

Dialog Designer 2.7

Welcome to the Dialog Designer. This has been a pet project of mine since January 2007. With it you can create the essential Magik source code for simple or complex GUIs in a matter of minutes, rather than hours. This tool allows you to create and manipulate the key portions of a GUI without writing any Magik or XML. The idea is to develop the GUI quickly to the stage where the GUI activates and the internal behavior can be developed – the really interesting part of GUI creation in my opinion. With a little care you can keep the GUI and background development separate but linked so you can continue to use the Dialog Designer for future GUI manipulations.

The Dialog Designer generates the entire module for a dialog design, complete with XML, message and Magik files. Reverse engineering of existing dialogs so they can be manipulated is **not** supported – I have looked into the problem a bit and it is going to be tricky whenever I get the time to try to code it.

You can save a design you have created to XML and later reload it. These XML files could be used to standardize dialogs, store common GUI element sets or GUI prototype designs and are easily shared between GUI designers/developers.

I have implemented the most common Magik GUI widgets, but there are many less common widgets that a developer may desire. For these, I suggest inserting a common widget into the dialog design for layout purposes and once the Magik GUI code is finalized, update the code with the desired widget. In addition, I have added a few ‘custom’ widgets that address common programming tasks like opening a file, selecting a directory, selecting a style, choosing a date or inserting an action from another plugin.

This application was developed against SW4.2 (SWAF) and SW4.1 (SWAF) images and relies on some classes specific to SWAF images. This release is compatible with SW4.3 (SWAF), though the new GUI classes that are shipping with SW4.3 are not implemented as they are still expected to change in the near term.

Use this tool to create GUIs, generate sample code for cut/pasting into your existing GUI code or just to see the syntax. I hope you find the tool intuitive and useful, as I do.

If you have any suggestions, feel free to send me an email as I am always open to ideas that improve this tool.

Cheers,

Graham Garlick
graham@ifactorconsulting.com

Sept 2012

Table of Contents

Release Notes	4
Reporting Bugs.....	6
Installation and Activation.....	7
FAQ.....	8
What's the fastest way to learn to use the Dialog Designer?	8
How do I save my work?	8
Can I resize the GUI while designing it?	8
The activated GUI on my machine does not look exactly like the rendered design, why?.....	8
How do I change the target path for the generated code?.....	9
How do I make my design activate from my application?	9
How do I preserve my code modifications when rebuilding?	9
The Build_GUI() code is hard to read/edit, why?	9
Why base_class, framework and plugin class code?	10
How do I see parts of my GUI when it is wider than the 'GUI Layout' window?.....	10
Is there an easy way to get an image of the design so I can put it into a document?	11
How do I set the tab text when inserting my design into an existing application tab box?	11
The activated dialog is to right of the DD. Can I activate it elsewhere?.....	11
Internationalization.....	12
Overview.....	13
Main Menubar	15
Main Toolbar.....	16
Status Bar.....	17
Dialog Designer Tabs	17
GUI Layout	18
GUI Layout Toolbox:.....	19
Display Tools.....	20
Show Row Col Grid?.....	20
Refresh	20
Editing Tools	21
Copy	21
Paste	21
Cut.....	21
Select.....	21
Layout Management Elements.....	22
Row Column Grid.....	22
Paned Window	22
Group Box.....	22
Radio Group.....	22
Tab Box.....	22
Window Stack	22
Action/Input Elements	24
Button	24
Label.....	24
Text Input	24
Unit Text Input.....	24
Radio Item	24
Check Box	24
Image Button	25

Image Toggle	25
Slider	25
Number Input	25
Text Choice / Combo Box	25
Row Column Separator (SW 4.2+ required)	25
Custom Action/Input Elements	26
Style Choice	26
File/Director Selector	26
Plugin or Action	26
Date Time Input	27
Outlook Bar	28
Panel Separator	29
Canvas Elements	30
Text Window	30
Simple List	30
Tabular List	30
Recordset GUI Component	30
Tree List	30
Canvas	31
Element Editor	32
Menubar	33
Statusbar	34
Docks	35
Databus	36
Magik / XML	37
Module	38
Messages	40
Embedded messages	40
Sample Dialogs	41
First Dialog	44
Second Dialog	48
Third Dialog	52
Fourth Dialog	55
Fifth Dialog	59
Sixth Dialog	62
Seventh Dialog	66
Testing Dialogs	70
Embedded Dialogs	70
Floating Dialogs	70
Future features under consideration	71

Release Notes

2.7 Sept 7, 2012	<ul style="list-style-type: none">- code tweaks to better manage memory demands for large GUI designs- date_time code changed to better handle current value- fixed bug in outlookbar widget editor so icons load correctly- fixed bug in date time widget
2.6 Oct 15, 2011	<ul style="list-style-type: none">- it_it language added (thank you Francesca Favilli [f.favilli@ictsol.it])- exposed :enabled? property for panel items- exposed :stripes? property for tabular lists- hardened the code generated for Recordset GUI Component- download link updated to point to updated download page
2.5 June 30, 2011	<ul style="list-style-type: none">- increased 'too many elements of type' warning threshold to 10,000- added progress statements when loading a design XML- added :custom_build_gui() hook to generated :build_gui() method- reworked drag/copy/paste mechanism to address a scalability issue- added 'Auto Date Code' toggle for suppressing the dating of generated code- added context action to remove an extraneous layout container- automatically sub-divide the generated init_actions() method to avoid method-too-long parsing error.
2.4 April 15, 2011	<ul style="list-style-type: none">- added more input validation logic for *_spacing attributes- added 'Activation Position' choice to the Module tab- fixed toggle item bug introduced with v2.3
2.3 March 31, 2011	<ul style="list-style-type: none">- add handler for existing condition warning- added text search for base class messages- messages text window now wraps message text- SW4.2 compatibility modifications- fixed combo-box gui-code generation issue- added rowcol_separator widget (SW4.2+, disabled for earlier SW versions)- added 7th example dialog to the Help Document
2.2 Sept 3, 2010	<ul style="list-style-type: none">- sv_se language added (thank you Roger Nyberg [roger.nyberg@ge.com])- added version stamp to the autogenerated public comments.- better validation of base class name- auto-stub the :post_activation() method to the <baseclass>_2.magik file- add support for user-defined extra messages for the *_plugin class- Snapshot recursively generates a PNG for all unique tab/window_stack pages- package statements are now placed in the *_2.magik files- create \source\glue directory that holds code that will be loaded last.- added button to copy foreign icons from referenced modules- cleaner refresh of the GUI Layout- language files updated (en_gb,en_us,zh_cn,fr_ca,es_es,de_de)
2.1 Mar 24, 2010	<ul style="list-style-type: none">- zh_ch language added (thank you Zhaoyong [zhaoyong@gmail.com])- fixed bug with loading unit_text_item from XML- handle missing or bad output path cleanly- render dock actions in the rendering window- updated PDF documentation to reflect dock action rendering- allow the re-initialization of a single dock- a pasted dialog element will now retain (where possible) its unique ID- new 'Snapshot' functionality (saves a PNG of the GUI Layout rendering)

	<ul style="list-style-type: none"> - new 'Download Website' button on 'About Dialog Designer' frame - new toggle to show/hide 'Additional Messages' tab
2.0 Dec 10, 2009	<ul style="list-style-type: none"> - expose balloon_help_text attribute for (text) button_item - handle special XML characters within widget attributes and custom messages to avoid XML syntax errors - expand image selection filtering to use the product name too - fixed additional slot bug - place a sample pragma statement in *_2.magik files for reference - fixed databus XML bug - fixed a scalability issue for very large/complex GUIs. - improve navigation context menu for tab boxes - support dragging of an dialog design within the rendering window
1.9.6 Nov 2, 2009	<ul style="list-style-type: none"> - fixed SW4 compatibility issue (critical, so quick re-release) - added 'Sort Messages' action to toolbar - fixed panel_separator bug
1.9.5 Oct 23, 2009	<ul style="list-style-type: none"> - create additions/changes sub-source directories for code redefinitions - bug fix (statusbar progress indicator element XML generation/parsing) - exposed :allow_filtering? attribute for a tabular_list - fixed statusbar rendering of multiple springy panes - fixed window_pane rendering
1.9 July 15, 2009	<ul style="list-style-type: none"> - added messages tab (define messages to support user-written code) - fix issue where module.def 'requires' section was not updating - various bug fixes, all minor. - added recordset_gui_component widget - added numeric input widget - added 6th example Dialog
1.8 Oct 23, 2008	<ul style="list-style-type: none"> - autosave a copy of current XML dialog description to the generated module. - dt_dt language added (thank you Frank Wolff [frank.wolff@ge.com]) - added button to refresh of icon list, to capture newly loaded modules - expose the double_click_notifier attribute for tabular_list and tree_list
1.7 July 1, 2008	<ul style="list-style-type: none"> - code review, public comments - added Panel Separator widget - added input mechanism for setting which language directory to create - move documentation to PDF format - added 5th example dialog - added link to 'Release Notes' file under menu 'Help'
1.6 May 2, 2008	<ul style="list-style-type: none"> - added Outlookbar widget - support progress_bar pane type in statusbar - support dropdown image button groups within a toolbar - click on rendered menubar or statusbar to jump to its editing tab - added 4th example dialog
1.5 Jan 30, 2008	<ul style="list-style-type: none"> - fixed issue with hanging plugin selection when a CORE raster is missing - fixed issue with non-refresh of Magik/XML code when first going to the Magik/XML tab - added input mechanism to add extra slots to dialog Base Class - added input mechanism for setting pragma statement code settings - added input mechanism for setting the package for the dialog - es_es language edited (thank you Alejandro Cañas [alexcaas@gmail.com]) - added unit_text_item widget

	<ul style="list-style-type: none"> - fixed select_directory (open_file_dir widget) incompatibility with SW4. - fixed internationalization issue with the :top_dock,:bottom_dock (etc.) messages - added date_time_input widget
1.4 Nov 25, 2007	<ul style="list-style-type: none"> - bug fixes (tabular_list, element_editor, tree_list) - modified button_item icons/cursor - fr_ca language added (internet translation ... corrections likely) - es_es language added (internet translation ... corrections likely) - pt_pt language added (internet translation ... corrections likely) - ko_kr language added (thank you Heewook Park [heewook.park@ge.com])
1.3 Oct 15, 2007	<ul style="list-style-type: none"> - added plugin_item widget - removed dependency on select_path module- added file/directory selection widget - added style selection widget - added auto-generation of supplementary gui/framework & plugin files so developers can expand/customize produced GUI functionality in a place that the Dialog Designer won't overwrite if the GUI layout is tweaked and re-built. - added FAQ section - added this "Release Information" section - remove dependency on the 'select_path' module.
1.2.1 Aug 8, 2007	<ul style="list-style-type: none"> - de_de language added (thank you Stefan Alpers [stefan.alpers@its-service.de]) - Fixed scalability limitation so the generated build_gui() is not more than 256 lines long - now multiple methods are recursively generated. - Fixed initialization issue with image_resource setting tool
1.2 July 2, 2007	<ul style="list-style-type: none"> - Initial release version, uploaded to SourceForge July 2, 2007.

Reporting Bugs

Expect a few bugs. The Dialog Designer is fairly mature at this point but has such deep functionality that it is not surprising that you occasionally run into a bug. That said, most confusing 'features' you may run into are easily fixed once I know how to re-create them.

If you find something you would like to tell me about then:

- 1) Try it a couple more times to make sure it really is a problem.
- 2) Document the steps for recreating the problem.
- 3) Send the steps and any supporting data (like an XML file) to me at: graham@ifactorconsulting.com and I will see what I can do.

Installation and Activation

Put the dialog_designer module somewhere logical. I have included a product.def file if you want to keep it separated from other code.

You can start Dialog Designer directly from the Magik prompt once the module is loaded:


```
MagikSF> dialog_designer.open()
```

This will open a new, non-cached instance of the dialog designer.

If you want to launch the Dialog Designer from a button within an existing application then add these lines to your application config.xml and gui.xml respectively.

```
<plugin name="dialog_designer_plugin" class_name="dialog_designer_plugin"/>
```

```
<action name="dialog_designer_plugin.activate_dialog"/>
```

If you place the action in a toolbar the icon will look like this: 

FAQ

What's the fastest way to learn to use the Dialog Designer?

Take the 10 minutes to read the “[Installation and Activation](#)” section and then work your way through the first tutorial as key concepts are demonstrated there. Read the “[Overview](#)” section as you need more details or are just plain confused. There are a further 5 tutorials that present other GUI elements.


How do I save my work?

There are two ways. First, you can save your design to an XML file.

Menupick : File | Save as ...

Once the XML file is specified, you can also quickly save your design version.

Menupick : File | Save

Toolbar : 

Shortcut : Ctrl-s

Can I resize the GUI while designing it?

No. The double chevron that appears just outside the lower right corner of the rendered GUI indicates that when the dialog is activated it **will** be resizable.

Use nested RowCol items to organize the elements in your design and then build/activate the test code to see how it looks. If you really want to change an element's size you can tweak its parameters using the Dialog Designer or by directly modifying the generated code.

Note that the dialog rendered in the Dialog Designer expands/contracts automatically as elements are added, edited, moved or deleted so there is no need for the user to do it.

See also the FAQ: [How do I see parts of my GUI when it is wider than the 'GUI Layout' window?](#)

The activated GUI on my machine does not look exactly like the rendered design, why?

Your desktop settings will greatly affect the final appearance of your GUI. That's why there is a quick way to launch the dialog design, so you can check how it will appear.

How do I change the target path for the generated code?

Code is generated when you press the “Generate” or the “Generate/Activate” buttons. The output directory path defaults to ‘c:\temp’ initially and you can change this “Path” value in the “Module” tab.

How do I make my design activate from my application?


Read the [“Testing Dialogs”](#) section that appears near the end of this document.

How do I preserve my code modifications when rebuilding?

Alongside the dialog, framework and plugin magik code files three more files with the same filenames + “_2.magik” are generated. These are the files where you should put any code for redefining methods, defining shared constants, etc. The Dialog Designer will NOT overwrite these file if they exist, so you can tweak the GUI and rebuild the module without losing your modifications.

Even if you want to do more advanced GUI development than the Dialog Designer currently supports you will find that by placing a ‘post_activation()’ method in the <BaseClass>_2.magik file you can quickly add or modify GUI elements as you see fit.

If you are really stuck and can’t see a way to use this technique you’ll have to manage the modifications you make to the autogenerated code. Once you have made your modifications

you can use the ‘Launch’  button to reload the module code and launch a test dialog. Again, I don’t advise this last method and up till now I have never had to resort to it.

The Build_GUI() code is hard to read/edit, why?

The form of the generated code, especially the build_gui() method, is driven by the need to auto-generate the code. The algorithm was developed to produce code that works for simple or complex GUIs, not for coding beauty.

As a rule, I NEVER modify the auto-generated files. I ONLY write code into the *_2.magik code files where it will not be modified when the code to support any GUI modifications is regenerated by the DD. Coding in this manner, you should be able to create most, if not all, of the GUIs you develop.

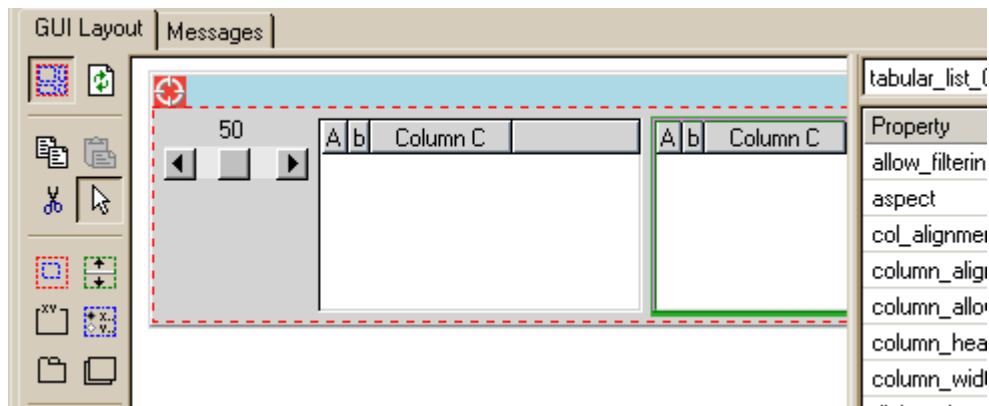
Use a ‘post_activation()’ method to initialize your GUI when it first activates – you can set whatever lists, pulldowns, context menus, labels , ... etc without touching the auto-generated code. If you want to set more complex things (like an editor on a column of a tabular list) you can still do it after the GUI initializes. In addition there is a hook to the ‘custom_build_gui()’ method that is called from the very bottom of the auto-generated ‘build_gui()’ method ... redefine the ‘custom_build_gui()’ method if you need to manipulate the GUI before it is returned to the SW code interpreter (this moderately rare).

Why base_class, framework and plugin class code?

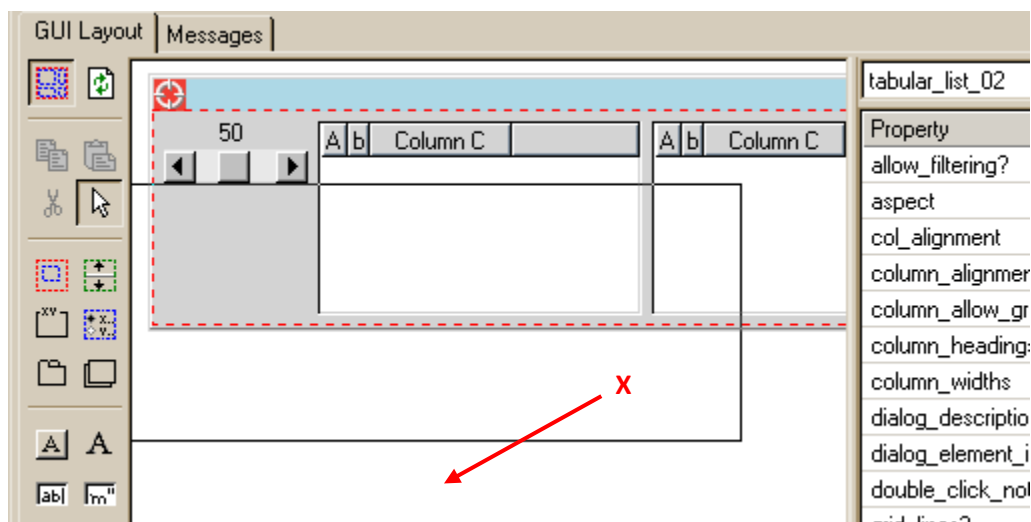
The DD generates code that works whether it is activated as a floating dialog or embedded into another framework. You need all the code for a floating dialog. You only need the 'Base Class' when embedding into an existing framework. See "[Testing Dialogs](#)" for the syntax. The vast majority of the code is written on the base_class, so leaving the framework and plugin class code as auto-generated will not appreciably bloat your image.

How do I see parts of my GUI when it is wider than the 'GUI Layout' window?

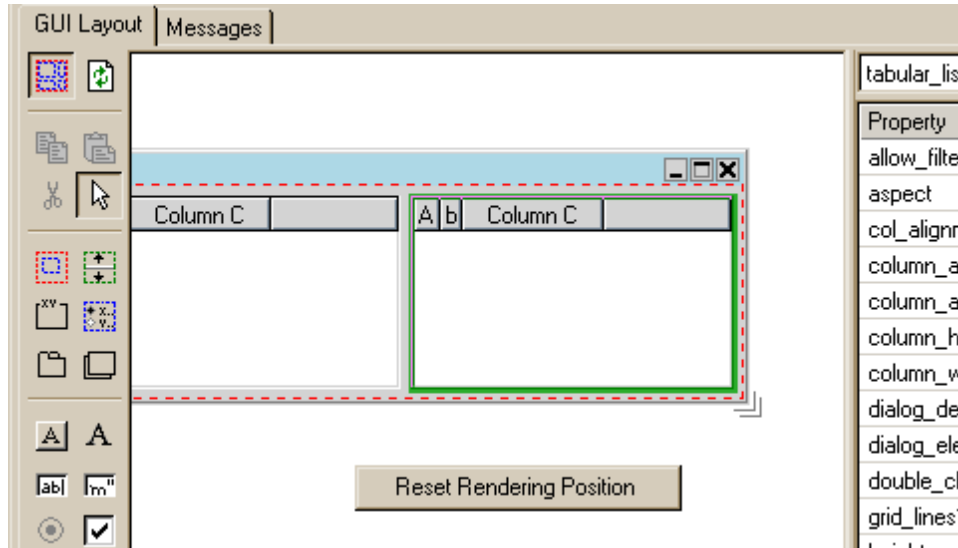
You can drag the entire rendering in any direction by left-clicking **OUTSIDE** the rendered area and dragging in any direction. For example, in this design a part of the design is obscured by the attribute editor.



By clicking (X) and dragging (→) the rendering position outline will be shown like this:




Releasing the drag will refresh the dialog in its new position:



To return the rendering to its default position, right-click **outside** the rendered area and select the 'Reset Rendering Position' action.

Is there an easy way to get an image of the design so I can put it into a document?

Yes, use the 'Snapshot' tool  that appears in the main toolbar. This saves a copy of the rendered design to one or more PNG file(s). Since more complex dialog designs can have embedded tab boxes and window stacks multiple PNG files will be generated so each tab/window is shown in at least one of the saved files.

To see where the PNG file will be saved, hover over the 'Snapshot' button. Once the dialog code has been generated at least once the tool will place the PNG file(s) next to the module.def file in the generated directory structure.

How do I set the tab text when inserting my design into an existing application tab box?

The tab text label will use the title value you set for the frame of your dialog.

The activated dialog is to right of the DD. Can I activate it elsewhere?

Yes, use the 'Activate Position' choice (Module Tab) to select where to activate the test dialog. The position is still relative to the DD and the options are Right (default), Below, In Front and Cascade.

Internationalization

The current languages supported with message files are:

- en_gb English (base development language)
- en_us English in US (base extension language)

- zh_ch Chinese contributor: Zhaoyong [zhaoyong@gmail.com]
- dt_dt Dutch contributor: Frank Wolff [frank.wolff@ge.com]
- de_de German contributor : Stefan Alpers [stefan.alpers@its-service.de]
- fr_ca French editor?
- it_it Italian contributor: Francesca Favilli [f.favilli@ictsol.it]
- ko_kr Korean contributor : Heewook Park [heewook.park@ge.com]
- pt_pt Portuguese editor?
- es_es Spanish editor : Alejandro Cañas [alexcaas@gmail.com]
- sv_se Swedish contributor: Roger Nyberg [roger.nyberg@ge.com]

The base language is 'en_gb', for which all messages are defined. This language should be one of the !current_languages! in your image. The base extension language is 'en_us', all messages that I would consider for translation when adding a new language are included in the 'en_us' message files. A contributor translates the language files for a particular language and periodically updates those message files as needed. An editor completes/corrects the rough translation I have done for a particular language and then takes over the periodic update task.

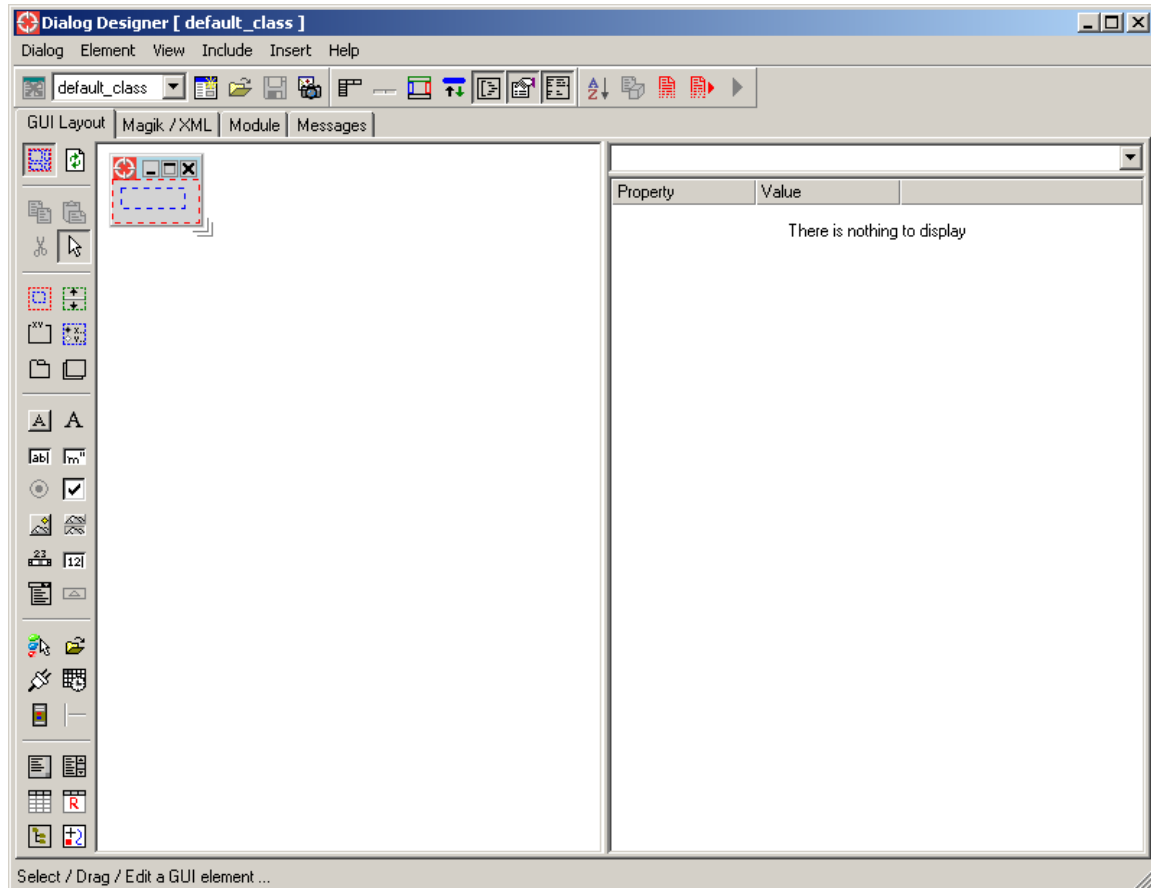
Here is a list of commonly used languages. If you are interested in contributing, just translate all the messages defined for 'en_us' into one of these languages and I will be happy to include it in the next distributed version.

- | | |
|-------------|-------------|
| • Afrikaans | • Hungarian |
| • Arabic | • Japanese |
| • Bulgarian | • Polish |
| • Czech | • Romanian |
| • Danish | • Russian |
| • Finnish | • Slovak |
| • Greek | • Turkish |
| • Hebrew | • Urdu |
| • Hindi | |

If you know of any other language that is used by a SW client, I will gladly add it to the language wish list.

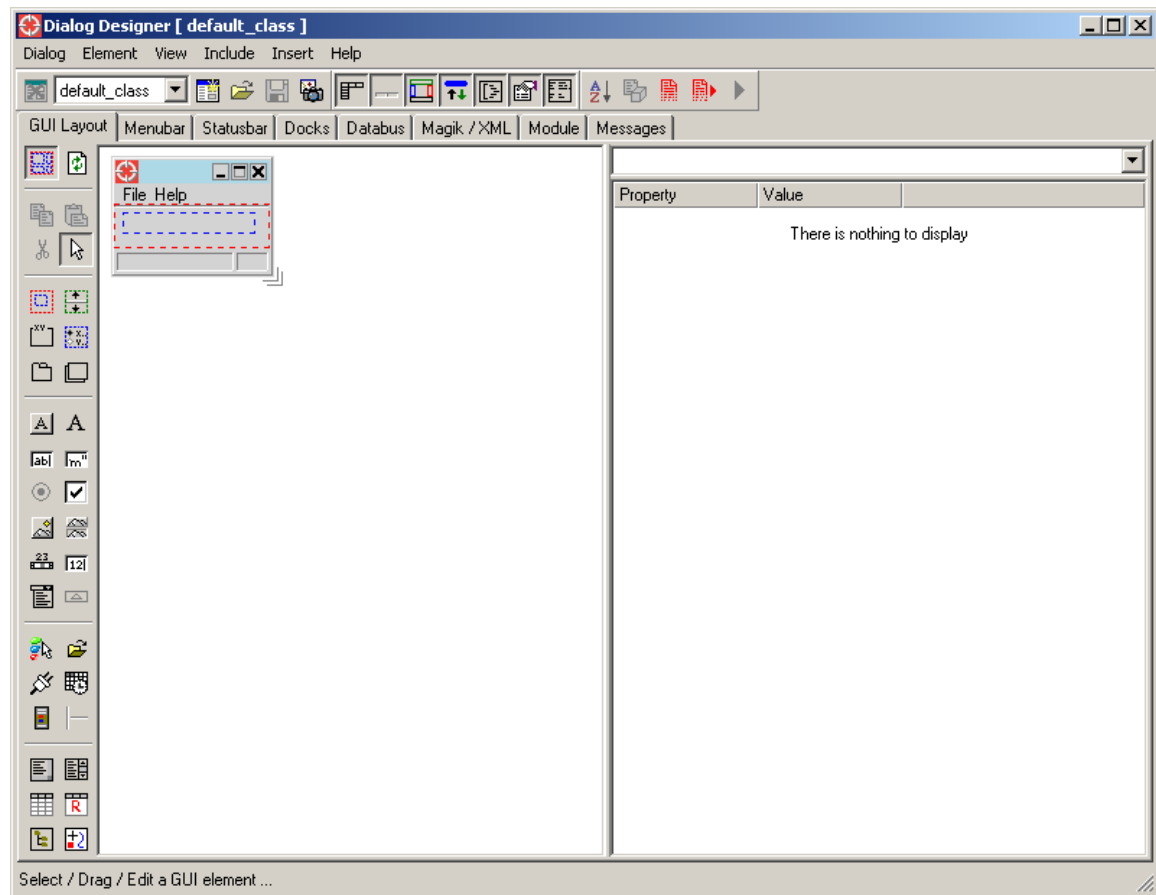
Overview

Here's the Dialog Designer as it will first appear. There are the usual GUI elements: drop down menus, toolbar, tab box and a status bar.



An empty design with minimal elements is current and a rendering is shown in the graphic portion of the 'GUI Layout' tab. The rendering is not resizable ... all resizing will occur automatically as you place elements into the GUI or move them from place to place. The chevron visible just outside the lower right corner of the rendered dialog indicates the dialog **will** be resizable when activated.

Nothing is shown in the editor to the right as no element has been selected yet. There are also a number of hidden tabs that become available when a design contains certain elements. Here is the tool with all the other tabs (Menubar, Statusbar, Docks, & Databus) visible:



Note that the design rendering shows the default menubar and statusbar now that those tabs are visible.

Main Menubar

Below is a listing of all the actions found in the main menubar. Some of these actions also appear as buttons/toggles elsewhere in the Dialog Designer:

<ul style="list-style-type: none">• Dialog<ul style="list-style-type: none">○ New○ Open○ Save○ Save as ...○ Close Dialog Design○ Exit• Element<ul style="list-style-type: none">○ Copy (Ctrl+C)○ Cut (Ctrl+X)○ Paste (Ctrl+V)○ Delete (Ctrl+D)• View<ul style="list-style-type: none">○ Refresh (F5)○ Show RowCol Grid? (Toggle)○ Show/Hide Magik/XML tab○ Show/Hide Module Properties tab• Include<ul style="list-style-type: none">○ Show/Hide Menubar tab○ Show/Hide Statusbar tab○ Show/Hide Docks tab○ Show/Hide Databus tab	<ul style="list-style-type: none">• Insert<ul style="list-style-type: none">○ Canvas○ Check Box○ Date Time○ File or Directory Selection○ Group Box○ Image Button○ Image Toggle○ Label○ Number Input○ Outlook Bar○ Plugin or Action○ Paned Window○ Panel Separator○ Radio Button○ Radio Group○ Recordset GUI Component○ Row Column Grid○ Row Column Separator○ Simple List○ Style Choice○ Tab Box○ Tabular List○ Text Button○ Text Choice○ Text Input○ Text Window○ Tree List○ Unit Text Input○ Window Stack• Help<ul style="list-style-type: none">○ Documentation○ Release Notes○ About Dialog Designer
--	---

Main Toolbar



- Button : Close Dialog Design
- Choice : Choose Dialog from list of current designs
- Button : New Dialog
- Button : Load from XML
- Button : Save to XML ('Save as ...' action is in "Dialog" drop menu)
- Button : Snapshot (save a PNG of the design rendering, hover over for path)



- Toggle : Show/Hide Menubar Tab
- Toggle : Show/Hide Statusbar Tab
- Toggle : Show/Hide Docks Tab
- Toggle : Show/Hide Databus Tab
- Toggle : Show/Hide Magik/XML Tab
- Toggle : Show/Hide Module Properties Tab
- Toggle : Show/Hide Additional Messages Tab



- Button : Alphabetize the user defined messages in the Messages tab
- Button: Copy image files from source into the module's \bitmaps directory
- Button : Generate dialog module, magik/xml and msg files
- Button : Generate dialog module, magik/xml and msg files and launch
- Button : Reload the code and launch the GUI (useful when tweaking the code)

Status Bar

The Dialog Designer Status Bar shows contextual info and user prompts.

To add/remove a statusbar to/from the current dialog design toggle the Statusbar toggle. The rendered GUI Layout will display a statusbar whenever the Statusbar toggle is depressed.

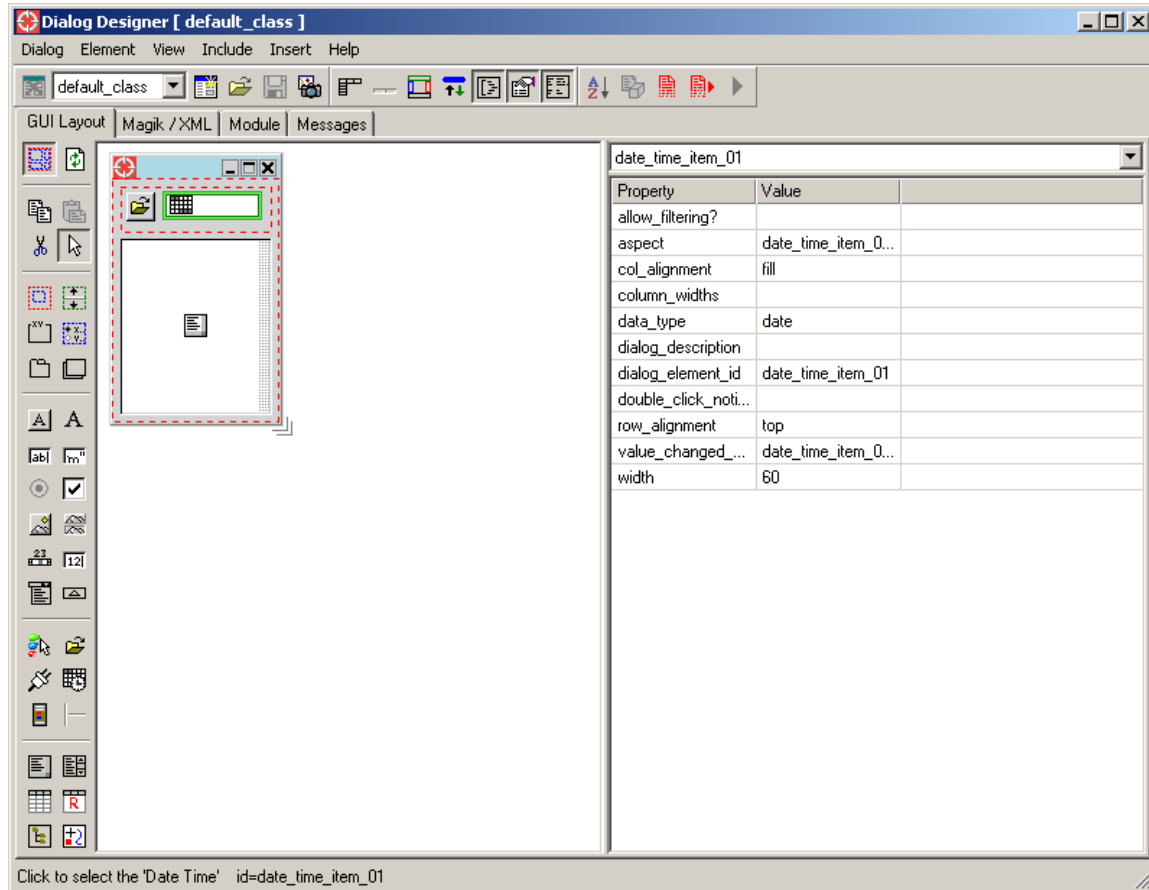
Clicking on a rendered status bar within the GUI Layout tab will immediately bring the Statusbar tab to the front for inspection and editing.

Dialog Designer Tabs

The tabs act as separate but linked dialogs for entry and display of dialog specifications and derived output. Except for the 'GUI Layout' tab, they can be hidden by toggling their associated button in the main toolbar. When the menubar, statusbar, docks or database tab is visible the magik/xml for that part of the dialog design will be generated. Hiding one of the tabs does not delete the element; it just excludes it from the generated code.





































The tabs are listed below, along with general features.

GUI Layout



- Switch current input widget by clicking a mode toggle
- Click-to-drop widgets
- Drag widgets within design (select mode)
- Drag entire design within the window if the design is too large to see all at once
- Toggle visibility of rowcol outline (dashed red) & empty rowcol cells (dashed blue)
- Auto alignment via rowcols
- Radio button insertion control (only allowed into a Radio Group)
- Tab Boxes/ Window Stacks/ Paned Windows / Group Boxes for cleaner, more usable GUIs – use them!
- Edit key parameter values shown in the attribute editor
- Control Frame button availability
- Menubar / Statusbar / Dock visuals if included in design
- Copy / Cut / Paste / Delete elements (includes enclosed elements) within a design or between designs
- Right click context menu for currently selected widget in the design, including:
 - Copy/Cut/Delete a widget
 - Remove an extraneous rowcol object
 - Remove all sub-elements of a container object
 - Add/remove/navigate amongst Tabs within a tab_box
 - Add/remove windows to a window stack

GUI Layout Toolbox:

Show Row Col Grid?			Refresh
Copy			*Paste
Cut			*Select
Row Column Grid*			*Paned Window
Group Box*			*Radio Group
Tab Box*			*Window Stack
Button*			*Label
Text Input*			*Unit Text Input
Radio Button*			*Check Box
Image Button*			*Image Toggle
Slider*			*Number Input
Text Choice/Combo Box*			*Row Colum Separator (SW 4.2+)
Style Choice*			*File/Directory Selector
Plugin or Action*			*Date Time Input
Outlook Bar*			*Panel Separator
Text Window*			*Simple List
Tabular List*			*Recordset GUI Component
Tree List*			*Canvas

An * indicates a 'mode toggle'. Only one mode can be active at a time. There are basically two mode types: 'Select' mode and all the other 'Insert' modes. Pressing the 'Spacebar' on your key board will move you to 'Select' Mode.

When in 'Select' mode (as the graphic above indicates) you can select any GUI element to view/edit its attributes in the visible property editor. Once selected any element (except the outer-most 'Row Column Grid' and the 'Frame' titlebar) can be dragged to another valid location.

All the other modes allow you to insert a default instance of various GUI element types. The cursor, as it appears within the dialog rendering area, will reflect the current mode. All inserted objects have two common fields:

- | | |
|--------------------|--|
| dialog_element_id | This is the unique (within a design) key to the GUI element. You can edit the key of any element and modifying the field in the element editor. All keys are a single word, lowercase and start with a letter. |
| dialog_description | This descriptive text is optional. If set this is the value that will appear in the text_choice_item above the element_editor that lists all the visible elements of a design. This text, if available, will be inserted into generated code as a private comment. |

Display Tools

Refresh the design rendering and toggle the display of the layout guide lines.

Show Row Col Grid?

Depress this toggle to show guidelines on the design rendering. Useful while constructing a design, de-toggle to see a clear view of your design.

Refresh

This button refreshes the design rendering.

Editing Tools

Copy, cut and paste a single element or a group of GUI elements. Select an individual element to view/edit its attribution.

Copy

Press this button (or use the shortcut keystroke Ctrl-C) to copy the currently selected design element along with any elements it may contain. It is common to 'Copy' and then immediately 'Paste' so the user is switched to that mode automatically after the 'Copy' is complete. You can copy from one design to another.

Paste

Depress this toggle (or use the shortcut keystroke Ctrl-V) to enter 'Paste' mode. Hover the cursor over the design and any valid 'Drop Area' will be highlighted with a RED rectangle. Click the left mouse button to drop a copy of the current clipboard element into the design.

To see what the current clipboard element is just hover over the Paste button, the tooltip will show the element class and (if any) the number of enclosed elements.

Use the 'Copy' or 'Cut' buttons to change the clipboard element.

Cut

Press this button (or use the shortcut keystroke Ctrl-X) to cut the currently selected design element along with any elements it may contain. The element is removed from the dialog design but a complete copy is saved into the design clipboard from where it can be pasted. It is common to 'Cut' and then immediately 'Paste' so the user is switched to that mode automatically after the 'Cut' is complete. You can cut elements from one design and paste them into another.

Select

Depress this toggle (or tap the SPACEBAR) to enter 'Select' mode. In this mode you can select any element. The current selection will be highlighted with a GREEN rectangle. Right click the mouse to see any special actions for a selected element in a popup menu.

The attributes of the selected element are shown in the element editor and can be modified directly. A selected element can be dragged to any other valid 'Drop Area' (highlights in RED).

Layout Management Elements

These elements serve to organize elements within the GUI.

Row Column Grid

Depress this toggle to enter 'Row Column Grid' insert mode. Use this object to organize your design elements in to rows and columns. RowCols can be nested as deep as you need.

When you are inserting (or dragging) some element a "Drop Area" will be highlight (RED) whenever the mouse passes over an empty RowCol cell or between cells or beside border cells. Inserting an element 'between' existing cells will simply expand the RowCol Grid at that point. Conversely, when you delete the last element in a row or column, that row or column will be removed from the RowCol Grid.

Paned Window

Depress this toggle to enter 'Paned Window' insert mode. This object allows you resize the individual panes after the dialog is activated and can be very useful.

Group Box

Depress this toggle to enter 'Group Box' insert mode. This object appears as an etched line with a 'label' shown near the top-left corner.

Radio Group

Depress this toggle to enter 'Radio Group' insert mode. Though not visible in the activated GUI, this element works to manage the `radio_button_items` inside itself so that only one is 'on' at a time. Radio items can ONLY be inserted into a Radio Group.

Tab Box

Depress this toggle to enter 'Tab Box' insert mode. Once inserted you will have single tab into which you can add GUI elements. Select the tab box by left-clicking the visible tab and right-click to add another tab.

When more than one tab is available you can move through the tabs using the right and left arrows that appear beside the tab. Right-click to see more actions when you have multiple tabs available.

Window Stack

Depress this toggle to enter 'Window Stack' insert mode. Once inserted you will have single window into which you can add GUI elements. Select the `window_stack` by left-clicking the visible window name and right-click to add another window to the stack.

Since a `window_stack` only shows one window at a time, some code is required to switch windows. Setting the `text_choice_item_driver` attribute of the `window_stack` to the `dialog_element_id` of an existing `text_choice_item` will trigger the Dialog Designer to link the `text_choice_item` to the window stack. When the module is generated for the design the code linking the two will be added and choosing a value with the `text_choice_item` will make that indexed window appear. Modifying the `dialog_element_id` of either the `text_choice_item` or the window stack after this link has been made may break the link – in that case re-enter the id into the `text_choice_item_driver` attribute to re-link the elements. You may not want to use a `text_choice_item` to switch the visible window but you can learn how to change the window by looking at that code in the Magik/XML tab.

Action/Input Elements

These elements are the small input widgets. Though many of these elements have a label you can set directly, that can lead to miss-aligned elements resulting in a messy and visually jarring GUI. Defining separate label elements avoids this issue. See the 'First Dialog' for an example of this technique.

Button

Depress this toggle to enter 'Button' insert mode.

Label

Depress this toggle to enter 'Label' insert mode. Insert one or many label quickly.

Change the `col_alignment` attribute to change the justification of the text within its Row Col Grid cell.

Text Input

Depress this toggle to enter 'Text Input' insert mode.

The 'activate_selector' is the method run when the user presses the 'Enter' key when this GUI element has focus. The 'change_selector' is the method that is run when the value is changed and the user either tabs to or selects another GUI element.

Unit Text Input

Depress this toggle to enter 'Unit Text Input' insert mode.

Select the Dimension (length, area, voltage, etc.), Display Units and internal storage units. The 'change_selector' is the method that is run when the value is changed and the user either tabs to or selects another GUI element.

Radio Item

Depress this toggle to enter 'Radio Item' insert mode. Radio items are simple toggles that usually appear in a group where only one value can be 'on' at time. This item can only be inserted directly into a 'Radio Group' object.

Check Box

Depress this toggle to enter 'Check Box' insert mode. Label will appear to right of the box.

Image Button

Depress this toggle to enter 'Image Button' insert mode. Change the `image_file_name` and `image_module_name` to change the icon used. If unset the `image_module_name` will default to the module generated for the dialog.

Image Toggle

Depress this toggle to enter 'Image Toggle' insert mode. Change the `image_file_name` and `image_module_name` to change the icon used. If unset the `image_module_name` will default to the module generated for the dialog.

Slider

Depress this toggle to enter 'Slider' insert mode. The label appears under the slider, left justified.

Number Input

Depress this toggle to enter 'Number Input' insert mode. The placed widget looks like a textbox but is a numerical value input with built-in validation. Specify whether or not an entered value is expected to be an integer or float, the precision and a numeric interval the value must fall into. Also choose whether to raise an alert box or simply auto-correct the entered value to the middle of the valid interval.

Text Choice / Combo Box

Depress this toggle to enter 'Text Choice' insert mode. Enter a comma delimited list of values (and strings if necessary) to populate the list.

Set the `'is_combo_box'` attribute to "Yes" to create a Combo Box.

Row Column Separator (SW 4.2+ required)

Depress this toggle to enter 'Row Column Separator' (RCS) insert mode. This widget is introduced with SW CST 4.2 and supports the quick hide/unhide of a neighboring rowcol cell. I have seen some odd behavior in my testing but I still think it is worth having in the Dialog Designer. A new Row Column Separator must be placed in an existing Row Column Grid which must have more than two elements already inserted and all elements must be in a single row or column. The DD prevents you placing an RCS in the wrong place but does not attempt to validate the placement restrictions afterwards. If you manage to create a design that does not follow the rules then it probably will have some un-desirable behavior. See the [Seventh Dialog](#).

In addition, I have often seen that only a portion of widget may be visible in the activated design when multiple RCS are inserted into a GUI ... the hide/unhide functionality is still there though.

Custom Action/Input Elements

These elements are the small handy input widgets that are useful in dialogs but are relatively hard to code. Since the requirements of these widgets are well defined the DD can generate all the needed code with minimal user input.

Style Choice

Depress this toggle to enter 'Style Selector' insert mode. Change the 'style_type' attribute to change the appearance and behavior of the generated widget. These are simple styles only, colours, line thickness, etc. When the style_type=point the drafting point styles are used to populate the selector. View the generated code to see how this was done.

File/Director Selector

Depress this toggle to enter 'File/Director' insert mode. Change the 'open_type' attribute to switch between file or directory select mode.


If you are working within a SW4.0 image, the directory selector you will see is an instance of the select_path object. To use your dialog in an image without loading the Dialog Designer you will need to copy the select_path.magik file to a location where it may be loaded during your image build.

Plugin or Action

Depress this toggle to enter 'Plugin or Action' insert mode. Though you can specify any plugin/action in the attributes you can only test activate your GUI if the source application of the plugin/action you reference is already activated. Since it is tiresome to find all the actions/plugins available to you for specific applications a right-click function has been added to the attribute editor.














Since actions can be defined to be many different types of GUI element the exact layout impact is unknown until the dialog design is activated. No attempt is made by the Dialog Designer to predict this layout impact when rendering in the 'GUI Layout' tab.

Plugin/Action Filtering

App : Professional - Smallworld Core 

Plugin : map 2/22

Action : ro 30/256

Plugin	Action
map_trail 	
map_trail	 trail_properties_dialog
map_trail	 trail_probe_near
map_trail	 trail_probe_int
map_trail	 trail_probe_tan
map_trail	 trail_probe_seg
map_trail	 trail_probe_relprot
map_trail	 trail_probe_con
map_trail	trail_delete_from_begin
map_trail	 trail_probe_cen
map_trail	 trail_probe_grid
map_trail	 trail_probe_free
map_trail	 trail_probe_absprot
map_trail	 trail_probe_perp

OK Cancel


Select the source application from the 'App' choice item. Refresh the choice list if you change the available applications using the SW Application Manager.

Plugin – the list of plugins displayed is filtered as you type in text. The number of plugins displayed / all plugin count is shown.

Action – the list of actions displayed is filtered as you type in text. The number of actions displayed / all actions count is shown.


Choose either an entire plugin (no value in the Action column and a 'plug' icon after the plugin name) or a single action (an icon is show prior to the action name if available).

The 'Ok' button will be available when a selection is made – pressing this button will update the 'plugin_name' and 'action_name' attribute immediately.

When the source plugin is set the icon will appear like this:  . When the source action is set it will use the associated action icon, if available.

Date Time Input

Depress this toggle to enter 'Date Time' insert mode. Change the 'data_type' attribute to switch between date and date-time entry. When activated the calendar drop-down date selection editor can be used for date entry, as shown below. Time values are editable directly.

07/01/2008

Mon	Tue	Wed	Thu	Fri	Sat	Sun
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

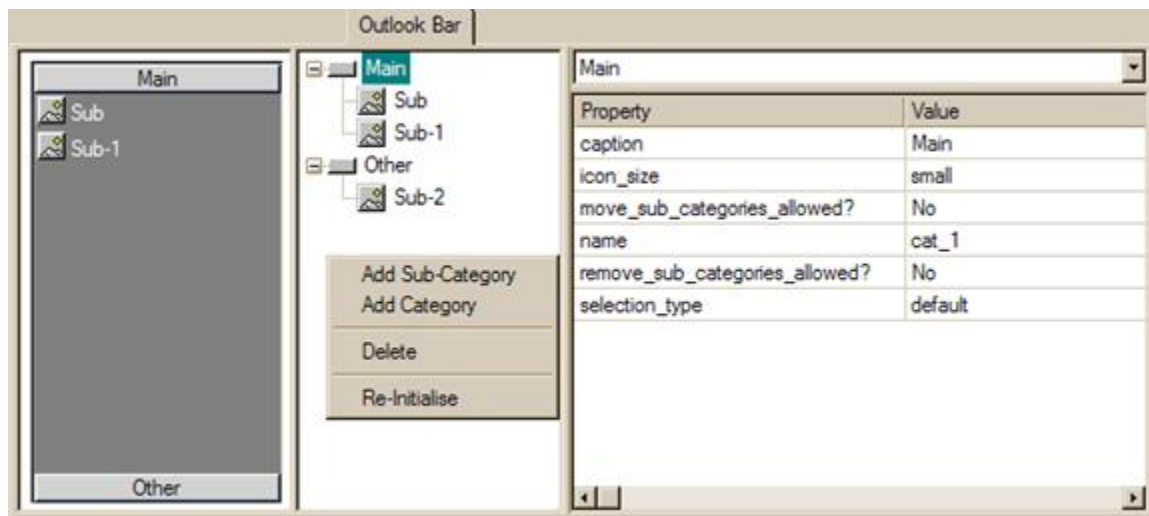
Dec 2007 January, 2008 Feb 2009

Today

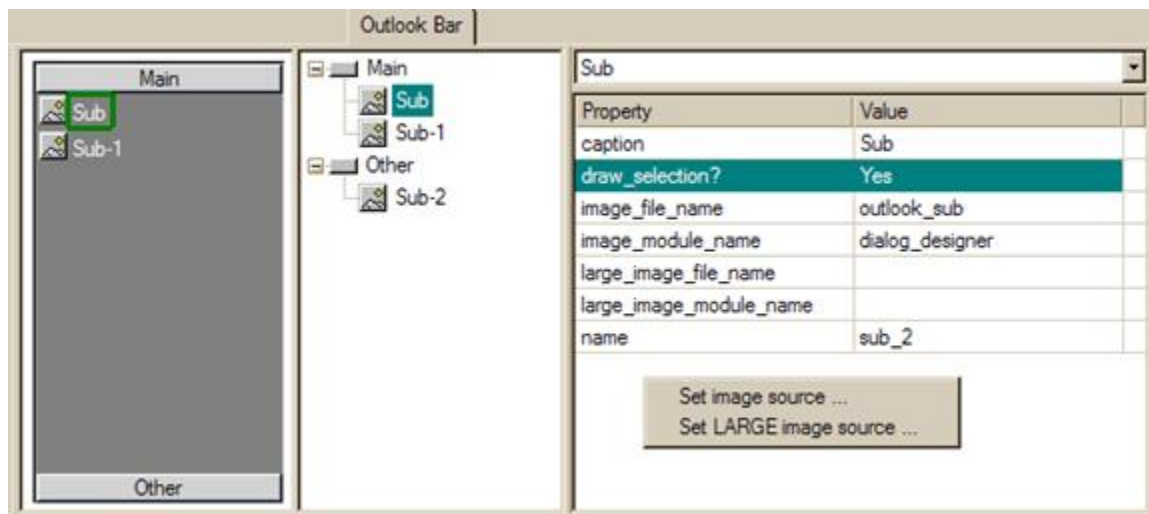
Outlook Bar

Depress this toggle to enter 'OutLookBar' insert mode. Basic attributes are editable when an instance of this widget is selected. Use the RightMouseButton menu to access the editor for defining the internal elements of the widget – it appears in another tab.

The three-pane editor shows a preview of the outlook bar, a tree of the structure and the usual attribute editor for the currently selected element. You can switch between elements by selecting one within the left or middle panes. Change the structure/order by dragging elements within the second pane and use the RightClickMenu to add categories and sub-categories. Edit values using the attribute editor.



Set the small and large image resources for a 'sub' element using through the RightClickMenu in the attribute editor.



Panel Separator

Depress this toggle to enter 'Panel Separator' insert mode. These etched lines serve to visually separate areas within a GUI and can be vertical or horizontal.

Canvas Elements

These elements are the larger widgets that gather and display data.

Text Window

Depress this toggle to enter 'Text Window' insert mode. The 'Text Window' icon will appear in the rendered dialog to differentiate it from a `tree_list` or canvas.

Simple List

Depress this toggle to enter 'Simple List' insert mode. Enter a comma delimited list of values (and strings if necessary) to populate the list.

When the 'aspect' value is set for this object the 'strings' and 'values' entries will be ignored : sample code is generated to populate the activated GUI.

Tabular List

Depress this toggle to enter 'Tabular List' insert mode. This and the 'Tree List' are just the two major types of the most complex and versatile GUI object available, the `tree_item`. Only the basic attributes are listed here. To learn how to use this object, the SW help has good explanations and the `tree_examples` module found in the 'sw_common' product has numerous clear examples.

When the 'aspect' value is set for this object, sample code is generated to populate the activated GUI.

Recordset GUI Component

Depress this toggle to enter 'Recordset GUI Component' insert mode. These widgets are often used to list the results of queries and have a number of built-in features. The generated code should put you most of the way towards implementing a clean object list ... there are a number of extra features of the `recordset_gui_component` class the DD does not expose ... view the class comments to see if they will be useful for your dialog.

Tree List

Depress this toggle to enter 'Tree List' insert mode. Only the basic attributes are made available though this is an extremely powerful GUI element. To learn how to use this object, the SW help has good explanations and the `tree_examples` module found in the 'sw_common' product has numerous clear examples.

When the 'aspect' value is set for this object, sample code is generated to populate the activated GUI.

The 'Tree List' icon will appear in the rendered dialog to differentiate it from a `text_window` or canvas.

Canvas

Depress this toggle to enter 'Canvas' insert mode. The complexity and utility of this object is found in the mouse interactions which are handled by the canvas agent. A set of example agent mouse definitions are provided in the generated code, though commented out so the user may choose the type of interactions needed. These definitions provide enough detail so a developer will be able work out other agent interaction definitions.

The 'Canvas' icon will appear in the rendered dialog to differentiate it from a text_window or tree_list.

Element Editor

In the GUI Layout, Menubar, Statusbar and Docks tabs there is also an embedded editor for the selected element in the tab. Properties can be edited directly in the editor, pressing <Enter> will update the element and refresh the GUI.

The pulldown list above the editor lists all the elements visible in the current tab. Changing the choice will put that element into the editor. Note that pulldown may not contain the complete list of all elements in the GUI if some of the elements are not currently being rendered. Some elements may, for example, be on a tab or window not currently rendered.

The screenshot shows the 'Element Editor' for the selected element 'image_button_item_01'. At the top is a pulldown menu showing 'image_button_item_01'. Below it is a table with two columns: 'Property' and 'Value'. The table contains the following data:

Property	Value
col_alignment	left
dialog_description	
dialog_element_id	image_button_item_01
has_border?	
image_file_name	image_button_item
image_module_name	
row_alignment	top
selector	image_button_item_01()

Below the table is a button labeled 'Set image source ...'.

In the example to the right, the element has an image associated to it so a function to 'Set image source ...' is available when the editor is right-clicked.

Here is the embedded GUI for selecting any icon from any module in the SW image.

This shows the 10 image files that are in modules with names containing the letter "d" (38 of 91) with names containing the letter sequence "ace". There are a total of 1765 images available.

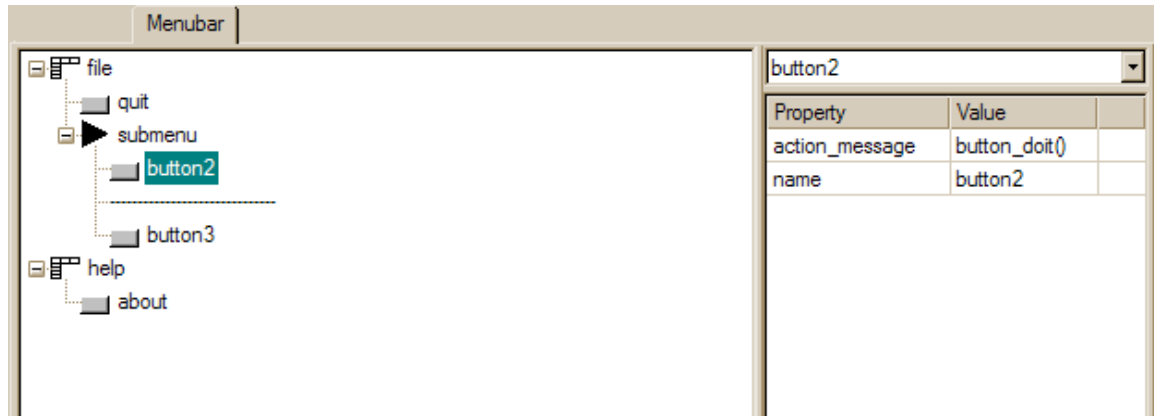
The filters act incrementally so the list will change as the filter inputs are modified.

The current image "new_ace" will be set if the 'OK' button is pressed and the image_file_name and image_module_name attributes of the selected image_button_item will be updated. The GUI will refresh, displaying the new graphic.

The screenshot shows the 'Image Filtering - Incremental' dialog box. It has three input fields at the top: 'Product' with value 'sw' and count '2/10', 'Module' with value 'd' and count '32/104', and 'Filename' with value 'ace' and count '10/1634'. Below these is a table with four columns: 'Image', 'Filename', 'Module', and 'Product'. The table contains 10 rows of image data, with 'new_ace' selected. At the bottom are three buttons: 'OK', 'Update', and 'Cancel'.

Image	Filename	Module	Product
	ace_dialog	admin_ace_plugin	sw_common
	ace_dialog_32x32	admin_ace_plugin	sw_common
	ace_dialog_grey	admin_ace_plugin	sw_common
	delete_ace	admin_ace_plugin	sw_common
	delete_ace_grey	admin_ace_plugin	sw_common
	new_ace	admin_ace_plugin	sw_common
	new_ace_grey	admin_ace_plugin	sw_common
	traceback_viewer	dev_tools_application	sw_dev_tools
	place1	editor_plugin	sw_common
	place1_grey	editor_plugin	sw_common

Menubar

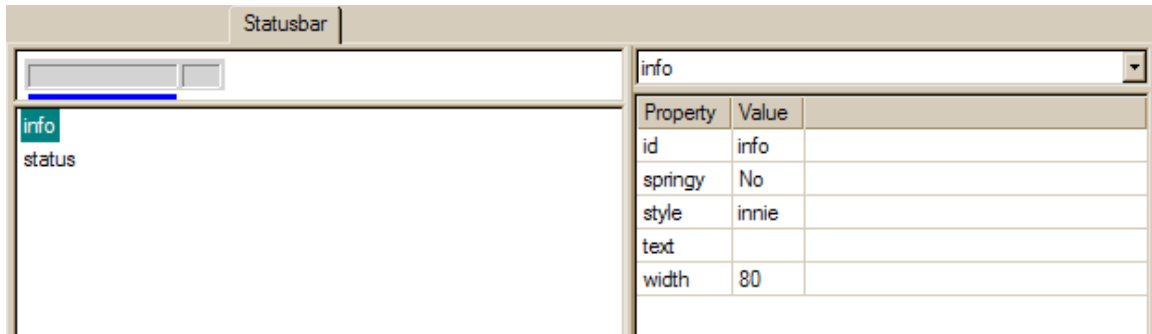


Select a menubar element to view/edit its attributes in the editor, or right-click to access other functions.

- Create
- Delete
- Drag
- Edit
- Rename
- Buttons / Pullout Submenus / Separators

Selecting the rendered menubar within the GUI Layout tab will bring the Menubar tab to the front so the menubar configuration may be inspected and edited.

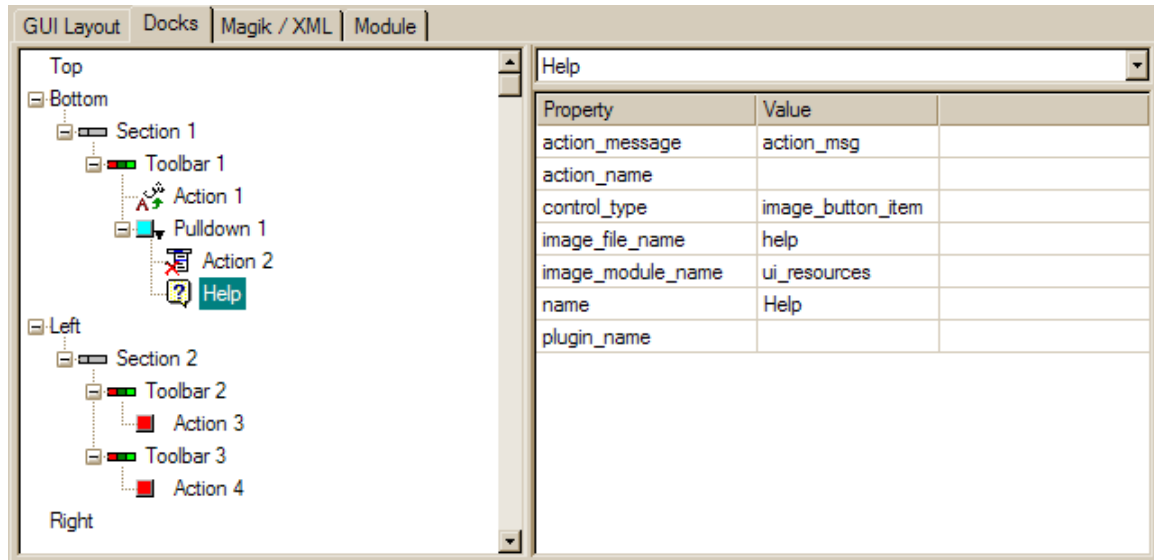
Statusbar



Select a statusbar element to view/edit its attributes in the editor, or right-click to access other functions.

- Create a new pane
- Create a new progress_bar pane
- Delete the selected pane
- Drag a pane to new position
- Edit the attributes of a pane

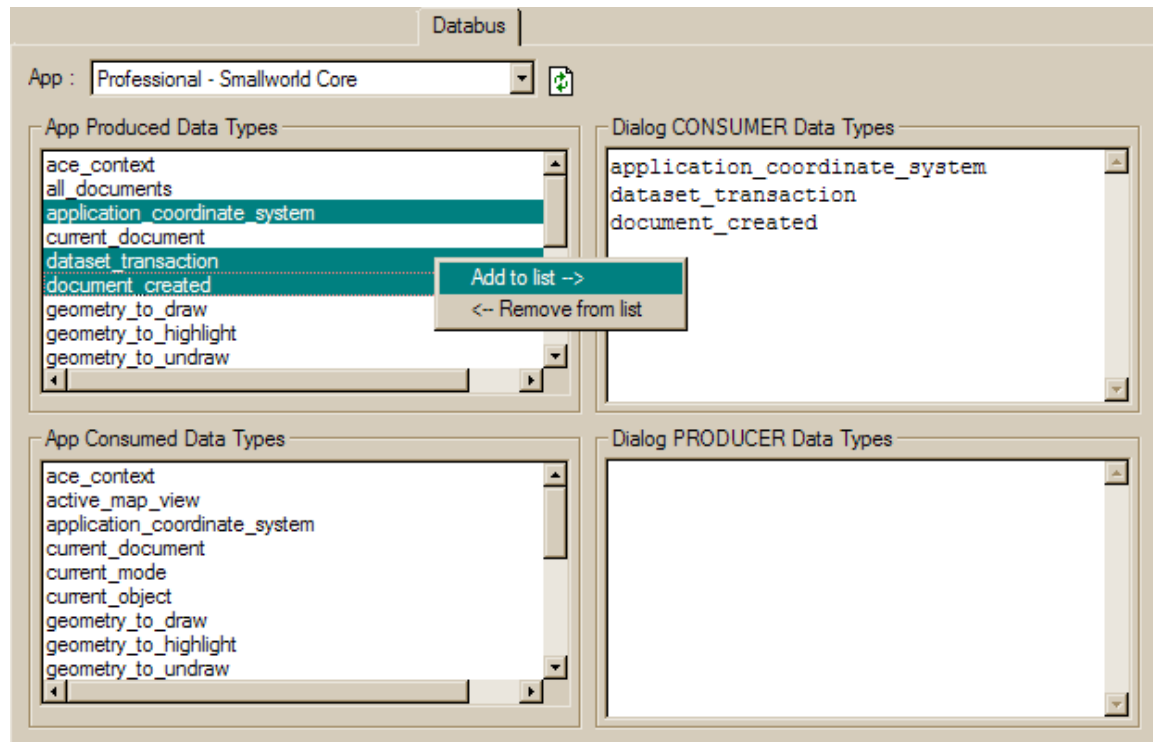
Docks



Select a docks element to view/edit its attributes in the editor, or right-click to access other functions.

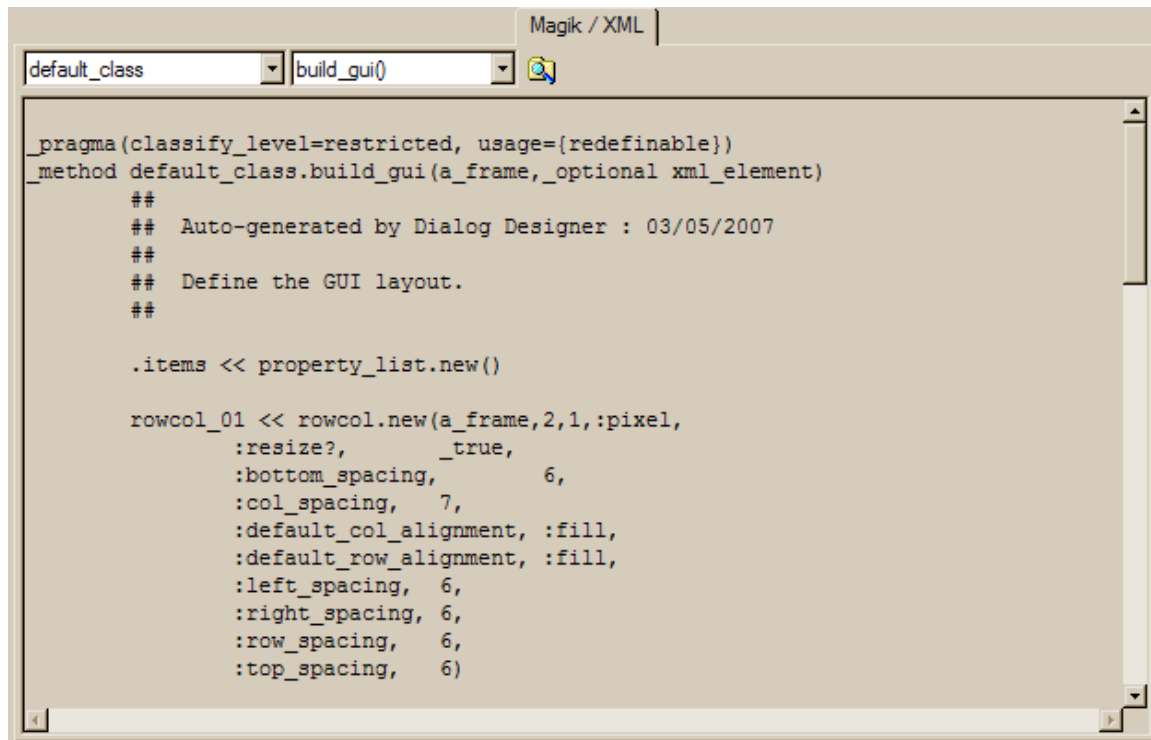
- Create a section, toolbar or action
- Delete a section, toolbar or action
- Drag a section, toolbar, action or action-group to another location
- Rename a section, toolbar or action
- Add an image pulldown action-group to a toolbar

Databus



- Right-click left list to add/remove elements from list at right
- Edit Dialog lists (at right) directly

Magik / XML



The screenshot shows a software window titled "Magik / XML". Inside the window, there are two dropdown menus at the top: the first is set to "default_class" and the second is set to "build_gui()". Below these menus is a text area containing the following code:

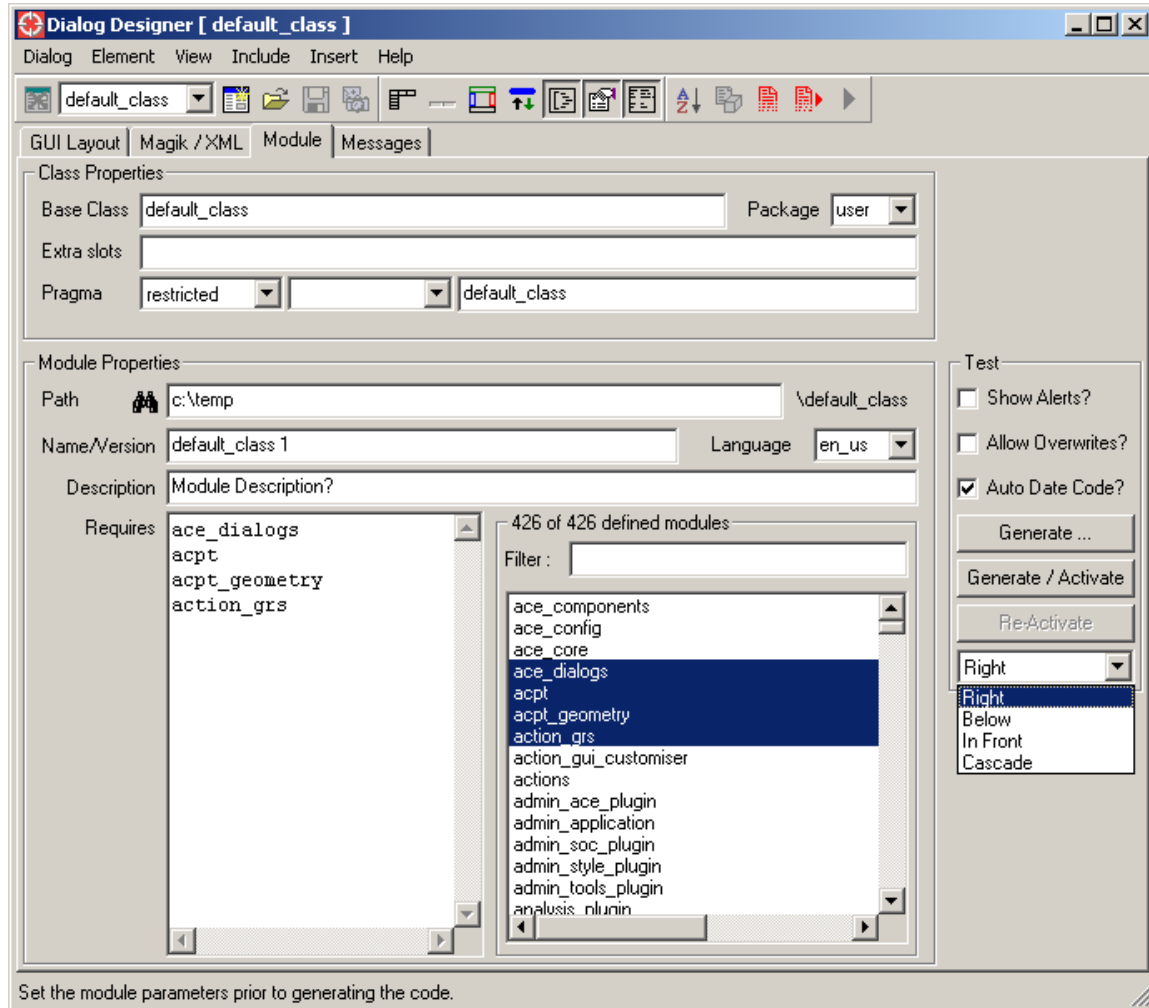
```
_pragma(classify_level=restricted, usage={redefinable})
_method default_class.build_gui(a_frame,_optional xml_element)
##
## Auto-generated by Dialog Designer : 03/05/2007
##
## Define the GUI layout.
##

.items << property_list.new()

rowcol_01 << rowcol.new(a_frame,2,1,:pixel,
    :resize?,      _true,
    :bottom_spacing, 6,
    :col_spacing,   7,
    :default_col_alignment, :fill,
    :default_row_alignment, :fill,
    :left_spacing,  6,
    :right_spacing, 6,
    :row_spacing,   6,
    :top_spacing,   6)
```

- Class choice
- Method choice
- Explore code
- Select / Copy to system clipboard
- All code is auto-generated

Module



Set CLASS options:

- Base Class for auto-generated code
 - Must start with a letter but may also contain under_scores and digits.
 - Cannot contain the strings 'plugin' or 'framework' as these strings are referenced when auto-generating the dialog code
 - Trailing underscores remove
- Compilation package for generated code
- Specify extra slots for the base class
- Set the 'Classify Level', 'Usage(s)' and 'Topic(s)' that are included in pragma statements preceding magik code blocks.

Set MODULE options:

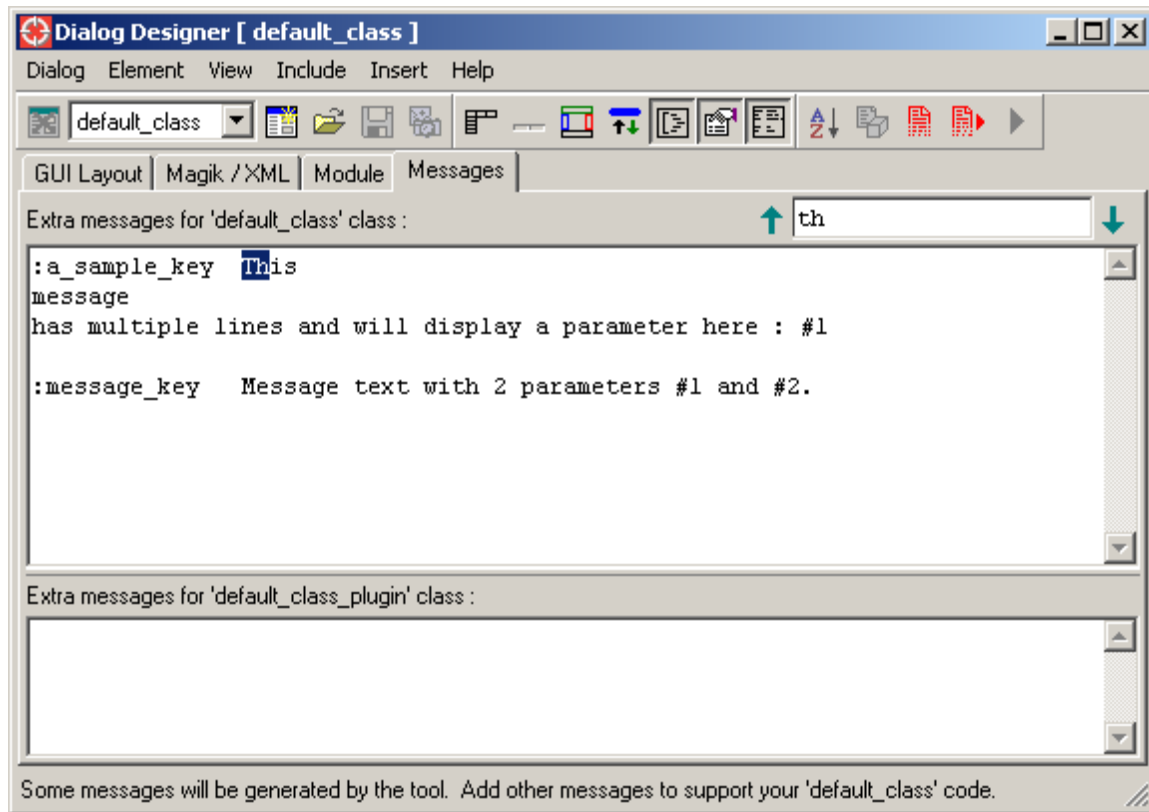
- Base Directory Path (shown above as c:\temp)
- Name and Version
- Language (drives the message resource directory)
- Description

- Requires listing that can be added to from an incrementally filtered module list (add/remove via right-click menu)

TEST your dialog:

- 'Show Alerts?' toggle (controls auto hook generation)
- 'Allow Overwrites?' toggle (controls overwriting of generated code that you have modified since it was generated)
- 'Auto Date Code?' will suppress the date/time stamping of the generated code if unchecked. This can be useful if you want to compare versions of the generated code.
- Build ... (the module with all associated magik and XML)
 - Read the "Testing Dialogs" section that appears near the end of this document.
- Build & Activate
- Activate (launch the generated dialog)
 - This simply reloads the module files and launches the GUI so you can hand manipulate the code and test what happens easily.
- Activation position (relative to the DD)
 - Right (default)
 - Below
 - In Front
 - Cascade


Messages



Define messages that will be added to the end of the BASE_CLASS.msg (upper) and BASE_CLASS_plugin msg (lower) files. Quite often you will need to define messages on the GUI class in support of the behavior you are developing. Define them here; they will be saved in the XML description of the dialog. Use the search tool to find some text in the base_class messages quickly ... the graphic shows the result of searching for 'th'.

To force an blank line in your message (this is handy when sending long messages to an alert box) put at least one space in the 'blank' line so Dialog Designer will correctly save/read it to/from XML.

When creating GUI functionality these custom messages are a convenient way to update labels and status bar information. As your GUI grows and evolves the number of messages will grow

... press the "Sort Messages" button  to sort all your messages alphabetically by their message keys.

Embedded messages

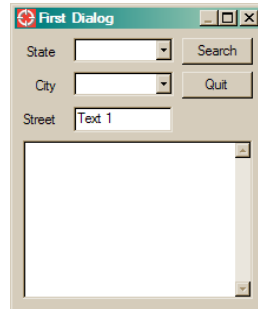
With DD, you can embed you message key and text into the GUI and the DD will define the effected GUI elements with `_self.message(<message_key>)` calls so that you can easily internationalize your code as needed. See the [Seventh Dialog](#) for examples.

Sample Dialogs

The following dialogs introduce key techniques and reveal the process of creating and testing new dialogs. Though the samples are independent, they are presented in ascending order of complexity so you are encouraged to start with 'First Dialog'.

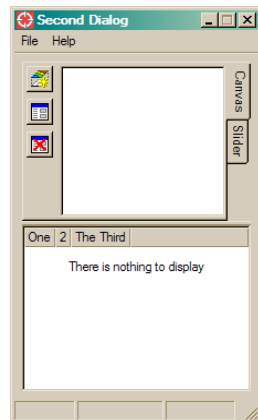
First Dialog

- 3 label items
- 2 choice items
- 1 text input
- 2 command buttons
- 1 text_window
- disabled frame minimize button
- text/choice items resize as the dialog is resized



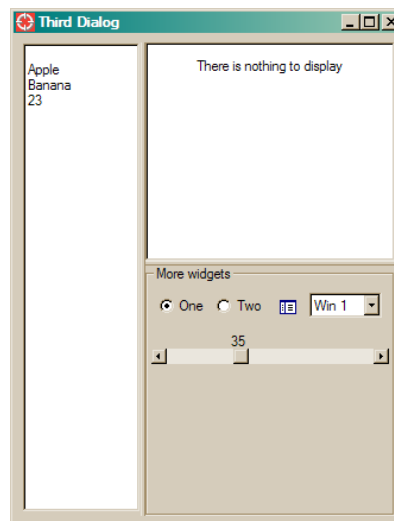
Second Dialog

- menubar
- paned window
- tab box, tabs on right border
- image buttons and canvas in first tab
- slider in second tab
- tabular list on second window pane
- disabled frame maximize button
- paned windows resize vertically in 80/20 ratio
- statusbar with 3 panes (1st and 3rd are stretchy)



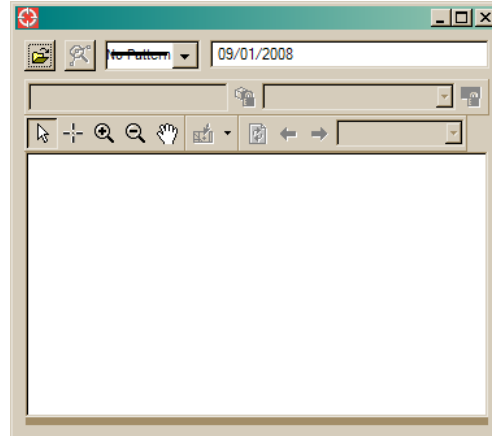
Third Dialog

- frame not resizable
- nested paned windows
 - left window resizable left-right
 - right windows resizable up-down
- group_box
 - radio buttons
 - image toggle with flat appearance
 - choice item that drives the window stack below
- slider on the first window
- text window on the second window (hidden in the graphic shown here)
- dialog consumes the :map_trail and produces :post_render_sets from/to the application databus



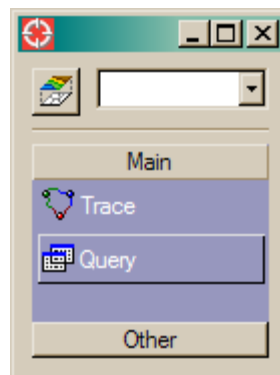
Fourth Dialog

- Directory Selection button
 - map_manager 'view_zoom_to_trail'
 - action**
 - Style choice item
 - date-time input
 - embedded spatial_context_viewer *plugin*
-
- base class = 'fourth_dialog'
 - package = "user"
 - extra slots = :one and :two
 - pragma code level = debug
 - pragma code usage(s) = {fourth,test}
 - pragma code topic(s) = {fourth_dialog,test}



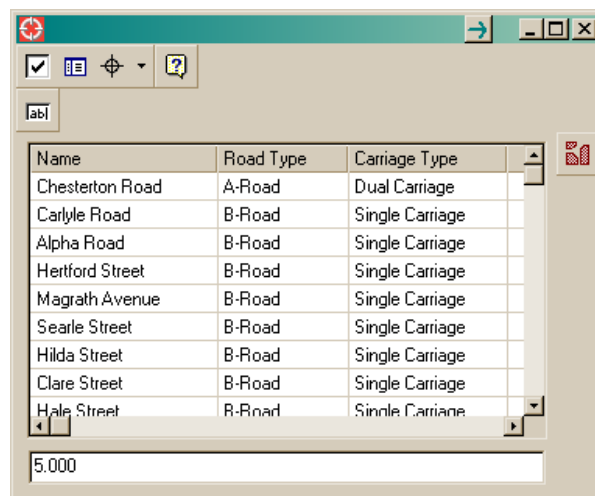
Fifth Dialog

- Outlook bar
- panel separator
- language selection



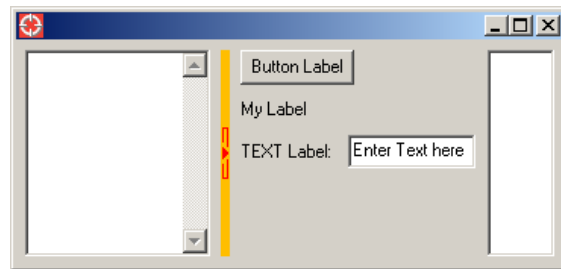
Sixth Dialog

- Recordset GUI Component
- Managed numeric input
- user defined messages
- dock actions



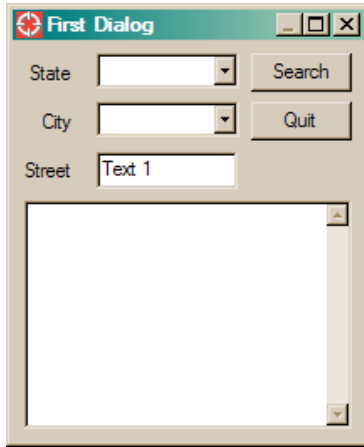
Seventh Dialog

- Row Column Separator
- Embedded Messages
- Message Search



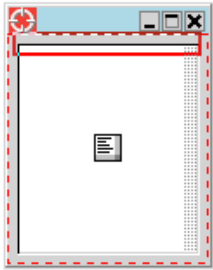
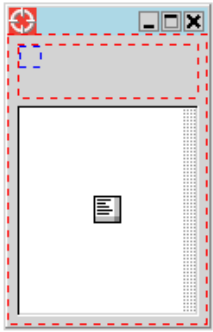

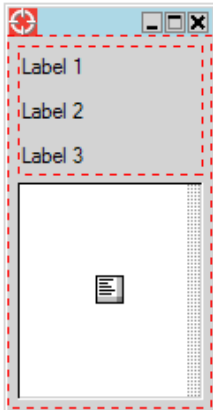

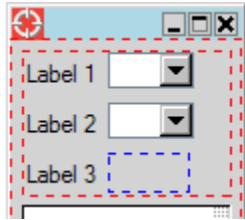

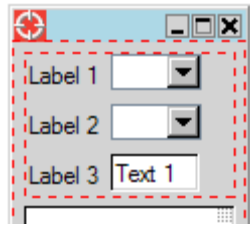
First Dialog


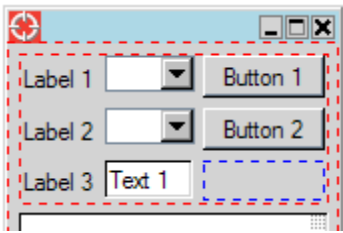

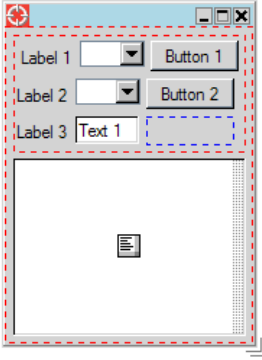


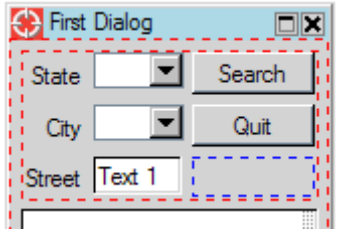
The graphic below shows a simple ‘Street Search’ dialog. The key features of this dialog are listed to the right.

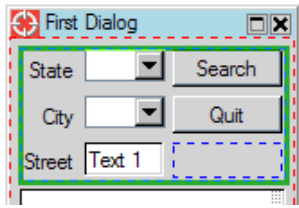

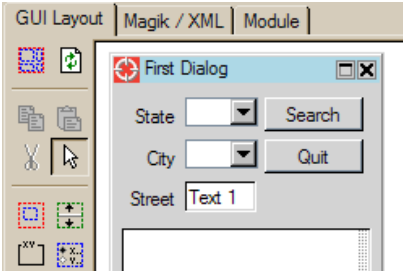


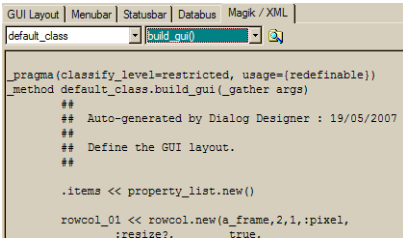


- 3 label Items
- 2 choice items
- 1 text input
- 2 command buttons
- 1 text_window
- disabled frame minimize button
- text/choice items resize as the dialog is resized

<p>The GUI Layout tab contains the default Dialog, an empty dialog ready for filling with GUI elements. The red dashed rectangle is a RowCol Grid element and the blue dashed square within is a valid cell into which any GUI element may be placed. The expansion chevron at the lower right corner indicates the outer frame is resizable.</p>	
<p>Press the text_window icon or use the Menu : Insert – Text Window. This changes you to “TextWindow” insert mode”.</p>	
<p>Use the mouse to point at the blue square in the default Dialog, a red outline will appear indicating a valid drop spot.</p>	
<p>Click to insert the text window. A simple list would work just as well but we’ll stick with the text_window. The window is inserted and rendered. Note the icon rendered in the middle of the text window to indicate what type of widget it is – there are a number of similar looking widgets (canvas, simple_list, tree_list and text_window)</p>	
<p>Change to “RowColumnGrid” insert mode.</p>	

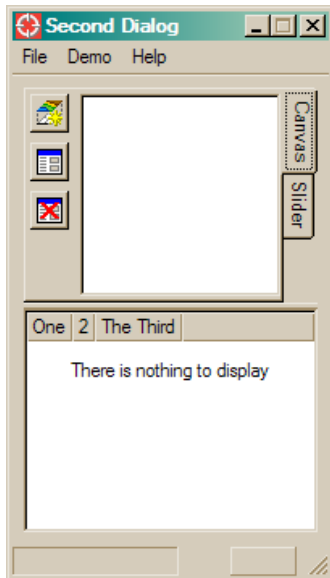
<p>Use the mouse to point at the top border of the text-window, when the red drop area is shown click to insert.</p>	
<p>Now you have a place for all the other widgets.</p>	
<p>Change to “Label” insert mode  and insert 3 labels in a column.</p>	
<p>Change to “TextChoice” insert mode  and insert 3 choice items in the next column.</p>	
<p>Change to “TextInput” insert mode  and insert one below the choice items.</p>	

<p>Change to “CommandButton” insert mode  and insert 2 buttons in the third column.</p>	
<p>Now that we have all the widgets in place we can edit them. We could do these edits as the widgets were inserted but it is simpler to explain this way.</p> <p>Change to “Select” mode by pressing the ‘Space’ bar or press .</p>	
<p>Select the title bar of your Dialog. A green rectangle will outline the currently selected dialog element and its properties will appear in the listing to the right.</p>	
<p>Change the value for ‘minimizable?’ to “No” and the title to “First Dialog” (no quotes needed). The dialog will update as you make changes.</p>	
<p>Select the labels and change their values to State, City and Street respectively and their col_alignment to ‘right’.</p> <p>Select the command buttons and change their labels to “Search” and “Quit”.</p>	

<p>Select the RowColumnGrid by clicking in the blue rectangle below 'Quit'.</p> <p>Change the col_resize_values to "0,100,0" (no quotes needed)</p>	
<p>Now that the design is done, de-toggle the 'Show RowColumnGrid' toggle  to see a clean preview of the GUI. Select anywhere outside the rendered dialog to temporarily clear the 'current selection' green rectangle.</p>	
<p>Build and activate your dialog by pressing the Build / Activate button. The new dialog should activate just to the right of the Dialog Designer.</p> <p>Grab the lower right corner and resize the dialog to check that the correct elements are resizing as expected.</p>	
<p>Now that the dialog is created you can explore the code through the "Magik / XML" tab. Either choose a class and method to view it immediately or press the  button to explore the generated module.</p>	 <pre> GUI Layout Menubar Statusbar Databus Magik / XML default_class build_gui() _pragma(classify_level=restricted, usage=(redefinable)) _method default_class.build_gui(_gather args) ## ## Auto-generated by Dialog Designer : 19/05/2007 ## Define the GUI layout. ## .items << property_list.new() rowcol_01 << rowcol.new(a_frame,2,1,:pixel, :resize?, true, </pre>


Second Dialog

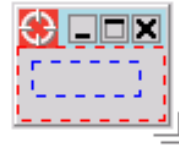
Now that you have the basics down, let's create this one:



Design elements :

- menubar
- paned window
 - tab box, tabs on right border
 - buttons and canvas in first tab
 - slider in second tab
- tabular list on second window pane
- disabled frame maximize button
- window panes resize vertically in 80/20 ratio
- statusbar with 3 panes
 - 1st and 3rd are stretchy, 2nd flat, 3rd outie

If you don't have a blank dialog design then start a new dialog by pressing  or using the shortcut Ctrl-n.



Let's start at the top ... select the frame, set the title to "Second Dialog" and set 'maximizable?' to "No". Remember that typing the first letter of a valid choice is sufficient to set the value.

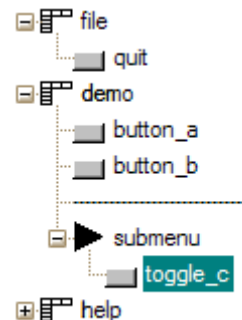
maximizable?	No
minimizable?	Yes
resizable?	Yes
title	Second Dialog

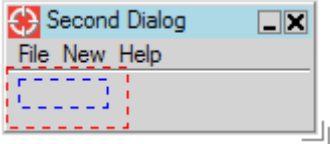


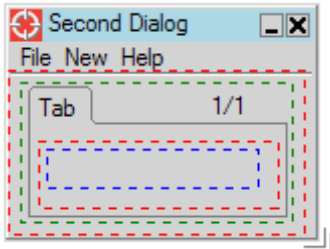
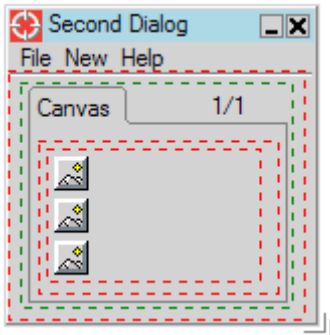
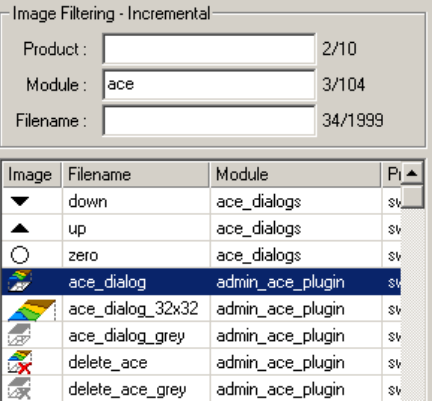















Depress the menubar toggle  to enable that tab.


Add a menu just before 'file' by selecting 'file' then select "Add Menu" from the right-click popup menu – the 'new' menu is created with a single button.

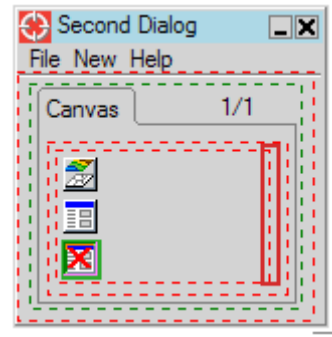
To change the order of the menus, just drag one onto another. Drag 'new' onto 'help'. Rename 'new' to 'demo'.

Expand the 'new' menu. Select the 'button' and use the right-click menu to add a button, separator and pullout menu. Rename the buttons to button_a, button_b and toggle_c (in the sub-menu). Change the 'control_type' of toggle_c to 'toggle_item'.




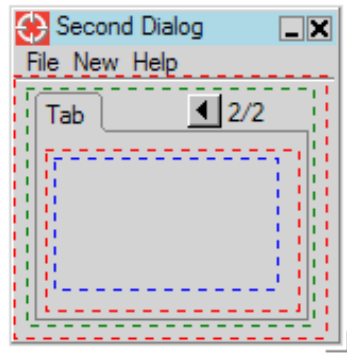
<p>Change back to the 'GUI Layout' tab – note the menubar now contains the 'Demo' menu.</p>																																					
<p>Now for the main GUI Layout elements ...</p> <p>Click the “PanedWindow” toggle  to switch to that insert mode and insert one into the empty rowcol.</p> <p>Insert the TabBox , the default tab_box is inserted with 1 page with a tab label = “Tab”.</p>																																					
<p>Select the RowCol in the tab and change the ‘tab_label’ to “Canvas”.</p> <p>Insert a RowCol inside the existing RowCol and 3 image buttons vertically.</p> <p>Switch back to select mode by pressing the space bar.</p> <p>Select the top image_button and then right-click in the editor to start the image-selector...</p>																																					
<p>Type ‘ace’ into the ‘Module’ incremental filter. Only two modules have that text string in their name so the icon list has only 24 icons in it. Select the “ace_dialog” icon from the “admin_ace_plugin”.</p> <p>If this icon is not in your list, it just means that the admin_ace_plugin is not registered in your image. Choose some other icon instead.</p> <p>Click the “OK” button to set the image for the selected image_button_item.</p> <p>Experiment with the incremental filter and set the images on the other two image_buttons.</p>	 <table><tr><th>Image</th><th>Filename</th><th>Module</th><th>Pi</th></tr><tr><td>▼</td><td>down</td><td>ace_dialogs</td><td>sv</td></tr><tr><td>▲</td><td>up</td><td>ace_dialogs</td><td>sv</td></tr><tr><td>○</td><td>zero</td><td>ace_dialogs</td><td>sv</td></tr><tr><td></td><td>ace_dialog</td><td>admin_ace_plugin</td><td>sv</td></tr><tr><td></td><td>ace_dialog_32x32</td><td>admin_ace_plugin</td><td>sv</td></tr><tr><td></td><td>ace_dialog_grey</td><td>admin_ace_plugin</td><td>sv</td></tr><tr><td></td><td>delete_ace</td><td>admin_ace_plugin</td><td>sv</td></tr><tr><td></td><td>delete_ace_grey</td><td>admin_ace_plugin</td><td>sv</td></tr></table>	Image	Filename	Module	Pi	▼	down	ace_dialogs	sv	▲	up	ace_dialogs	sv	○	zero	ace_dialogs	sv		ace_dialog	admin_ace_plugin	sv		ace_dialog_32x32	admin_ace_plugin	sv		ace_dialog_grey	admin_ace_plugin	sv		delete_ace	admin_ace_plugin	sv		delete_ace_grey	admin_ace_plugin	sv
Image	Filename	Module	Pi																																		
▼	down	ace_dialogs	sv																																		
▲	up	ace_dialogs	sv																																		
○	zero	ace_dialogs	sv																																		
	ace_dialog	admin_ace_plugin	sv																																		
	ace_dialog_32x32	admin_ace_plugin	sv																																		
	ace_dialog_grey	admin_ace_plugin	sv																																		
	delete_ace	admin_ace_plugin	sv																																		
	delete_ace_grey	admin_ace_plugin	sv																																		


Change to 'Canvas' insert mode  and insert to the right of the RowCol with the image buttons, the insertion area is highlighted in RED as show here.



To add the second tab, select the TabBox by clicking on the tab (where 'Canvas' appears). Right-click and "Add tab".

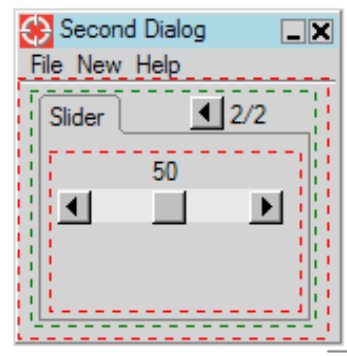
The second tab is created and displayed. To switch back to the first tab click the  button. Tab can be added, deleted and moved within the tab order ... all via the right-click menu when the tab_box is selected.




Select the interior RowCol and set the 'tab_label' to "Slider". Change to 'Slider' insert mode  and insert one in the tab.

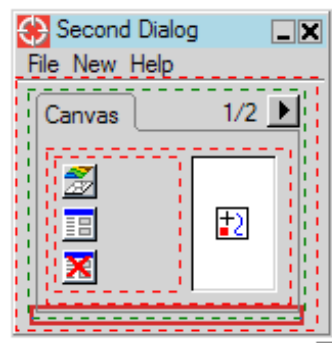
Lastly, we want the tabs on the right so select the TabBox again and set the 'tab_location' to "right".

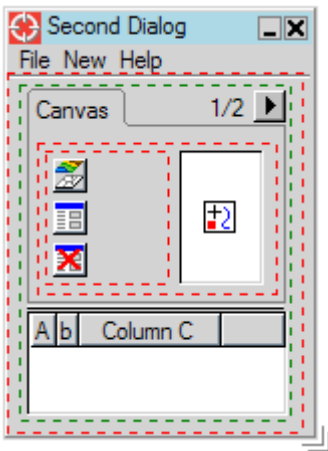

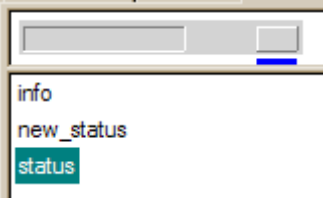
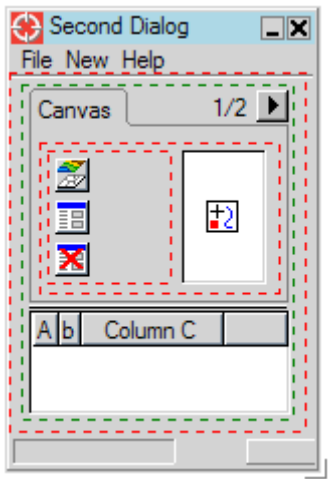

The render will NOT show the tab on the right, maybe that feature will make it into Version 2.



Return to the first tab, change to 'Tabular List' insertion mode  and insert it below of the TabBox, the insertion area is highlighted in RED as show here.

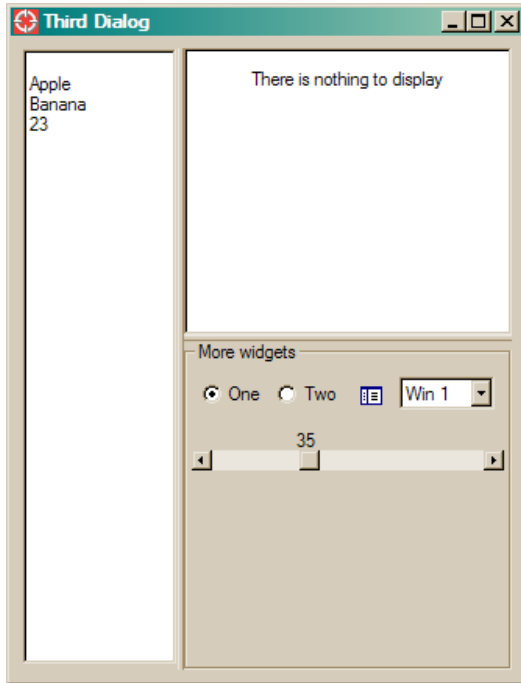
Be careful with this insertion as it is easy to insert it below the paned_window instead at the bottom of the PanedWindow. When you are hovering before you click to insert move the mouse slightly up and down the design to see the change in the insertion area.



<p>Ok, your GUI should look something like the graphic shown here.</p> <p>Select the PanedWindow by selecting it from the drop-down list above the editor. Change the 'row_resize_values' to "80,20" (no quotations needed).</p>	
<p>Depress the statusbar toggle  to activate that tab.</p> <p>Select 'status' then right-click and "Add Pane". You can change which pane you have selected by clicking the graphic, list or choosing from the drop down list above the editor.</p> <p>Set the style of the new_status pane to "flat" and the style of the "status" pane to "outie".</p> <p>Set the "springy" of the "info" pane to Yes.</p>	
<p>Return to the "GUI Layout" tab, you should see something like the graphic show here.</p>	
<p>Build and Launch your design by pressing  or explore the auto-generated code through the "Magik / XML" tab or both.</p>	


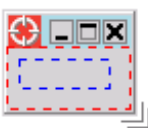
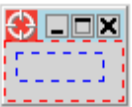
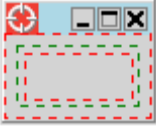
Third Dialog

This dialog has all the widgets not appearing in the first two example dialogs ...

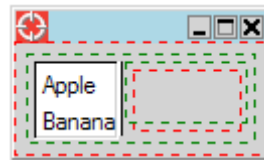


Design elements :

- frame not resizable
- nested paned windows
 - left window resizable left-right
 - right windows resizable up-down
- group_box
 - radio buttons
 - image toggle with flat appearance
 - choice item that drives the window stack
- slider on the first window
- text window on the second window (hidden in the graphic shown here)
- dialog consumes the :map_trail and produces :post_render_sets from/to the application databus
- base class = 'third_dialog'

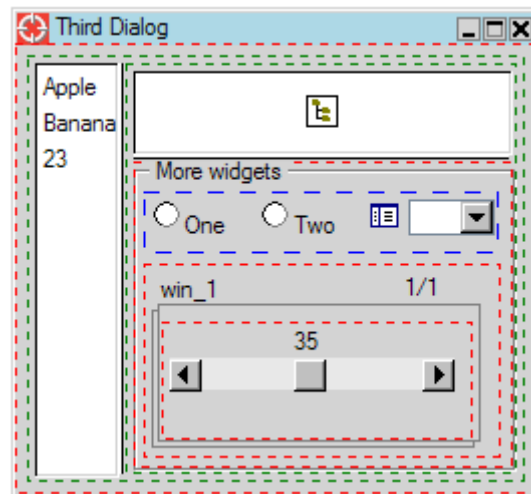
<p>If you don't have a blank dialog design then start a new dialog by pressing  or using the shortcut Ctrl-n.</p>	
<p>Select the frame, set 'resizable?' to "No". The dialog will refresh without the 'resizable?' chevron.</p>	
<p>Change to "PanedWindow" mode and insert one. Select the PanedWindow and set the 'side_by_side?' attribute to "Yes".</p>	

Change to “PanedWindow” mode and insert one inside the previous PanedWindow, to the right of the SimpleList.



Insert a TreeList in the new PanedWindow and a GroupBox below it. In the GroupBox insert a RadioGroup and a WindowStack. In the RadioGroup insert two RadioButtons, an image toggle and a TextChoice. In the WindowStack insert a Slider.

Select the GroupBox and set the label to “More Widgets”. Select the RadioButtons and set their labels to “One” and “Two”. Set the image toggle image to “open” from “ui_resources” and ‘has_border?’ to “No”. Select the TextChoice and change its dialog_element_id to “ch1”. Select the Slider and set the value to 35.



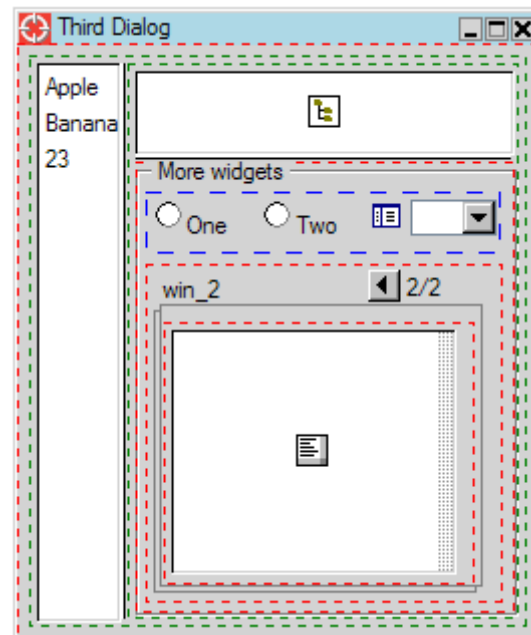
Select the WindowStack, right-click and “Add window to stack”. Insert a TextWindow onto the new window.



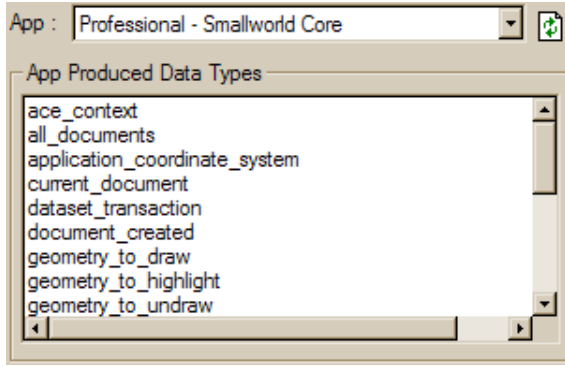
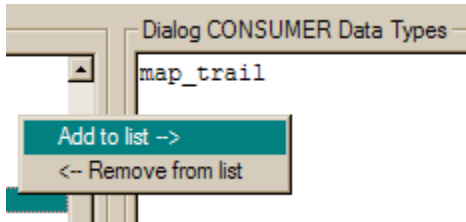
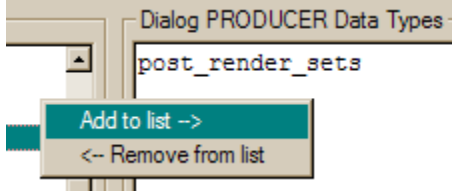

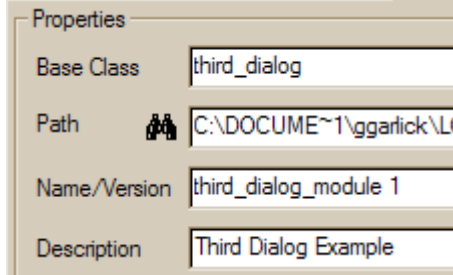

Select the TextChoice and note its ID, the dialog_element_id (changed to “ch1” in previous step). Now select the WindowStack and set the text_choice_item_driver to the ID you noted. This links these two elements via their current dialog_element_ids. If you change either of the IDs the link will be broken. If you do break the link, re-edit the text_choice_item_driver value to re-link them.

Select the TextChoice and you will see that the strings and values are now populated as shown here:

strings	Win 1,Win 2
values	win_1,win_2

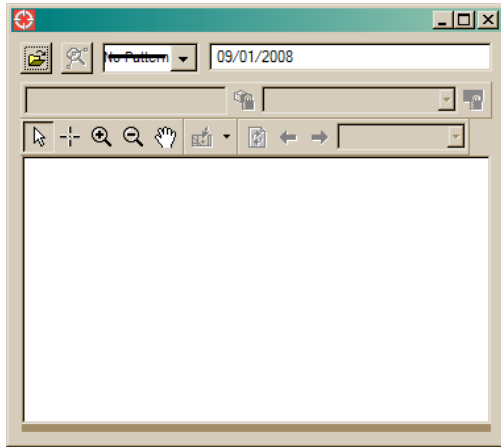
That’s it for the GUI.



<p>Press the Databus toggle  to activate that tab. I also have the “Professional – Smallworld Core” activity running so it is included in the “App” choice list. You can start any available application using the Smallworld Application Manager and the press the refresh  to update the ‘App’ lists that show all the data_types that this application is has producers or consumers of.</p>	
<p>Scroll down to the “map_trail” in the “App Produced Data Types” list, select it, right-click and “Add to List →” to copy it to the “Diag CONSUMER Data Types”.</p> <p>You could just type “map_trail” into the target list if you were sure of the spelling.</p>	
<p>Scroll down to the “post_render_sets” in the “App Consumed Data Types” list, select it, right-click and “Add to List →” to copy it to the “Diag Producer Data Types”.</p> <p>You could just type “post_render_sets” into the target list if you were sure of the spelling.</p>	
<p>Lastly, press the “Module” toggle  to activate that tab. Change the ‘Base Class’ to “third_dialog” and the Name/Version to “third_dialog_module 1”</p>	
<p>Build and Launch your design by pressing  or explore the auto-generated code through the “Magik / XML” tab or both.</p>	


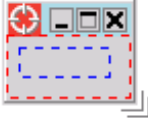


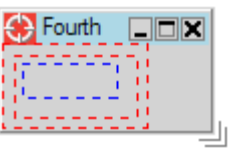

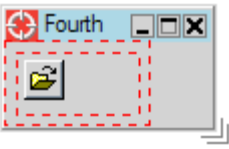

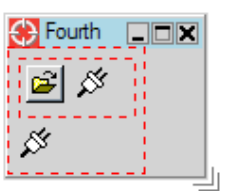
Fourth Dialog


This dialog shows recently developed widgets.

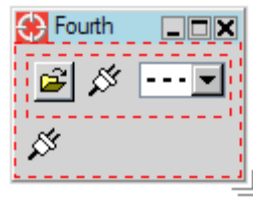



- Directory Selection button
- map_manager 'view_zoom_to_trail' **action**
- Style choice item
- date-time input
- embedded spatial_context_viewer **plugin**
- base class = 'fourth_dialog'
- package = "user"
- extra slots = :one and :two
- pragma code level = debug
- pragma code usage(s) = {fourth,test}
- pragma code topic(s) = {fourth_dialog,test}

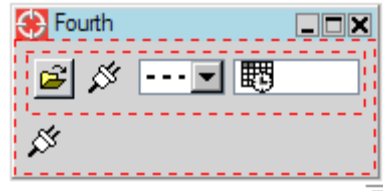
This dialog references the 'maps' and 'context_viewer_plugin' plugins so please start, for example, the 'Professional – Smallworld Core' application.

If you don't have a blank dialog design then start a new dialog by pressing  or using the shortcut Ctrl-n.	
Select the frame and change the title to "Fourth". The dialog will resize so the title can be seen.	
Change to "Row Column Grid" mode (icon : ) and insert one.	
Change to "File or Directory Selection" mode (icon : ) and insert one. Press <Space> to change to selection mode, select the widget you just inserted and change the 'operation' to 'select_directory'.	
Change to "Plugin or Action" (icon: ) mode and insert one to the right of the previous widget and below the inserted 'Row Colum Grid'.	

Change to “Style Choice” (icon: ) mode and insert one to the right of the upper ‘plugin’ widget. Press <Space> to change to selection mode, select the widget you just inserted and change the ‘style_type’ to ‘dash_pattern’.

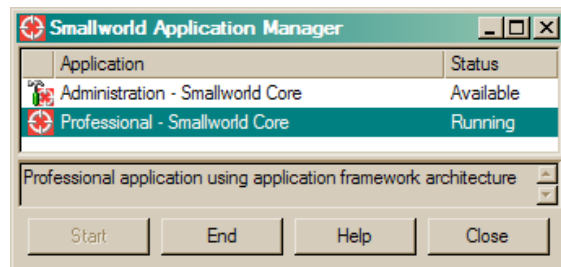


Change to “Date Time” mode (icon: ) and insert one next to the style widget. Select it and change the data_type to ‘date_time’. Change the dialog_element_id to “dt1” and press enter. Right-click in the element editor and “Set ‘Selector’ Values” – the ‘aspect’ and ‘value_change_notifier’ will update.



If you have not already done so. Start the ‘Profesional – Smallworld Core’ activity now.

Wait for the activity to start before proceeding to the next step.

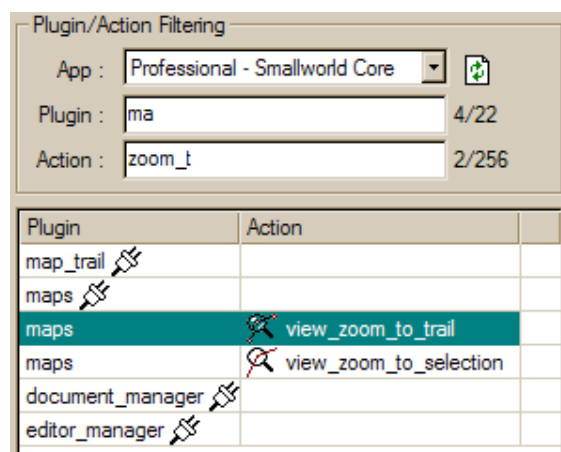


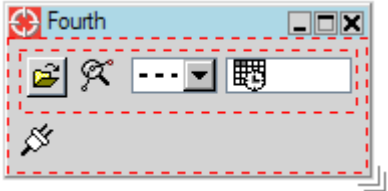
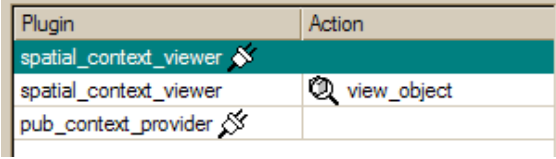
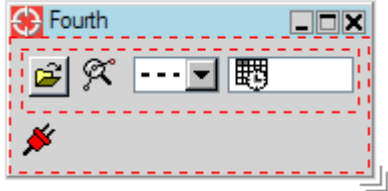
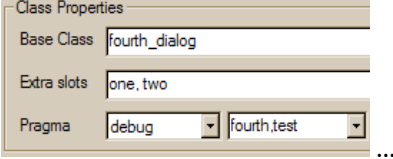
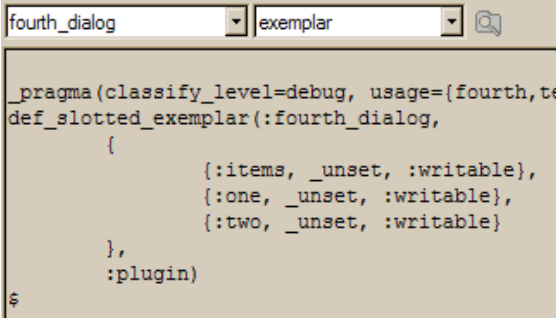
Press <Space> to change to selection mode, select the upper plugin/action widget. Right click in the editor and ‘Set plugin / action’.


The interface for selecting an action or plugin from the available activities will appear.

The action we want is on the plugin ‘maps’ called ‘view_zoom_to_trail’ so enter a few letters of each of these in the input boxes. The list will shorten to suit and you can select the action as show at right.

Double click the selection or select and press the ‘Ok’ button.



<p>The design rendering will show the action icon.</p>									
<p>Select the lower plugin/action widget. Right click in the editor and 'Set plugin / action'.</p> <p>Select the 'spatial_context_viewer' plugin.</p>	 <table border="1"> <thead> <tr> <th>Plugin</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>spatial_context_viewer</td> <td>view_object</td> </tr> <tr> <td>spatial_context_viewer</td> <td>view_object</td> </tr> <tr> <td>pub_context_provider</td> <td></td> </tr> </tbody> </table>	Plugin	Action	spatial_context_viewer	view_object	spatial_context_viewer	view_object	pub_context_provider	
Plugin	Action								
spatial_context_viewer	view_object								
spatial_context_viewer	view_object								
pub_context_provider									
<p>Now the lower plugin/action is RED to show it is set to some plugin. You will not see what the plugin will appear until it is test-launched.</p>									
<p>Change to the 'Module' tab and change the base_class to "fourth_dialog". Leave the package as "user". Add a couple of slot names like "one, two" to "Extra Slots".</p> <p>Change the Pragma code level to 'debug', enter "fourth, test" for code usage(s) and add ",test" to the list of code topics.</p>	 <p>Class Properties</p> <p>Base Class: fourth_dialog</p> <p>Extra slots: one, two</p> <p>Pragma: debug fourth.test</p> <p>Package: user</p> <p>fourth_dialog.test</p>								
<p>Before we build and launch this dialog let's look at the changes to the code being auto-generated.</p> <p>Switch to the 'Magik/XML' tab and look at the 'exemplar' of the 'fourth_dialog' class.</p> <p>Notice the changes to the pragma statement and the inclusion of the extra slots you added earlier.</p>	 <pre> fourth_dialog exemplar _pragma(classify_level=debug, usage={fourth,te def_slotted_exemplar(:fourth_dialog, { {:items, _unset, :writable}, {:one, _unset, :writable}, {:two, _unset, :writable} }, :plugin) \$ </pre>								

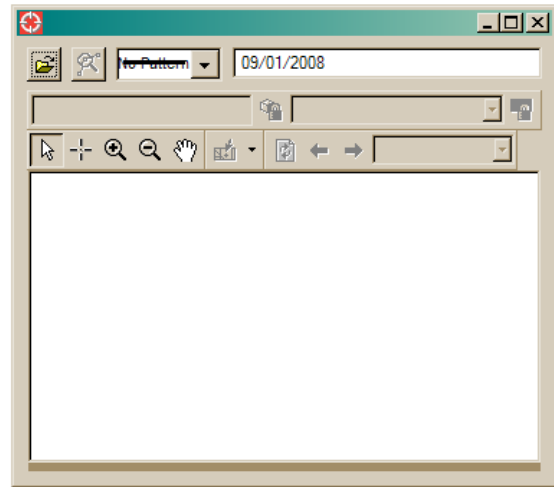
Build and Launch your design by pressing .

The 'dash_pattern' is 'unset' and the date_time is set to now.

Select the date_time item, then the dropdown arrow that appears to see the interactive date selection dialog.

To edit the 'time' modify the displayed text directly. This works for the date as well.

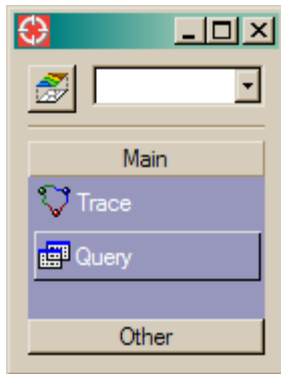
The 'view_zoom_to_trail' action will be available if you have trail set in the main GIS. The spatial context viewer is correctly embedded, resizes appropriately and is functional – try selecting something in the main GIS window.




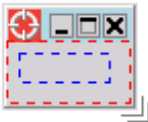

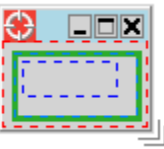

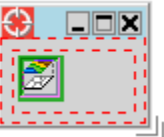

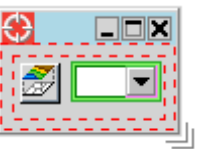

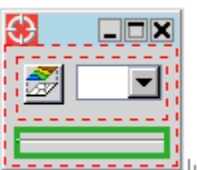
Though it is possible to embed whole plugins within your dialog, often the :build_gui() method written on the plugin class does not support this type of activation. In those cases you will either receive a traceback or a warning will be embedded into your activated dialog.


Fifth Dialog

This dialog shows recently developed widgets.

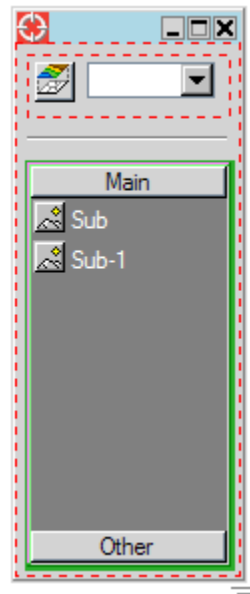


- Outlook bar
- panel separator
- language selection

<p>If you don't have a blank dialog design then start a new dialog by pressing  or using the shortcut Ctrl-n.</p>	
<p>Change to "Row Column Grid" mode  and insert one.</p>	
<p>Change to "Image Button" mode  and add one to the inserted RowCol. Select a 'Property' in the editor and right-click-'Set Image Source' to something interesting ... like ace_dialog.ico in the admin_ace_plugin.</p>	
<p>Change to "Text Choice" mode  and add one next to the button.</p>	
<p>Change to "Panel Separator" mode  and add one below the inserted RowCol.</p>	

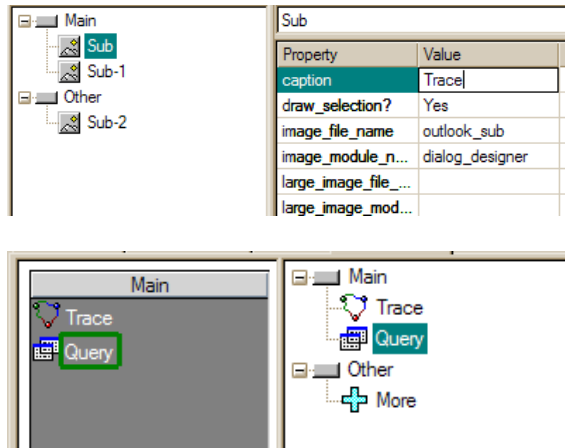
Change to “Outlook Bar” mode  and add one below the inserted panel separator.

Right Click on the “Outlook Bar” and select ‘Edit’.

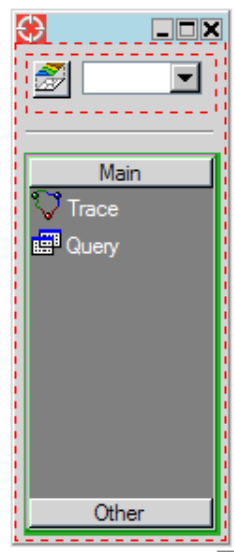


Select ‘Sub’ and change the ‘caption’ attribute in the editor to ‘Trace’. Right click in the editor and ‘Set image Source’ to ‘proximity_trace’ from the ‘ui_resources’ plugin.

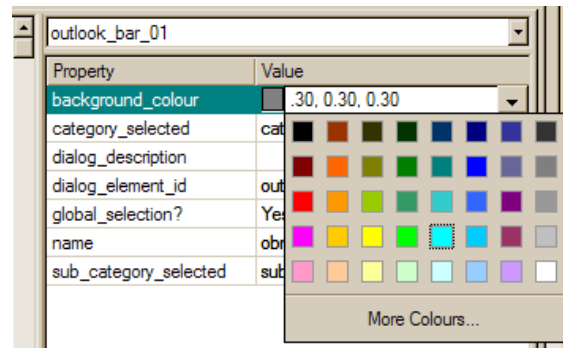
Select ‘Sub-2’ and change the ‘caption’ attribute in the editor to ‘Query’. Right click in the editor and ‘Set image Source’ to ‘query’ from the ‘ui_resources’ plugin.



Select the ‘GUI Layout’ tab. The ‘Outlook’ tab will disappear and the render will reflect the changes you made to the Outlookbar widget.



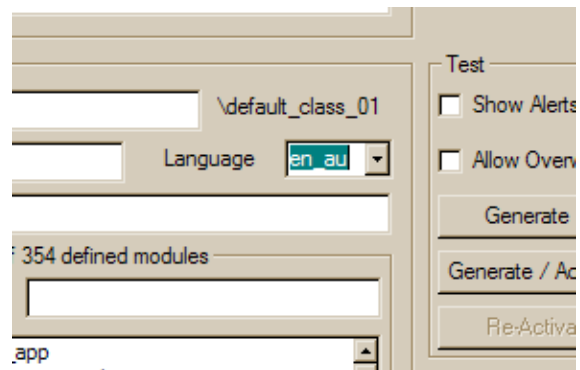
Change the background colour of the Outlook bar to 'Blue-Gray' (Second row, second-last column).




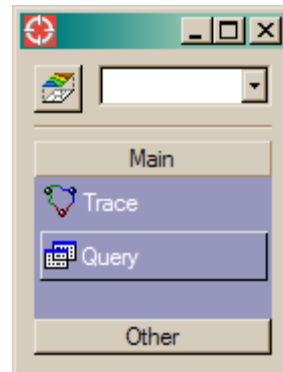
The GUI layout is done, just the language to set ... Select the 'Module' tab.

Directly edit the 'Language' combo-box to read en_au.

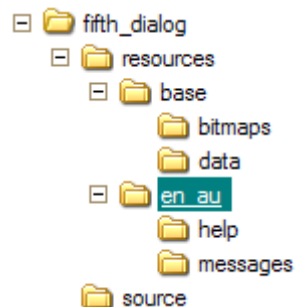
Change the "Base Class" to 'fifth_dialog' while you are there too.



Build and Launch your design by pressing .

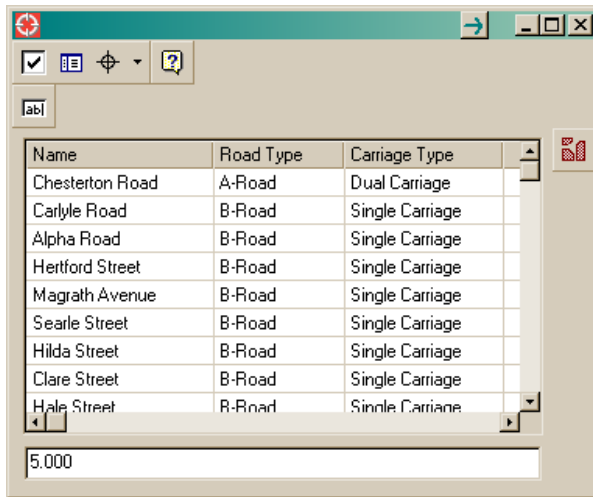


You can confirm that the message files were written to the 'en_au' messages directory by navigating to the generated code.




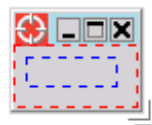
Sixth Dialog


This dialog shows recently developed widgets.



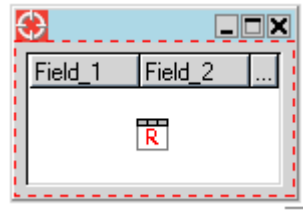
- Recordset GUI Component
- Managed numeric input
- user defined messages


If you don't have a blank dialog design then start a new dialog by pressing  or using the shortcut Ctrl-n.



Change to "Recordset GUI Component" mode  and insert one.

Change the ds_collection to "min_road" and the ds_name to "gis" (for Cambridge) or something similar for your system.



Change to "Number Input" mode  and insert one below the other widget.

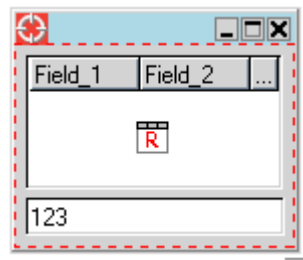
Set Attributes:

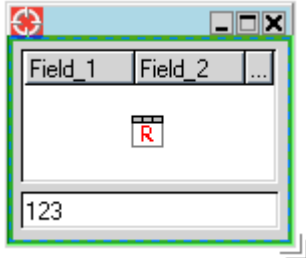

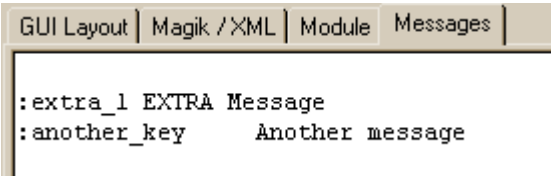

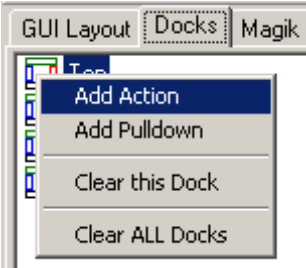
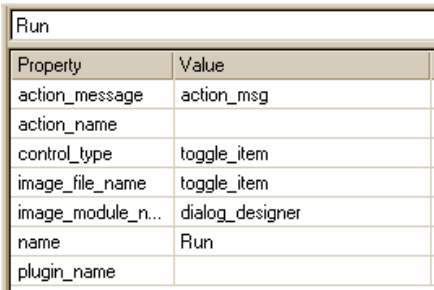
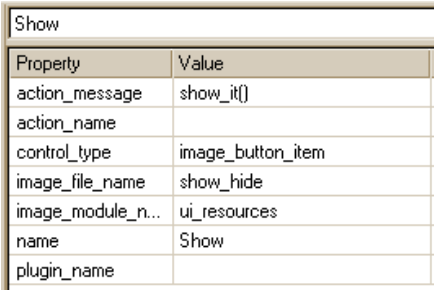
integer? → No

precision → 3

valid_interval → 0.1,9.4

value → 5.0



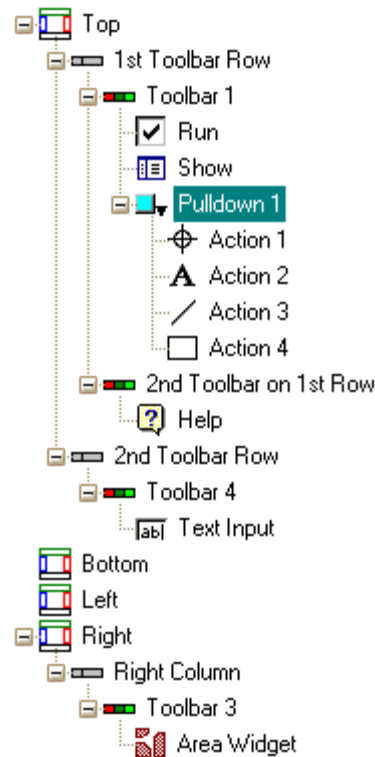
<p>Select the outer RowCol element and change the “row_resize_values” to 100,0</p>																	
<p>Change to the “Module” tab and change the base_class to “sixth_dialog”.</p>																	
<p>Change to the “Messages” tab and add a couple of messages ...</p> <p>:extra_1 EXTRA Message :another_key Another message</p>																	
<p>Press the ‘Use Docks’ toggle  to start adding dock actions.</p> <p>Select then Right-click the ‘Top’ Dock and ‘Add an Action’.</p>																	
<p>Select ‘Action 1’ and set: ‘name’ → Run ‘control_type’ → toggle_item</p>	 <table border="1"> <thead> <tr> <th>Property</th><th>Value</th></tr> </thead> <tbody> <tr> <td>action_message</td><td>action_msg</td></tr> <tr> <td>action_name</td><td></td></tr> <tr> <td>control_type</td><td>toggle_item</td></tr> <tr> <td>image_file_name</td><td>toggle_item</td></tr> <tr> <td>image_module_n...</td><td>dialog_designer</td></tr> <tr> <td>name</td><td>Run</td></tr> <tr> <td>plugin_name</td><td></td></tr> </tbody> </table>	Property	Value	action_message	action_msg	action_name		control_type	toggle_item	image_file_name	toggle_item	image_module_n...	dialog_designer	name	Run	plugin_name	
Property	Value																
action_message	action_msg																
action_name																	
control_type	toggle_item																
image_file_name	toggle_item																
image_module_n...	dialog_designer																
name	Run																
plugin_name																	
<p>Select ‘Toolbar 1’ and ‘Add Action’ Select the new action. Right click somewhere in the editor and ‘Set image source’ to ‘show_hide’ from the ‘ui_resources’ module.</p> <p>Set then name to ‘Show’ and action_message’ → show_it()</p>	 <table border="1"> <thead> <tr> <th>Property</th><th>Value</th></tr> </thead> <tbody> <tr> <td>action_message</td><td>show_it()</td></tr> <tr> <td>action_name</td><td></td></tr> <tr> <td>control_type</td><td>image_button_item</td></tr> <tr> <td>image_file_name</td><td>show_hide</td></tr> <tr> <td>image_module_n...</td><td>ui_resources</td></tr> <tr> <td>name</td><td>Show</td></tr> <tr> <td>plugin_name</td><td></td></tr> </tbody> </table>	Property	Value	action_message	show_it()	action_name		control_type	image_button_item	image_file_name	show_hide	image_module_n...	ui_resources	name	Show	plugin_name	
Property	Value																
action_message	show_it()																
action_name																	
control_type	image_button_item																
image_file_name	show_hide																
image_module_n...	ui_resources																
name	Show																
plugin_name																	

Select 'Toolbar 1' and 'Add PullDown' – this will add a pulldown button to the toolbar with an single action within.

Select 'Section 1' and add 'New Toolbar'.

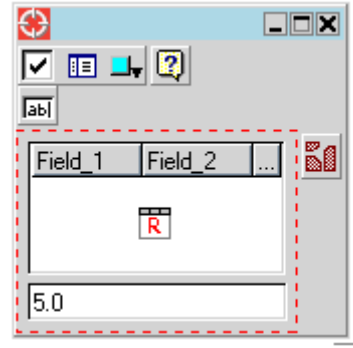
Select the 'Top' dock again and add a 'New Section'.


Modify the actions and add some other actions to the other docks ... usually you will only use the 'Top' dock but the others work fine though it is good GUI practice to only use button type actions on the Left and Right docks (eg. A text_item is too wide)



Returning to the GUI Layout tab the rendering will look (something) like ...

All types of toolbar widgets are shown using placeholder icons so larger widgets like choice_items and text_items will look smaller in the rendering than when the dialog is activated.

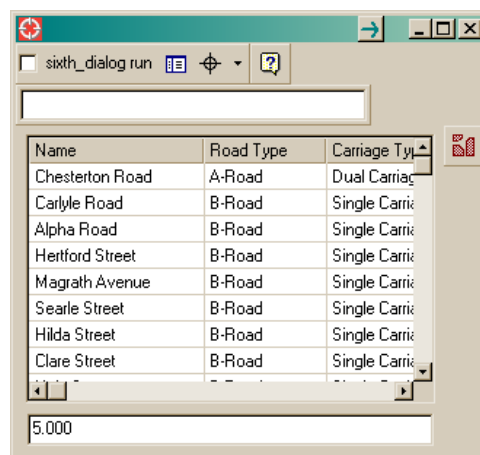


Build and Launch your design by pressing .

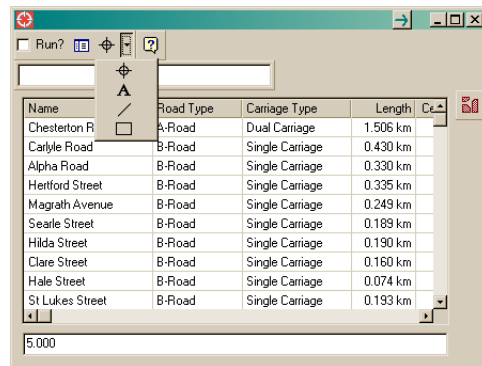
Right click the table header for grouping and visibility options. Drag whole columns by selecting and dragging to point between two other columns.

Resize the dialog to the size shown here.

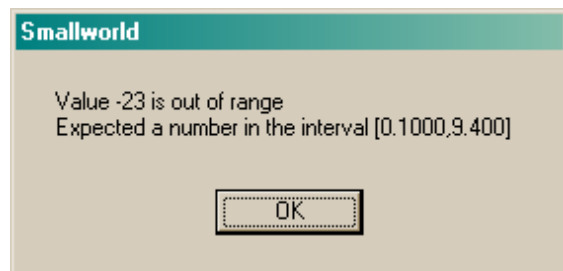
Note the missing message for the label of the toggle item. Add ":run Run?" in the 'Messages' tab and press  again ...



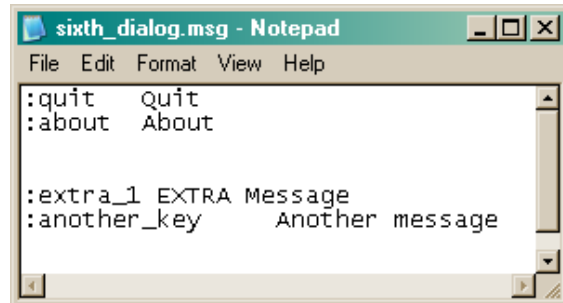
The pulldown contains the actions in the order they were set in the Dock tab ...



Entering an invalid value (eg. -23.3) in the numeric input box and then select a row in the table to see an alert.

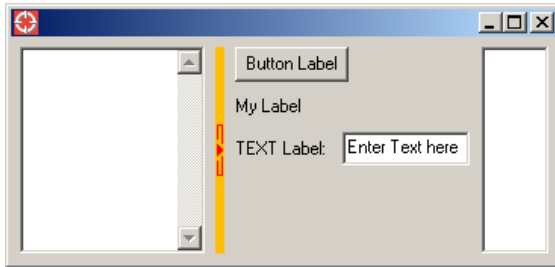


Goto the sixth_dialog.msg file to see that the added messages are there ...




Seventh Dialog

This dialog uses the Row Column Separator and demonstrates new messaging functionality.

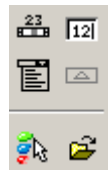


- Row Column Separator (SW 4.2+)
- Embedded Messages
- Message Search

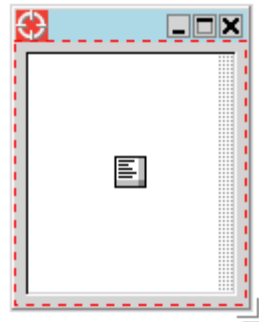
If you don't have a blank dialog design then start a new dialog by pressing  or using the shortcut Ctrl-n.



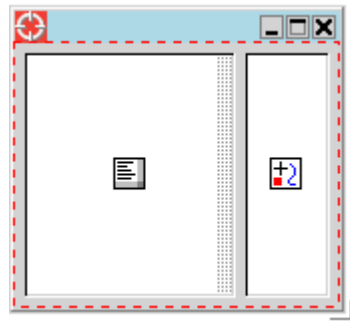
Note that the 'Row Column Separator' (RCS) widget is not available – it is only available when the current design contains a Row Column Grid with exactly one row or one column and that Row Column Grid has more than 1 element in it.

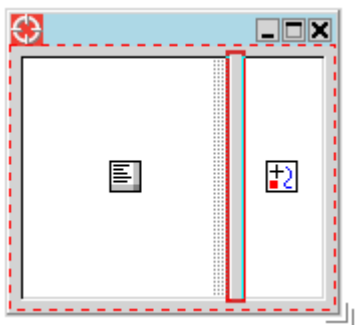
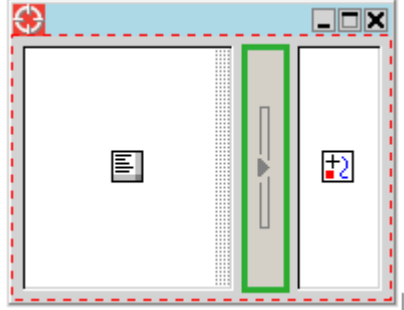
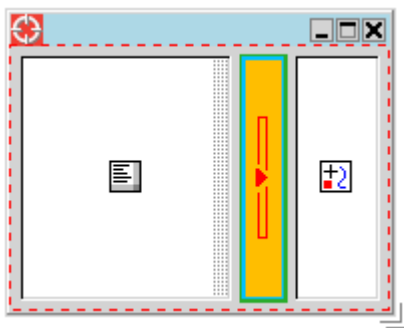

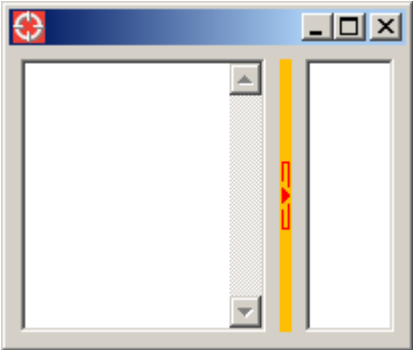
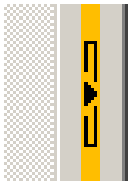


Insert a Text Window ... the RCS widget is still grey.

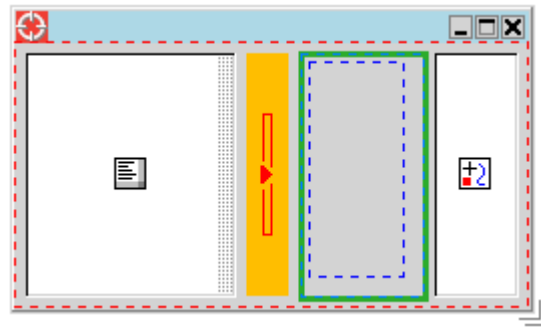


Insert a Text Window and a canvas in a row – the RCS widget is now available.



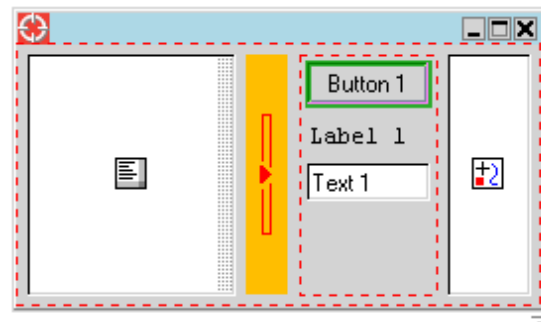
<p>Select the RCS widget and see how many places in the GUI you can insert it – exactly one ... between the text window and the canvas.</p>	
<p>The default minimize direction is 'right' for a horizontal row and 'down' for a vertical column.</p>	
<p>Change the colours ... here they are changed to:</p> <p>Separator – orange Default – red Highlight – black</p> <p>The 'highlight' color is only apparent when the GUI is activated.</p>	
<p>Build and Launch your design by pressing .</p>	
<p>Hover over the RCS to see the 'highlight' color, black in this case.</p>	

Change to “Row Column Grid” mode  and insert one next to the RCS.



Insert a Button, Label and Text Input in a column.

Select the Button.



In the editor you can see the usual properties.

Some of these properties are simple text: balloon_help_text, dialog_description and label. You can modify any or all of these and the hard-coded text will be inserted into generated code.

In addition, now you can specify the message key and message so that internationalizing your GUI will be easy.

button_item_01	
Property	Value
balloon_help_text	
col_alignment	left
dialog_description	
dialog_element_id	button_item_01
label	Button 1
min_height	
min_width	60
row_alignment	top
selector	button_item_01()

In balloon_help_text enter the text string “:button_tt Button Tooltip” and for label enter “:button_lab Button Label”.

The colon ‘:’ that starts the string tells the DD to treat this string as a message key and text.

balloon_help_text	:button_tt Button Tooltip
col_alignment	left
dialog_description	
dialog_element_id	button_item_01
label	:button_lab Button Label

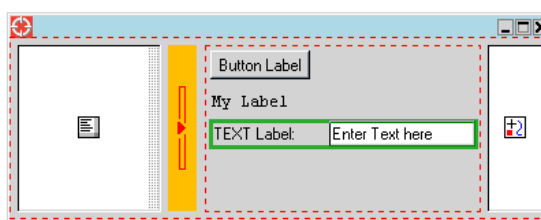
Similarly, you can add embedded messages for the ‘value’ of Label GUI element and value and label of the Text Item GUI element.

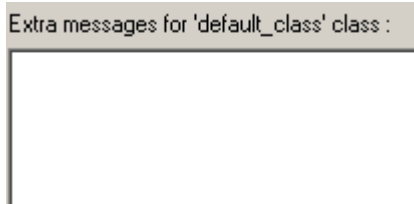

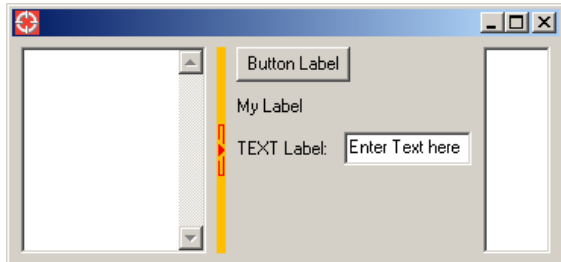
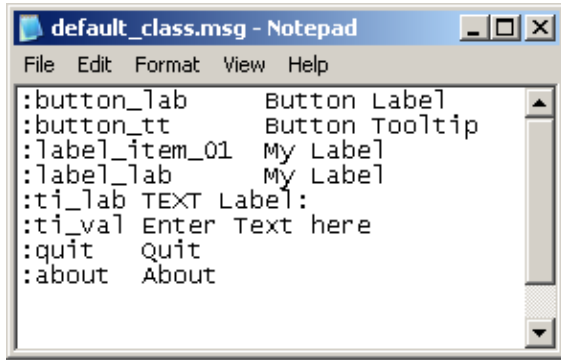

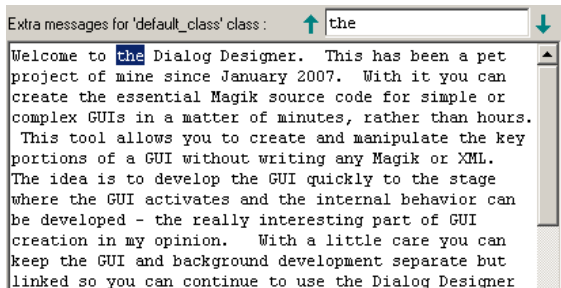
Enter:

Label.value = “:label_lab My Label”


Text_item.value = “:ti_value Enter Text here”

Text_item.label = “:ti_lab TEXT Label:”




<p>You won't see these messages in the 'Messages' tab, they will be written to the generated msg file directly.</p>	
<p>Build and Launch your design by pressing .</p> <p>Confirm the RCS works as expected.</p>	
<p>Find the generated MSG file for the base class ... for me it was “c:\temp\default_class\resources\en_us\default_class.msg”</p> <p>View it using any text editor to see how the embedded messages were written.</p> <p>Now, if you want to internationalize your GUI you can simply translate these messages into the target language and they will be used if that language precedes en_us in your !current_languages! list.</p>	
<p>Goto the Messages tab and enter some messages ... or just paste in some text – I just copied the introduction section of this document.</p> <p>After pasting text the pointer is probably at the bottom of the text window so click on the top line to set it nearer the beginning.</p> <p>Enter a word you can see in the search box and press 'Enter'. I entered 'the'. Subsequent pressing of the 'Next' arrow  will highlight the next matching text.</p> <p>The text window will scroll to ensure that you can see the current highlighted text.</p> <p>Pressing the 'Previous' arrow will work search upwards through the text.</p>	

Testing Dialogs

As you add elements to your design you can build the code module and launch an instance of the dialog by pressing: .

To test a dialog within the application you need to add the plugin to the application config.xml and gui.xml files. The dialog can be embedded into the application framework or launched using a button.

Below are code snippets for the config.xml and gui.xml for the dialog_design with a “Base Class” of “third_design”. View and edit the “Base Class” of your design on the “Module” tab, available when the  toggle is depressed. Once the XML is updated, start the application and look for your embedded dialog or activating action.

If you have specified consumer and producer datatypes for your design (Databus tab) these databus events will be logged to the Magik prompt.

For example, for the “Third Dialog’ designed earlier:

“third_dialog' data requested : post_render_sets” is written after each map refresh; and

“third_dialog' data consumed : map_trail sw:simple_vector:[1-2]” is written whenever the map_trail is updated.

Embedded Dialogs

Add into the <plugins> block of the application config.xml:

```
<plugin name="third_design" class_name="third_design"/>
```

Add into the application gui.xml (here it is placed in the tab_box where the object editor sits):

```
<tab_box role="viewport_mapping">
  <plugin plugin_name="third_dialog"/>
  ...
</tab_box>
```

Floating Dialogs

Add into the <plugins> block of the application config.xml:

```
<plugin name="third_design_plugin" class_name="third_design_plugin"/>
```

Add into the application gui.xml (here added to the top of the ‘Tools’ menubar):

```
<submenu name="tools" mnemonic_id="tools_m">
  <action name="third_design_plugin.activate_dialog"/>
  ...
</submenu>
```

Future features under consideration ...

- reverse engineer existing GUIs
 - o preliminary analysis of this feature has revealed how tricky it will be
- tool for generating an XML file for class properties (ie. GUI startup options)
- flagged_tree_item

... any suggestions? Email me: graham@ifactorconsulting.com

.. want to just talk it up? <http://groups.google.com/group/smallworld-gis-dialog-designer>

Glossary

CST	Core Spatial Technology (Smallworld)
GUI	Graphic User Interface
Magik	The coding language used by SW. This is the primary code for designing dialogs and their behavior in the SW environment.
SW	Smallworld (Core Spatial Technology)
SWAF	Smallworld Application Framework
XML	eXtensible Markup Language. These files (*.xml) contain configuration and GUI layout information for SW dialogs.