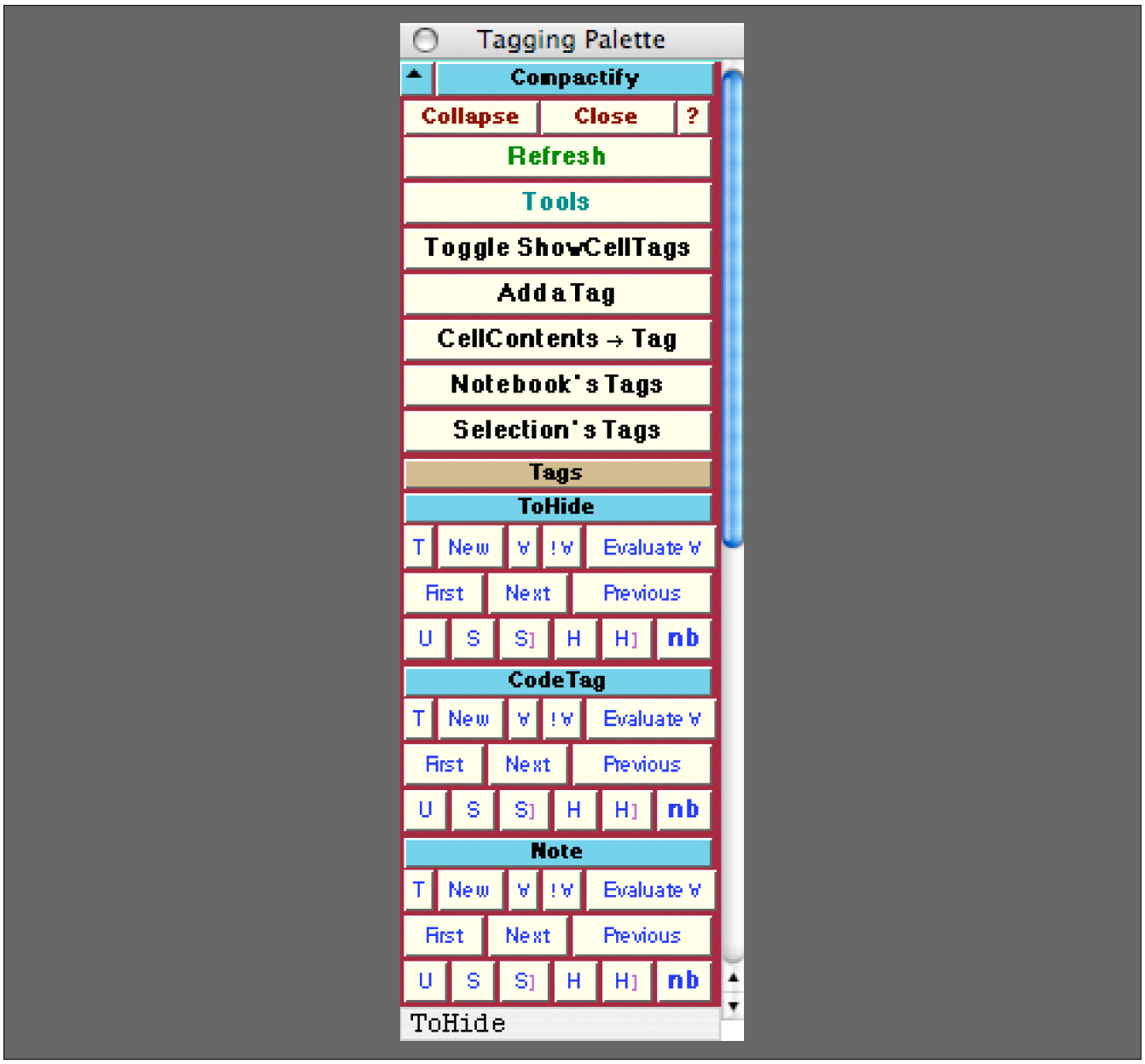
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Tagging** button on the All Palettes Palette, you will open the Tagging Palette. You can also open the Tagging Palette by executing:

```
TaggingPalette[];
```

The **Tagging** Palette provides administrative buttons for tagging cells in *Mathematica* Notebooks and manipulating those notebooks based on the `CellTags` within them.

An example of the **Tagging** Palette looks like:



The Tagging Palette

This image shows an example of this Palette for the case where there are three tags listed: "ToHide", "CodeTag", and "Note". The tags shown in this Palette are governed by the current value of the parameter `$TaggingList`. All three of "ToHide", "CodeTag", and "Note" always appear at the top of the **Tagging** Palette. The "ToHide" tag is intended to be used on cells that you generally wish to toggle from being visible to invisible, and the "CodeTag" is generally intended to be used for those executable cells that you may want to execute independent of any other executable cells in a given Notebook. In this way it can act to create a different set of "Initialization Cells" that you can use to reexecute selected cells in the Notebook. This is for convenience—as you will see in the following discussion of the palette buttons in the Tagging Palette, any tag in the notebook can be used for this purpose, thus allowing for an arbitrary number of effective "Initialization Cells." Finally the "Note" tag is intended for commentar and notes that you might want to be aboe to quickly extract from a notebook..

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

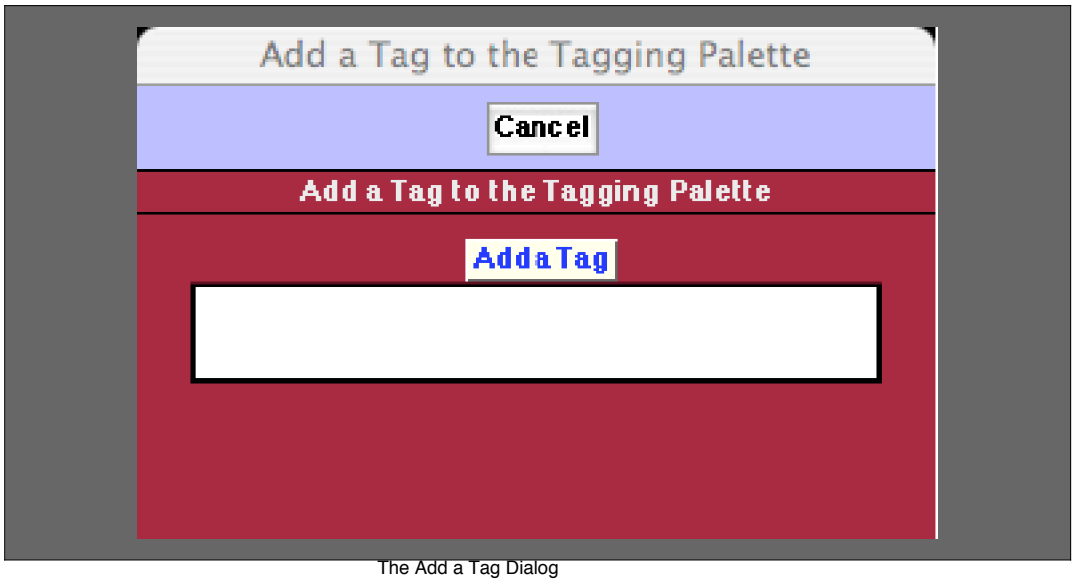
- **Toggle ShowCellTags**

The **Toggle ShowCellTags** button performs the same functions as *Mathematica*'s Find ▹ Show Cell Tags menu item. It toggles between displaying and hiding all of the Notebook's Cell's CellTags.

- **Add a Tag**

The **Add a Tag** button opens a dialog that allows you to add a new tag to the list of tags and their associated buttons included below in this Palette. Through the dialog the **Add a Tag** button adds the new tag to the parameter `$TaggingList`.

This dialog looks like:



The Add a Tag Dialog

- **CellContents → Tag**

The **CellContents→Tag** button takes the contents of the selected Cell in the current InputNotebook and adds it to the list of CellTags for that cell. This, of course, should be used carefully since it can lead to very unwieldy tags.

The **Notebook's Tags** button reads all of the CellTags from the current InputNotebook and changes all of the buttons that follow to reflect that set of tags. The current set of buttons displayed is governed by the list of tags in the parameter `$TaggingList`.

Note that when **Notebook's Tags** *is clicked, all of the buttons listed in this Palette to administer tags will be replaced by those representing the tags in the current InputNotebook. In particular, if you have added a tag to this Palette using the* **Add a Tag** *button above—and if there are no cells with that tag in the current InputNotebook—then that tag will be removed from this Palette.*

The **Selection's Tags** button reads all of the CellTags from the current selection in the current InputNotebook and changes all of the buttons that follow to reflect that set of tags. The current set of buttons displayed is governed by the list of tags in the parameter `$TaggingList`.

Note that when **Selection's Tags** *is clicked, all of the buttons listed in this Palette to administer tags will be replaced by those representing the tags in the current InputNotebook. In particular, if you have added a tag to this Palette using the* **Add a Tag** *button above—and if there are no cells with that tag in the current InputNotebook—then that tag will be removed from this Palette.*

This is a heading button for the sets of buttons that follow.

In the following we illustrate the buttons for each tag with the specific case of the "ToHide" tag.

Opens and closes the **ToHide** sub-palette for the "ToHide" CellTag which includes the buttons that follow.

The set of "ToHide" and "CodeTag" CellTaging buttons always appears as the first set in this Palette. They are included whether or not there are any cells tagged with "ToHide" or "CodeTag" in the current InputNotebook.

This first set of buttons has the following behaviors:

T: Tags the currently selected cells in the current InputNotebook with the "ToHide" CellTag.

New: Creates a new cell below the current selection in the current InputNotebook that is tagged with the "ToHide" CellTag.

V: Selects all of the cells that have been tagged with the "ToHide" CellTag in the current InputNotebook.

!V: Selects all of the cells that have ***not*** been tagged with the "ToHide" CellTag in the current InputNotebook.

Evaluate V: Evaluates all of the cells with the given tag if they are evaluatable cells (such as Input cells).

First: Selects the first cell t in the current InputNotebook hat has the "ToHide" CellTag.

Next: Moves the selection to the next cell that has the "ToHide" CellTag in the current InputNotebook.

Previous: Moves the selection to the previous cell that has the "ToHide" CellTag in the current InputNotebook.

This second set of buttons has the following behaviors:

U: Removes the "ToHide" CellTag from the currently selected cells in the current InputNotebook if any have that CellTag.

S: Shows (Opens) all of the cells in the current InputNotebook with the "ToHide" CellTag if they have been hidden (Closed). If their CellBrackets have been hidden then this button only shows the cells, it does not make their CellBrackets visible.

S]: Shows (Opens) all of the cells in the current InputNotebook with the "ToHide" CellTag if they have been hidden (Closed) and makes their CellBrackets visible if they have been made invisible.

H: Hides (Closes) all of the cells in the current InputNotebook with the "ToHide" CellTag if they are open. If their CellBrackets are visible then this button only hides the cells, it does not make their CellBrackets invisible.

H]: Hides (Closes) all of the cells in the current InputNotebook with the "ToHide" CellTag if they are open. If their CellBrackets are visible then this button also hides their CellBrackets.

nb: Creates and opens a new Notebook with that cells that were tagged.

The buttons for the "CodeTag" and "Note" tags—as well as any others that might appear on this Palette—all serve the same functoin as the ones above.

Web Search Palette

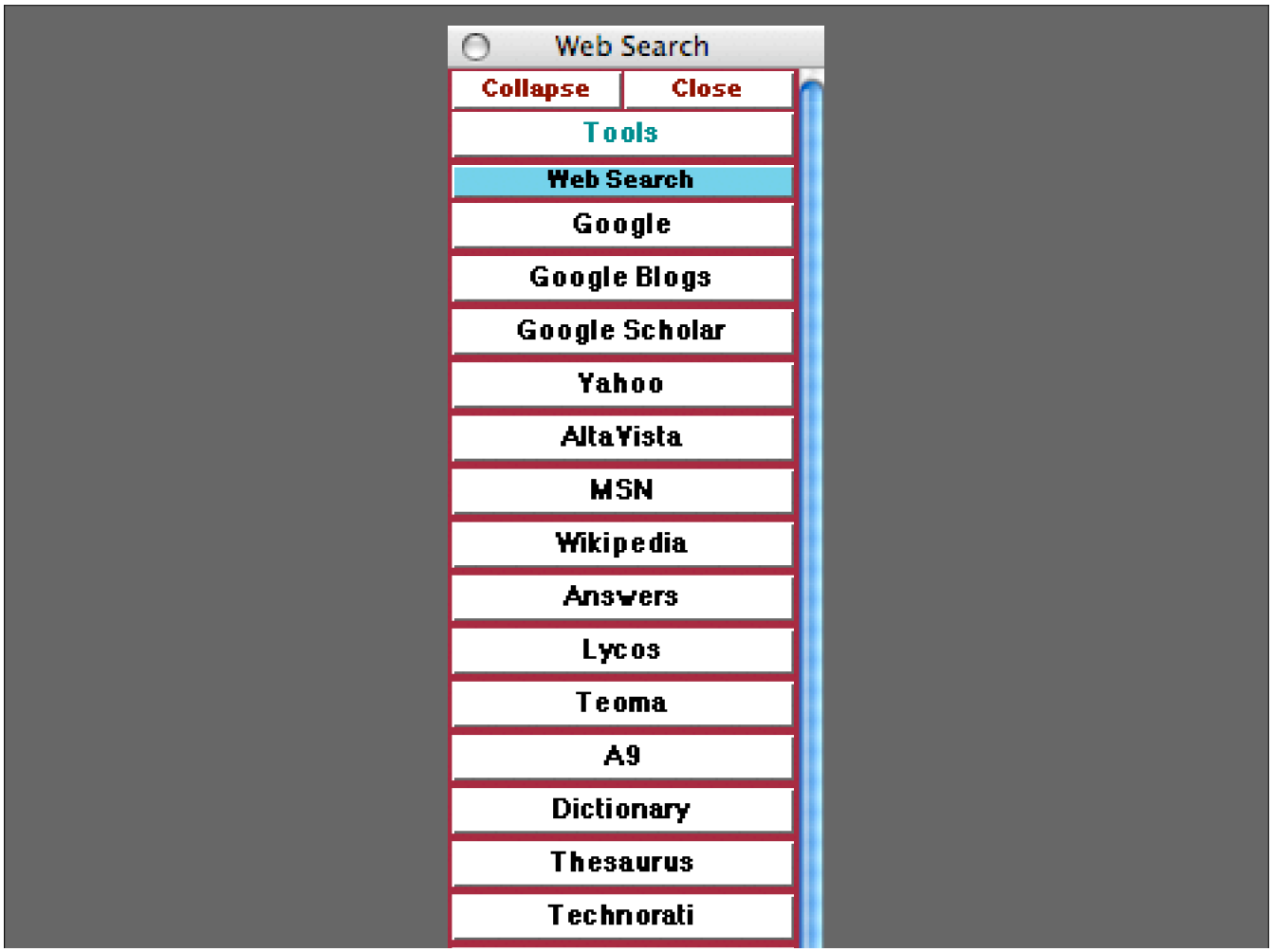
The Palette

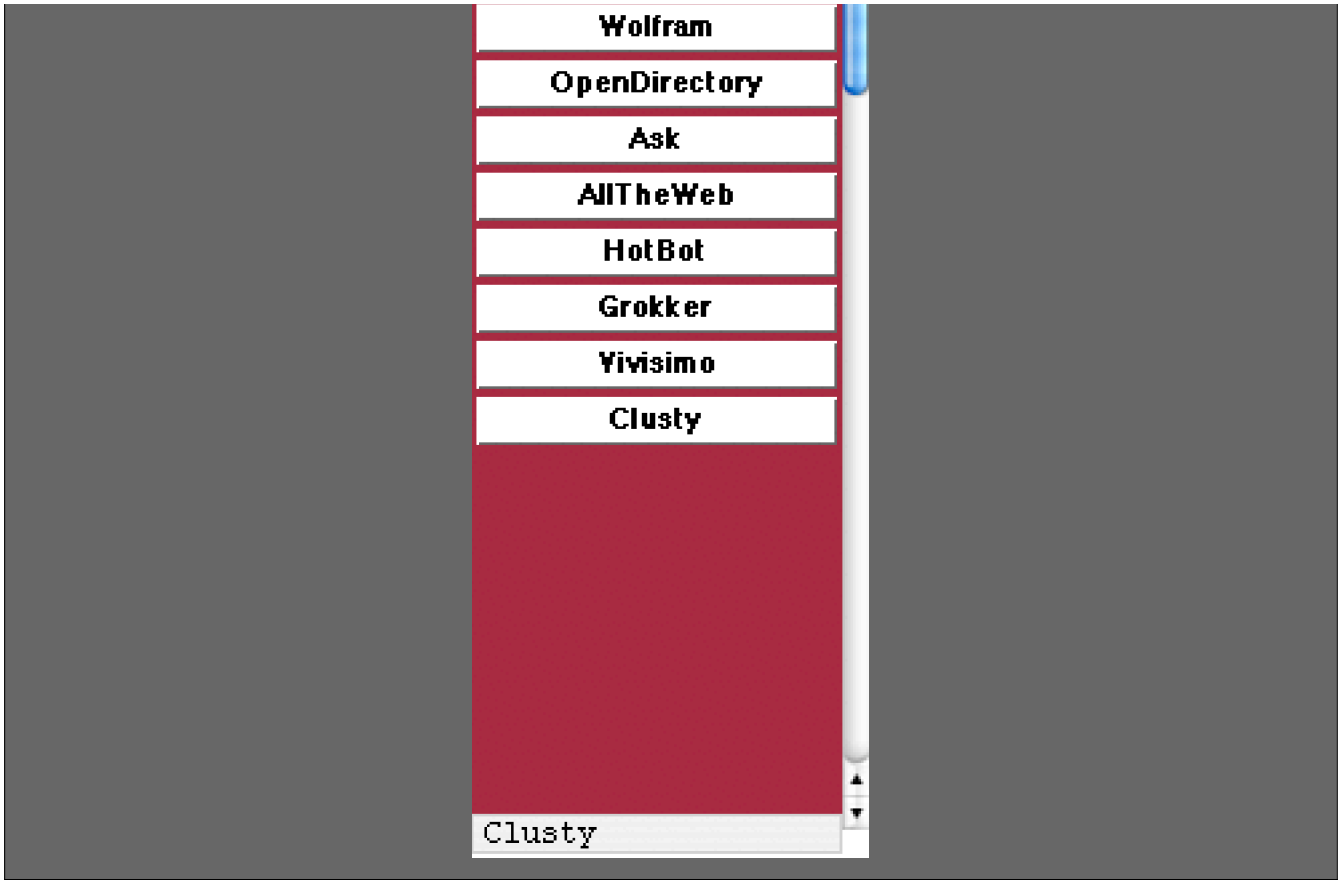
For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **Web Search** button on the All Palettes Palette, you will open the Web Search Palette. You can also open the Web Search Palette by executing:

```
WebSearchPalette[];
```

This Palette provides interfaces to a variety of search engines from within *Mathematica*. In each case the selected text in the current InputNotebook is sent to the chosen search engine and the results are displayed in the computer's default browser.





The Web Search Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command `Needs["Diary`Diary`"]`, or by clicking on the following button: **Load WorkLife FrameWork™**

Web Search

Opens and closes the **Web Search** sub-palette which includes the buttons that follow.

For each button the selected text in the current InputNotebook is sent to the given search engine and the results are displayed in the computer's default browser.

The search engines included in this list of buttons are given by the parameter `$WebSearchEngines`.

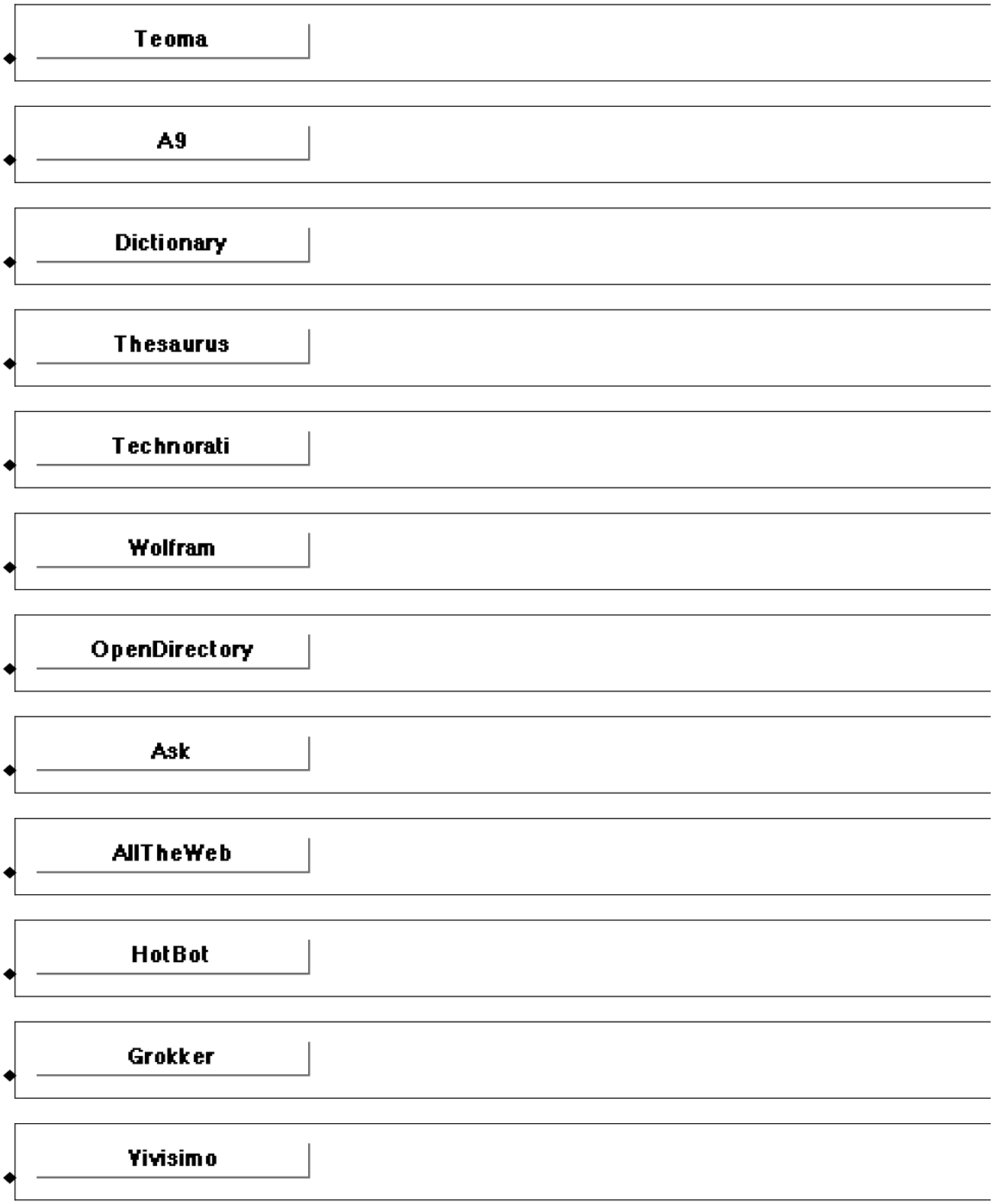
$WebSearchEngines gives a list of search engines that WebSearch knows about. Each element in this list is a list of 4 items.

1: The name of the search Engine
2: A pattern of differing choices for the name
3: A query string (URL) expressed as a pure function
4: A default URL expressed as a pure function

To see examples of these you can evaluate $WebSearchEngines. The correct form for the query string varies from search engine to search engine and generally can be determined by looking at the URL for a search with the particular search engine in a web browser.

Usage message for `$WebSearchEngines`

- Google
- Google Blogs
- Google Scholar
- Yahoo
- AltaVista
- MSN
- Wikipedia
- Answers
- Lycos



Clusty

WorkFlows Palette

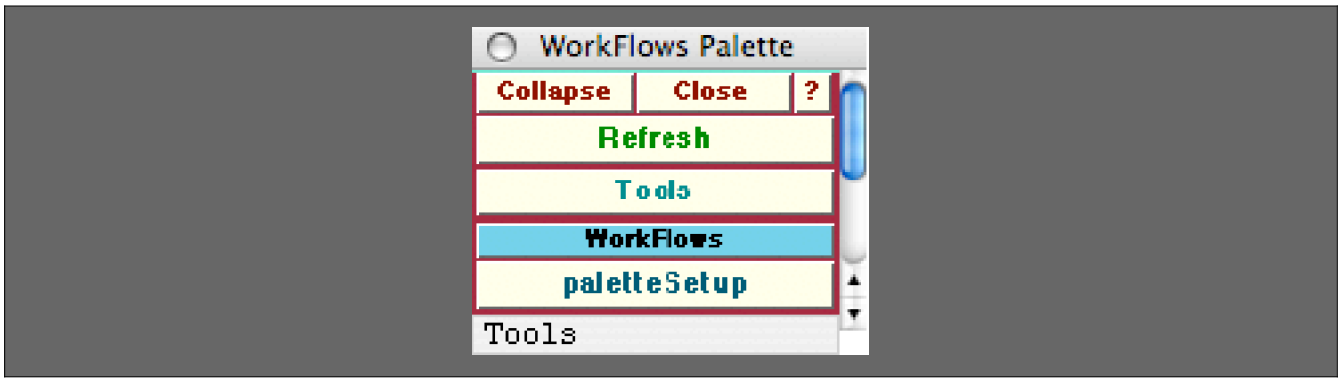
The Palette

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**

By clicking on the on the **WorkFlows** button on the All Palettes Palette, you will open the WorkFlows Palette. You can also open the WorkFlows Palette by executing:

```
WorkFlowsPalette[];
```

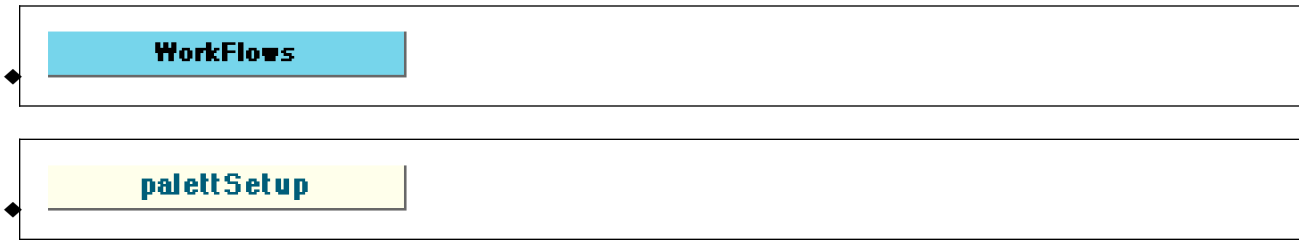
This Palette provides access to those `WorkFlows` that you have created. `WorkFlows` can be created through the function `CreateWorkFlow`.



The WorkFlow Palette

The Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**



Custom Palettes

The Palettes

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**

There are *six* possible Custom Palettes to which users can assign their own buttons or use sets of buttons that others have created and shared.

To open the first Custom Palette execute

```
Custom1Palette[];
```

To open the second Custom Palette execute

```
Custom2Palette[];
```

And so on..., up to the sixth Custom Palette which is opened by executing

```
Custom6Palette[];
```

None of these Palettes contain buttons other than the default heading buttons until the user assigns buttons using the function `AssignButtonsToCustomPalette`. How to do this is covered in the next section.

Adding Custom Palette Buttons

For the buttons and executable commands that are described n this section to work it is assumed that you have installed A WorkLife FrameWork™ and have loaded it. This can be done either from the **Load WorkLife Framework** button on the supplied palette, by executing the command **`Needs["Diary`Diary`"]`**, or by clicking on the following button: **Load WorkLife FrameWork™**

Each of the six possible Custom Palettes can have an arbitrary number of buttons assigned by the user. The function that is

used to assign a set of buttons to one of the Custom Palettes is `AssignButtonsToCustomPalette`.

`AssignButtonsToCustomPalette` takes two arguments: the first argument is an integer from 1 to 6 that specifies which Custom Palette to assign the buttons. The second argument is a list that gives the specifications for the buttons that will appear in the given Custom Palette. When `AssignButtonsToCustomPalette` is executed, the specification for the buttons is saved so that if you restart *Mathematica* and load the WorkLife Framework™ the Custom Palette will open with the specified buttons.

The form of the button specification a list, and each element of that list specifies one button. Each of the button specifications is a list with four parts that looks like

```
{
ButtonText,
ButtonFunction,
ButtonTextStyle,
ButtonButtonOptions
}
```

- ButtonText: This is the text that will appear on the button. It must be a string.
- ButtonFunction: This is a pure function that will be executed by the button
- ButtonTextStyle: This is a list of style directives for the ButtonText. Generally you can simply use the parameter `$CustomPaletteButtonTextStyle` for this.
- ButtonButtonOptions: This is a list of options for the Button. Generally you can simply use the parameter `$CustomPaletteButtonButtonOptions` for this.

`$CustomPaletteButtonTextStyle`	`{`
	`FontFamily→Helvetica,`
	`FontWeight→Bold,`
	`FontSize→10,`
	`FontColor→GrayLevel[0]`
	`}`

`$CustomPaletteButtonButtonOptions`	`{Background→RGBColor[1,1,0.9]}`

Default values for `$CustomPaletteButtonTextStyle` and `$CustomPaletteButtonButtonOptions`

As a simple example here are the button specifications in the second argument to `AssignButtonsToCustomPalette` to assign these buttons to the first Custom palette:

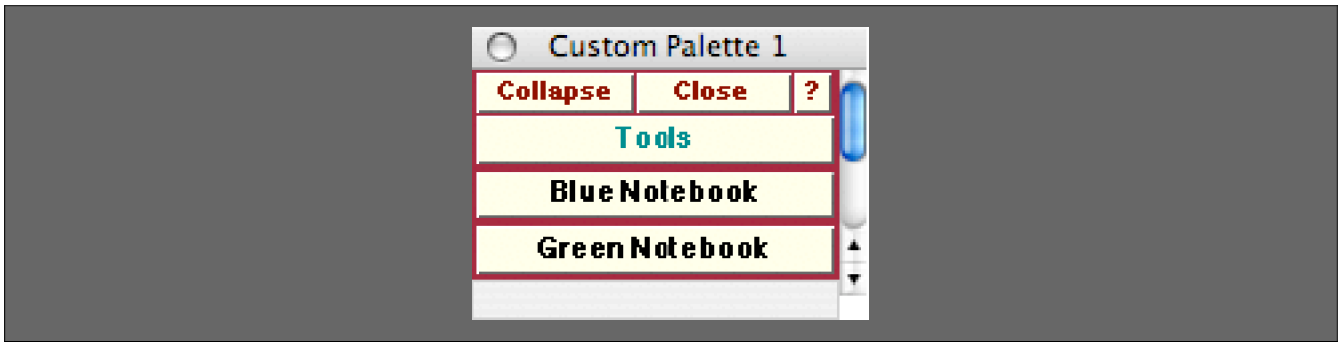
```
AssignButtonsToCustomPalette[1,
 {
  {
   "Blue Notebook",
   (NotebookCreate[Background → RGBColor[0, 0, 1]]) &,
   $CustomPaletteButtonTextStyle,
   $CustomPaletteButtonButtonOptions
  },

  {
   "Green Notebook",
   (NotebookCreate[Background → RGBColor[0, 1, 0]]) &,
   $CustomPaletteButtonTextStyle,
   $CustomPaletteButtonButtonOptions
  }
 }
]
```

The first of these two buttons creates a new notebook with a red background, and the second button creates a new notebook with a Green background.

After executing this and then executing

```
Custom1Palette[]
```

the First Custom Palette looks like



An Example of a Custom Palette

If, in addition to this, you want the colors of the text in the buttons to represent the color of the notebook that would be generated when the button is clicked on, you would modify the button specifications to

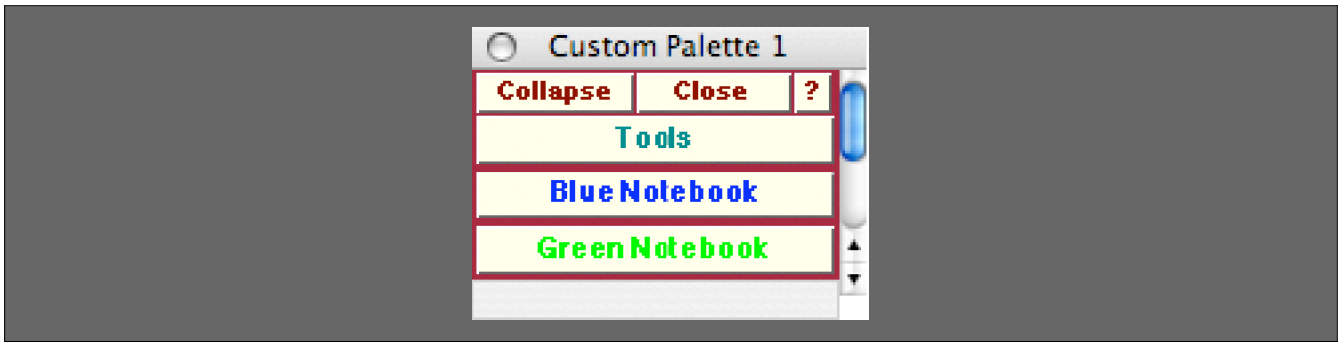
```
AssignButtonsToCustomPalette[1,
 {
  {
   "Blue Notebook",
   NotebookCreate[Background → RGBColor[0, 0, 1]] &,
   {FontFamily → "Helvetica",
    FontWeight → "Bold", FontSize → 10, FontColor → RGBColor[0, 0, 1]},
   $CustomPaletteButtonButtonOptions
  },

  {
   "Green Notebook",
   NotebookCreate[Background → RGBColor[0, 1, 0]] &,
   {FontFamily → "Helvetica",
    FontWeight → "Bold", FontSize → 10, FontColor → RGBColor[0, 1, 0]},
   $CustomPaletteButtonButtonOptions
  }
 }
]
```

Note that these are just simple modifications of the `FontColor` *directives in* `$CustomPaletteButtonTextStyle` *and* `$CustomPaletteButtonButtonOptions`.

After executing the preceding expression and then executing

```
Custom1Palette[]
```

the First Custom Palette now looks like



Another Example of a Custom Palette

The pure function that is used to define the action of the button can either be expressed in the `(...)&` *form or the* `Function[...]` *form. However, if there are any other pure functions that are used* within *the overall pure function, they must be expressed in the* `Function[...]` *form only.*

The function `AssignButtonsToCustomPalette` can actually be used with a simplified version of the argument that specifies the button information. This form simply leaves out the parts of the button information that specify the TextStyle and the ButtonOptions. If these are left out then they will be assigned the current values of `$CustomPaletteButtonTextStyle` and `$CustomPaletteButtonButtonOptions` respectively. So, in this form the first use of `AssignButtonsToCustomPalette` above would read:

```
In[49]:= AssignButtonsToCustomPalette[1,
          {
           {
            "Blue Notebook",
            (NotebookCreate[Background → RGBColor[0, 0, 1]]) &
           },

           {
            "Green Notebook",
            (NotebookCreate[Background → RGBColor[0, 1, 0]]) &
           }
          }
         ]
```

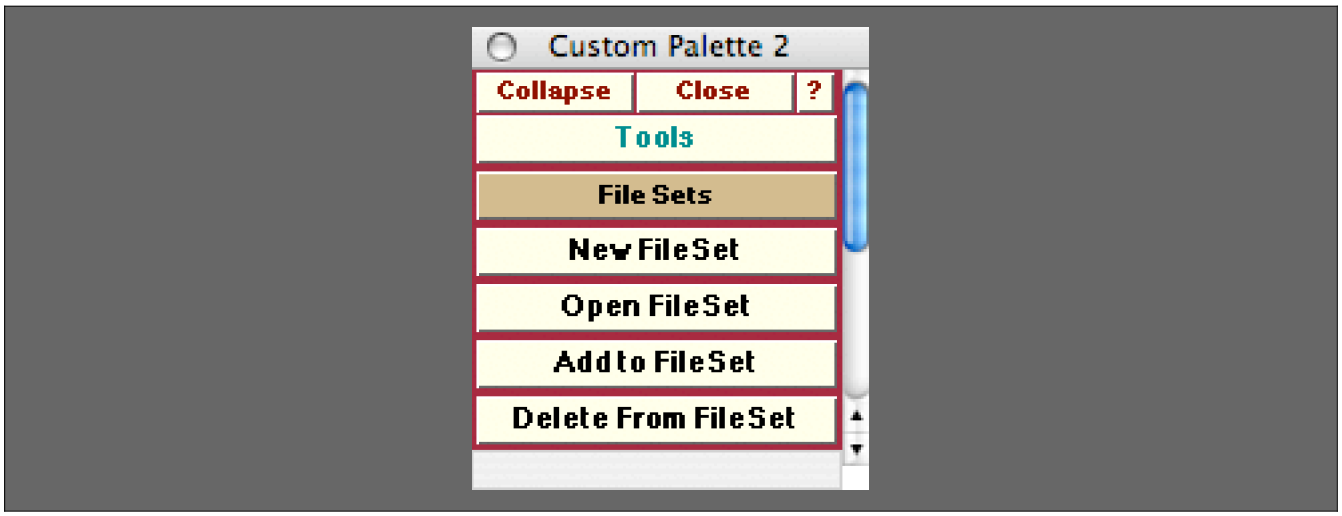
Special Button Sets

There are several special button sets that are provided for you to use in the Custom Palettes. These reproduce the **File Sets**, **Backup & Encryption**, and **Diary Archiving** button sets within the `AdditionalToolsPalette`.

For example, executing

```
AssignButtonsToCustomPalette[2, FileSetsButtonData[]]
```

will assign the File Sets buttons to the 2nd CustomPalette. As a result the `Custom2Palette` would look like:



An Example of a Custom Palette With the File Sets Buttons Assigned to it

Similarly, corresponding to the **Backup & Encryption**, and **Diary Archiving** button sets are the functions `BackupAndEncryptionButtonData[]` and `ArchivingButtonData[]`. And for the Other Tools sub-Palette there is the `OtherToolsButtonData[]` function. Respectively you could then assign these, for example, to custom palettes 3, 4 and 5 as follows,

```
AssignButtonsToCustomPalette[3, BackupAndEncryptionButtonData[]]

AssignButtonsToCustomPalette[4, ArchivingButtonData[]]
```

```
AssignButtonsToCustomPalette[5, OtherToolsButtonData[]]
```