guiliani.de

Guiliani Streaming Editor (GSE)
User Manual

Product:	Guiliani Streaming Editor (GSE)
Release version:	2.2
Release date:	August 10, 2018

Table of contents

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

Table of figures

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

1 Disclaimer

TES Electronic Solutions GmbH
Hanauer Landstraße 328-330
60314 Frankfurt am Main
Germany

support@guiliani.de
www.guiliani.de

2 Introduction

We welcome you to Guiliani and the Guiliani Streaming Editor. Guiliani is your solution for graphical user interfaces (GUI) on embedded systems, which combines the comfort of a PC based development tool-chain with the benefits of a highly optimized software framework specifically designed for use on embedded hardware.

Profit from a mature software that has been used for years in tens of thousands of devices already and start using Guiliani now!

2.1 What Is Guiliani?

Guiliani is the abbreviation of **G**raphical **U**ser **I**nterface, **L**ight and **I**nnovative with **Ani**mations.

Guiliani is a C++ software framework enabling creation of visually appealing, platform independent GUI's for desktop, mobile and embedded systems.

Guiliani adopts the philosophy of "write once, compile & run" on target hardware. Run a once developed application natively on all supported target platforms. When using Guiliani the usual development workflow is simple:

1. Design the application on a PC
2. Target a set of embedded platforms for production release (for example a Linux-based platform using OpenGL-accelerated graphics, and a Windows Embedded Compact system based on pure software-rendering)
3. Compile and run your application in the desktop simulation or directly on the target.

Guiliani features very high quality visual appearance using sub-pixel rendering, including advanced functionality such as alpha blending and anti-aliasing, making optimum use of the underlying graphics API. This enables development of appealing GUIs for applications running on a wide range of embedded devices, spanning from cost-optimized to high-end hardware platforms.

2.2 What Is The Guiliani Streaming Editor (GSE)?

The streaming editor is a comprehensive tool for Artists, Interaction Designers and Developers to create and develop intuitive user interface with the ability to rapid prototype and deploy on embedded targets. The streaming editor was built using Guiliani and supports all features that are integrated into Guiliani such as high portability, support of different OS (e.g. Windows, Linux), support for multiple languages etc.

In addition, the editor brings some extra features for efficient GUI development like WYSIWYG capabilities, a run-time environment to simulate the developed user interface, a resource generator

for exporting the generated resources, the Guiliani C++ HMI Framework for application and event binding and many more.

The streaming editor is included in the SDK. This allows an instant development start with Guiliani and its GSE.

2.2.1 Main Features of Guiliani Streaming Editor

- Support of standard widgets, including property editor
- Resource Manager to manage Images, Fonts, Sounds and other Resources
- Multi-Language support (switchable during Runtime)
- Object hierarchy tree browser
- Workspace management
- Export Engine (including several export-formats)
- Guiliani runtime engine for multi-platform GUI simulation
- Font support through Freetype, Bitmap Fonts or Cleartype
- Interaction editor (e.g. switch dialogs, toggle object-states)
- Text layouting
- Drag & Drop of Widgets
- Expandability (e.g. custom, custom widgets)
- DataPool-Editor
- Built-in Requirements-Editor

2.2.2 The Guiliani Workflow

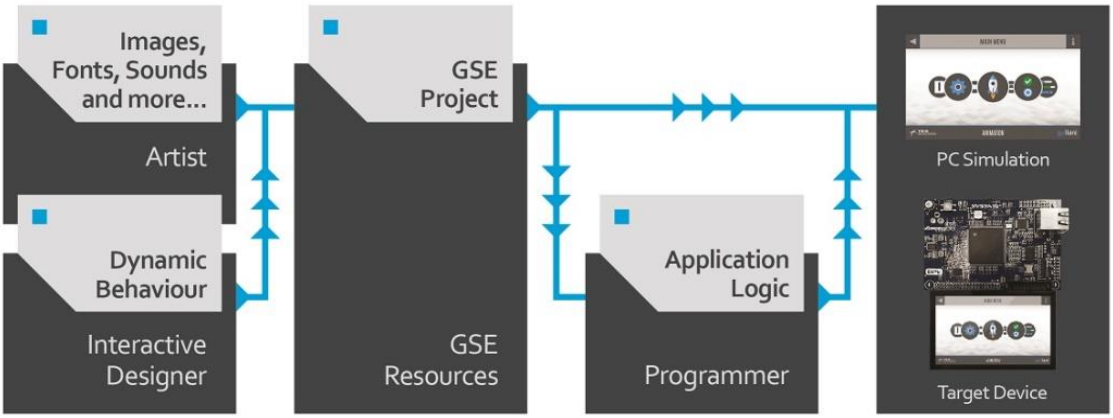


Fig 1 - The Guiliani Workflow

1. Artist creates HMI graphics (e.g. bitmap graphics for GUI background or buttons).
2. Interaction designer stitches together the event based application flow.
3. GSE's resource generator creates XML & Resources files (.h, .lng, .png, jpeg).
4. Programmer can visualize the GUI interaction using these resources and the Guiliani run-time executable.
5. The produced resources can be exported to an embedded target and visualized using the run-time port (StreamRuntime.exe).
6. Developer programs the application logic and binds data to the GUI
7. Resulting GUI and application on the target

2.3 An introduction to the principles of Guiliani

This chapter gives a brief introduction into the core concepts of Guiliani. If you are interested in learning more about them and the technical concepts involved in using them, please refer to the Guiliani documentation.

2.3.1 Streaming and the StreamRuntime

Streaming means the reconstruction of a GUI at runtime from a descriptive file. This file can be of arbitrary format - usually this will be an XML description (for human readability) or proprietary binary file (for optimized performance). Such a streaming file contains all required information to construct the GUI. As example, it stores:

- The objects used in the dialogs

- The Position and size of these objects
- Images used to visualize the GUI
- Texts to be displayed on Buttons, Text Fields etc.
- CommandObjects assigned to objects
- Attributes specific to a certain object (e.g. value range and preset of a Slider)
- And many more...

This information must be present within the streaming file, but the way in which it is represented is variable. Therefore, the file may have any format - and in fact, it is not limited to a physical file somewhere on local data storage, but could as well be read from a streaming source such as a network connection.

Another core concept of streaming is that each streamable object (i.e. each object within the GUI) is capable of reading and writing itself from/to a stream. In other words, the object itself knows which attributes it must read/write in order to completely (re)store its current state. This has the advantage that the overall complexity of the system is greatly reduced, since each object encapsulates its own streaming details within its own code.
Streaming is therefore the key concept for GSE itself. In fact, the editor stores its projects as a set of streaming files in XML (or binary) syntax. These files include all the information specified by the UI Designer during the process of building the GUI within GSE and are identical to the streaming files used on target.

The StreamRuntime is a basic Guiliani application, whose sole purpose is to initialize the Guiliani framework for the given target platform, and to read the files generated by GSE. Whether you hit "Run Simulation" within the editor or execute your final application on the target system, it will launch the StreamRuntime. This sets up the Guiliani framework and reads your UI designs from the streaming file(s). Of course, you are free to extend the plain standard StreamRuntime with application-code for your needs – its purpose is merely to serve as a slim entry point for getting your user interface up and running.

2.3.2 Resource Management

A GUI usually consists of a variety of resources such as images, fonts and sounds, which it uses to visualize itself. Efficient management of these resources is required to optimize the memory and CPU usage of your application. This is critical especially on embedded systems, where system resources are typically very limited.
Guiliani enables you to share resources among various objects within the GUI. Further, it allows you to tune the way in which they are loaded / unloaded so that it best matches your target setup. Usually this will be a trade-off between loading-times and amount of memory usage.
The core of this principle is that all resources will be referenced via abstract identifiers (Resource IDs) that serve as an additional level of indirection between concrete files on the file-system and their usage within the GUI.

2.3.3 Command Objects

Command objects represent a certain behavior within Guiliani; in reaction to user input. For instance clicking on a button, or moving a slider may execute a command object. You may see a command object as a predefined action (or chain of actions) that will be executed if a certain condition is fulfilled. The command object's Do() method describes this action.

A typical use case for command objects is the communication between the user interface and the underlying application logic. By deploying commands, you will gain three major advantages:

- Commands are thread safe. They will be serialized and executed in the GUI's thread context.
- They decouple GUI and application development. You can start developing the GUI even while the underlying application logic is not available - and vice versa. The typical work flow would be to start off with empty command-stubs whose Do() methods only include debug messages and attach the actual application-binding commands once they are available.
- Commands simplify automated testing. You can test the application-API through the respective command-objects without the GUI.

2.3.4 Behaviour Objects

The methods of CGUIBehaviour define Guiliani's standard event handling slots. These slots will be called by CGUIEventHandler automatically in response to specific user actions.

Thus if a control needs special handling for the associated events it can reimplement the corresponding slots with the desired functionality. When inheriting customized controls from Guiliani's standard set of widgets, it is advisable to call the base-class implementation of the overridden method. Failure to do so may break the base classes default behaviour.

2.3.5 Difference between Command and Behaviour

With behaviours you can override existing slots from Guiliani that will executed immediately.
With commands, you can add additionally functions to Guiliani that are executed after the normal process under conditions that will be defined by you.

2.3.6 DataPool

The "Datapool Module" is providing on-the-fly data updates of controls. Using the DataPool simplifies the task of linking objects within the GUI to external data sources. The DataPool will then automatically take care of updating observers of an entry within the DataPool whenever it

changes its value. Datapool-Connectors can connect to data from single variables, arrays, to databases and other places where data is stored.

Datapool is also usefull when synchronizing data between the application (e.g. sensor-data) and the GUI.

2.3.7 Nine Patch

A Nine Patch is a smart way of scaling up bitmaps without the usual quality and performance losses. The images are scaled by subdividing the source image into nine sections and blitting the border sections un-stretched, while only stretching the center section.

This is particularly useful for mostly rectangular images in use-cases such as a scaled button, a progress bar or a handle of a scrollbar.

The Nine Patch is defined by the width/height of its top, bottom, left and right borders. These are given in pixels and relative to the image's top/left, and bottom/right corners.

Typically, you will wish to set these values so that they encapsulate the areas of your source-image that shall remain in their original size, such as the edges or drop shadows of a button. The following image illustrates this concept.

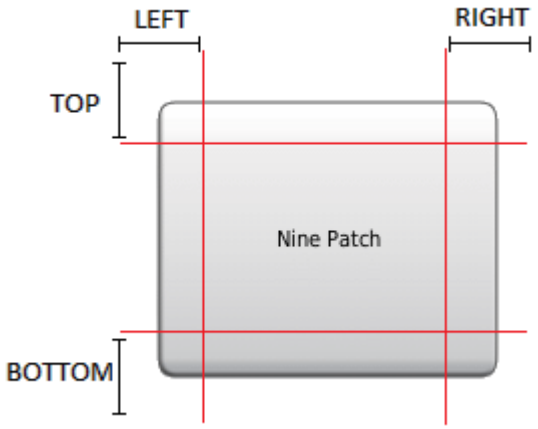


Fig 2 - Nine Patch

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

3 The Menu Bar

The menu bar at the top of the screen is your direct access to most of the editor's features, reaching from workspace management, via editing of resources related to the project, to simulation and export of the GUI.

Fig 3 - Menu Bar

A short-cut icon list is provided on the left side of the menu bar to speed-up the access to frequently used features "New Project", "Save Project", "Open Project", "Undo" and "Redo".

3.1 The File Menu

The "File" menu allows you to create new projects, save existing ones, add new dialogs, handle settings, manage requirements and simulate the current GUI.

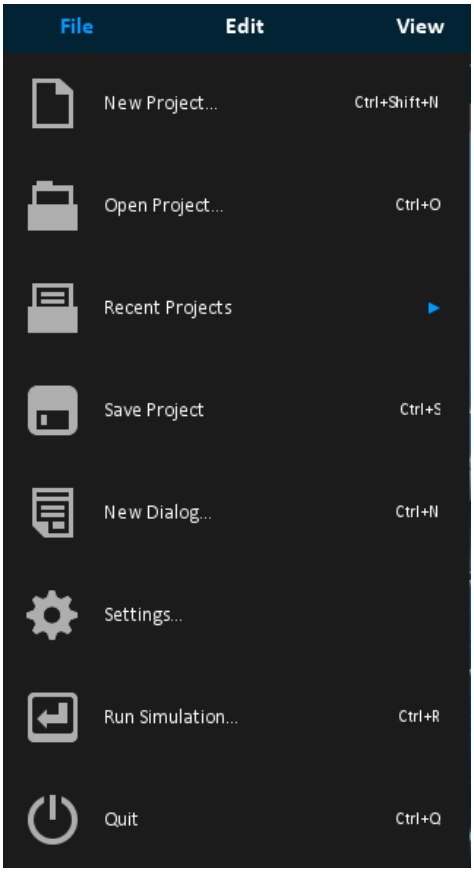


Fig 4 - File Menu

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

3.1.1 New Project

The “New Project” Menu is creating a new project. Enter a name for your new project and specify a folder on your disk where all project-relevant data will be stored. The HMI Editor will create a subfolder with the name of the project in the specified directory.

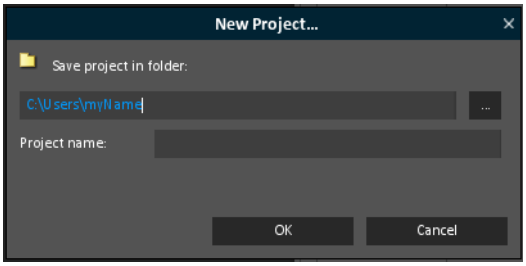


Fig 5 - New Project

3.1.2 Open Project

Use this option to open an already existing project. Select a Guiliani project file (“.gpr”) from disk. The corresponding project, including dialogs, resources, languages etc. will be loaded.

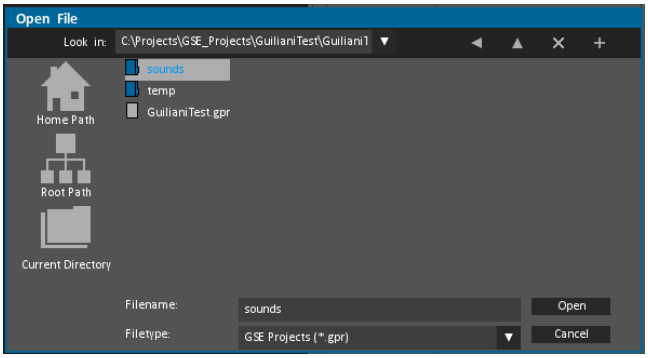


Fig 6 - Open Project

3.1.3 Recent Projects

Use this fold out menu to open a previously used project. Up to 10 projects will be listed here.

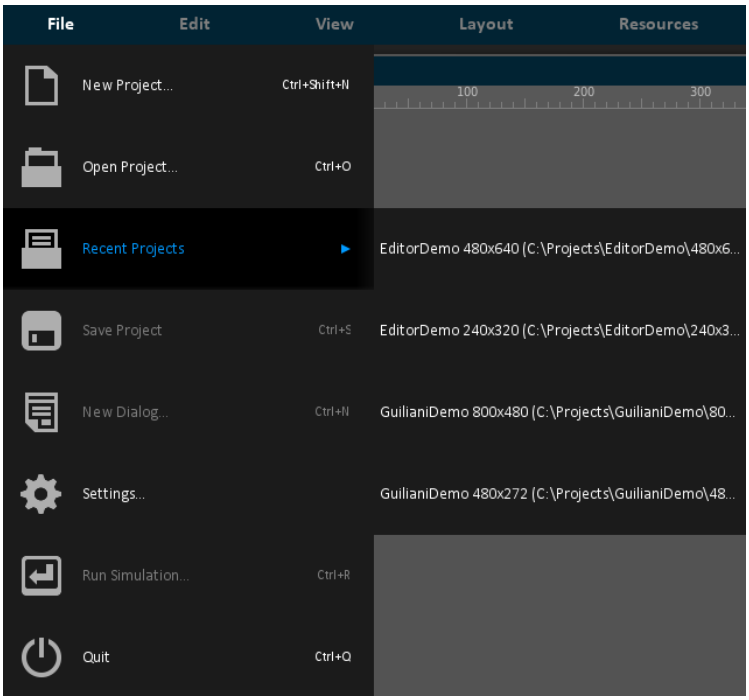


Fig 7 - Recent Projects

3.1.4 New Dialog

Use this option to add a new dialog to the currently opened project. You will be asked to supply a name for this dialog (this will also be used as the name of the Guiliani streaming file in which this dialog gets stored).

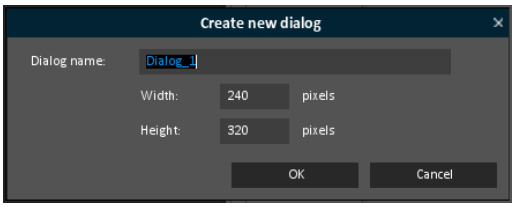


Fig 8 - New Dialog

You can also define the size of this dialog in pixels. This for example allows you to create non-full screen dialogs, which are used as Popup-windows.

3.1.5 Save Project

Use this option to save the current project to disk. The streaming editor will store all relevant information of the project, such as:

- DataPoolProperties.xml
- FontProperties.xml
- GeneralResourceProperties.xml
- ImageProperties.xml
- ObjectHandles.xml
- Properties.xml
- Requirements.xml
- SoundProperties.xml
- TextProperties.xml
- [ProjectName].gpr
- [ProjectName]_dialogs.xml

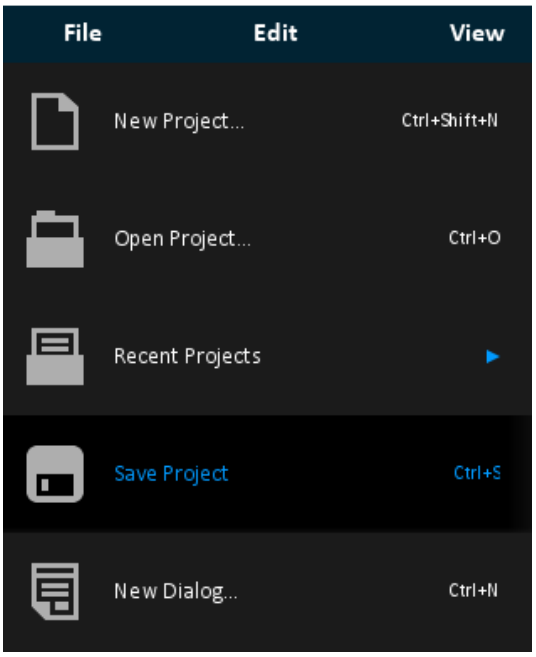


Fig 9 - Save Project

Additionally there are xml files for each of your dialogs and the corresponding folders, which contain all resources used by the project:

- *fonts, images, sounds, general resources*
- *temp* (temporary files created when you simulate your project)

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

3.1.6 Settings

You can set different values for the project you are working on and general GSE settings.

3.1.7 Global Settings

- Project directory: here you can define the directories where your projects will be created by default
- Help directory: location of the GSE help file
- GSE language file: select the language of the GSE (English or German)

Additionally you can specify whether to auto-open the console in case of errors (see Chapter 9 The console window). The console will give you more detailed information on occurring errors, by displaying any log-output that is received from GSE, Guiliani or its widgets.

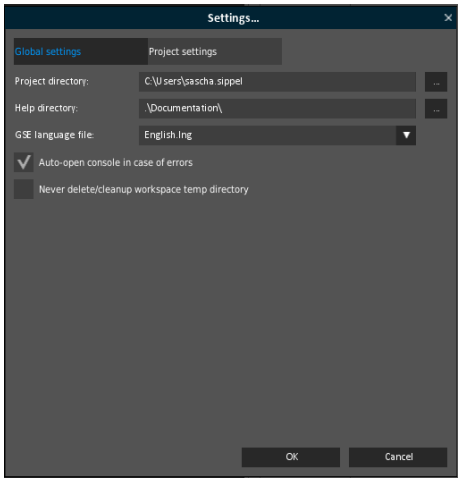


Fig 10 - Global Settings

3.1.8 Project Settings

The project settings define the path to dialogs, which can be imported to the current project.
For the import and export of language files, the language separator of the .csv file can be adjusted (typically ";").
The Overwrite option enables the overwriting of existing language files. If overwriting is enabled then existing languages will be overwritten by the imported languages with the same name.

When exporting to binary format endianness and alignment of the resulting bin-file can also be defined.

For special memory-usage-strategies different export-formats can be chosen.

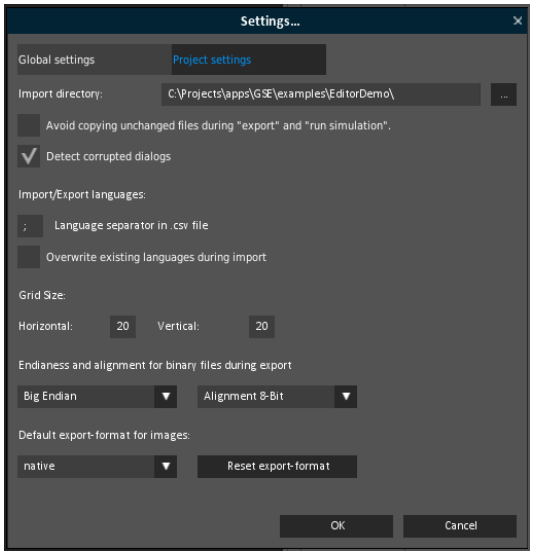


Fig 11 - Project Settings

By default all images are set to the "default"-setting which can be specified here. When using RLE or RAW formats images can be blit directly from Flash-memory to save RAM. The exported images are then converted into a special format which may take up more space in Flash-memory.

If during editing any images have been set to a different export-format than "default" a click on the button "Reset export-format" resets all images back to the default setting.

3.1.9 Requirements

The “Requirements” option enables you to track and assign project requirements. Using this dialog, you can specify requirements for the current project and assign them to dialogs and objects or mark dependencies to other requirements. Additionally, you can export the list of requirements into a “.csv” file for use in dedicated requirements tools.

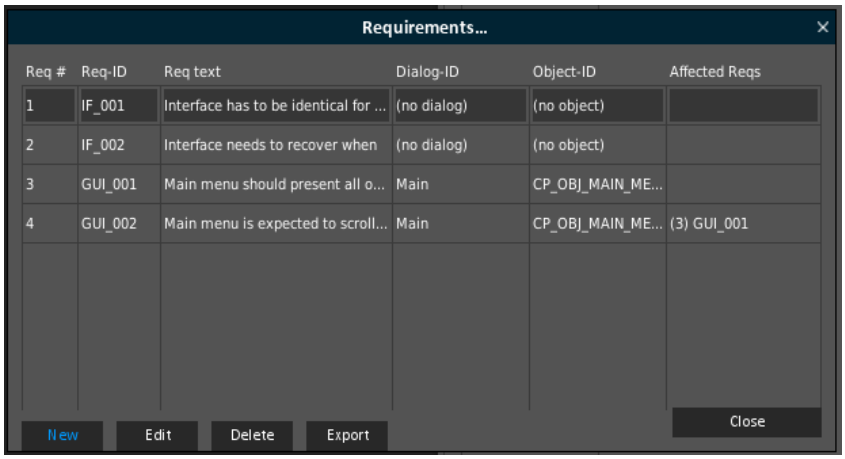


Fig 12 - Requirements

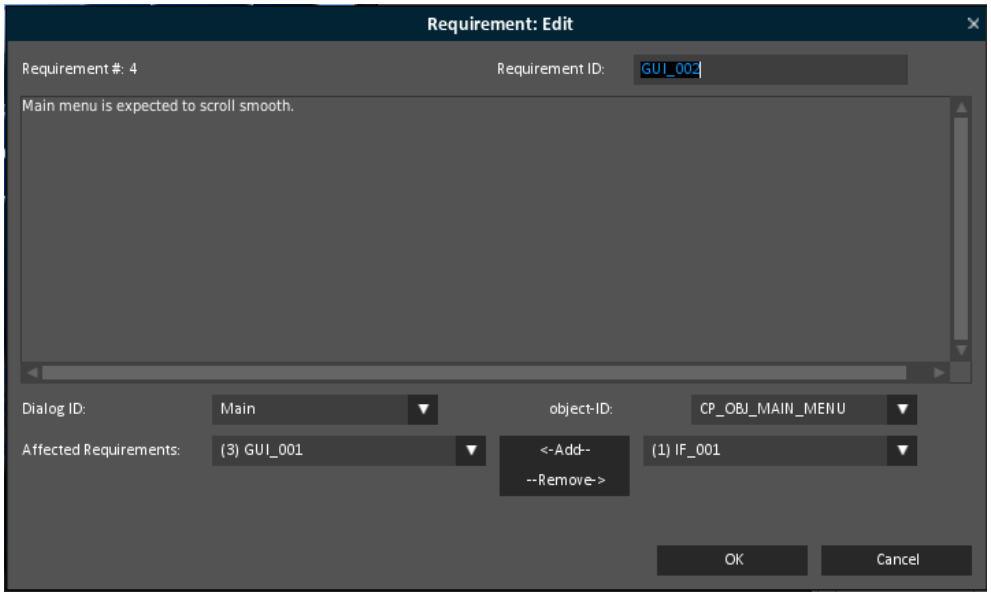


Fig 13 - Edit Requirements

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

3.1.10 Run Simulation

Use this option to run a simulation of the current GUI. This will perform a temporary export of your GUI and immediately execute it in a stand-alone Guiliani application (called StreamRuntime).

This application runs independently of GSE itself and offers a lifelike preview of your application, very similar to what it will look like on the final target platform.

You are free to choose the resolution for the simulation window and to select the resource sets that will initially be used for the simulation.
You may also choose which dialog is loaded initially within the simulation.

By default, the simulation will search for the GUI's resources right next to the runtime executable. If your setup requires the resources to reside in a different location, you may supply a path to this resource directory.

Fig 14 - Run Simulation

In case your GUI does not have a fully opaque image as a background, you may activate a definable background color, which Guiliani will use as a background for the simulation.

There are input fields to provide scripts to be run if desired. They will hook in at three different important points of the temporary export for the simulation; before the export, after the export but before it is packed and after packing.

For more details, see export scripts.

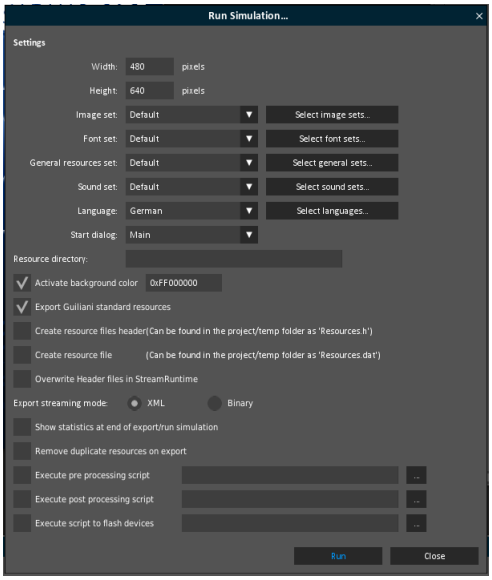


Fig 15 - Run Simulation window

3.1.11 Quit

This quits the HMI Editor. If there are any unsaved changes in your GUI, you will be asked to save your project.

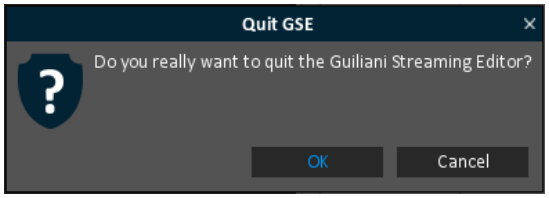


Fig 16 - Quit

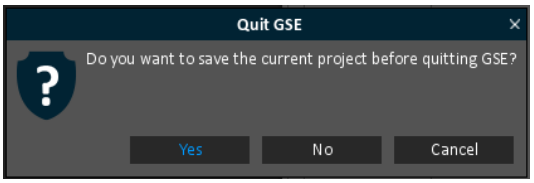


Fig 17 - Quit And Save

3.2 The Edit Menu

The edit menu offers you cut and copy & paste functionality. You may select one or several objects (e.g. by clicking them with the "CTRL" key pressed) and cut or copy & paste them by selecting the corresponding menu option. Note that you can also use this to copy objects between different dialogs.

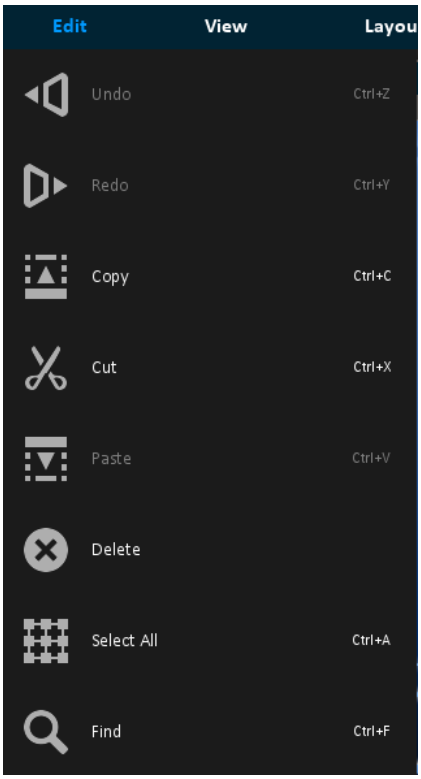


Fig 18 - Edit Menu

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

3.2.1 Find

Selecting find opens the “Search in dialog” dialog, where you can search your project for a variety of criteria. E.g., you can search for specific text strings, for uses of a specific resource-identifier, or for an object with a given Object ID.

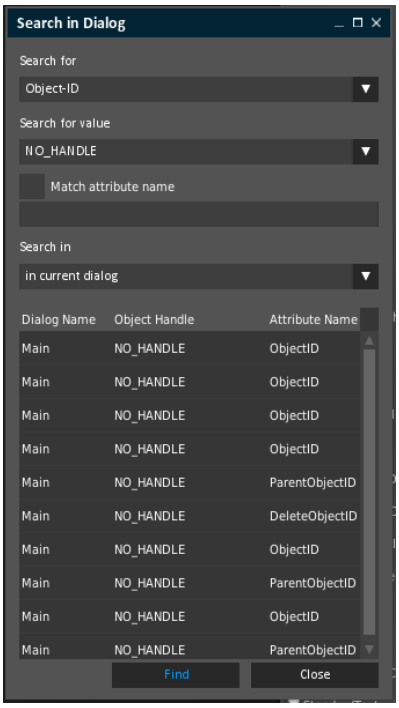


Fig 19 - Find

3.3 The View Menu

With “Toggle Grid” you can insert a grid or remove the grid if chosen before. The grid serves as a visual aid and will automatically snap any moved objects inside the preview window to positions that are multiples of the grid size. The grid size can be changed under File → Settings → Projet settings.

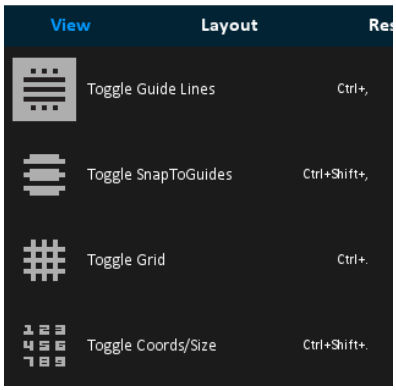


Fig 20 - View Menu

3.4 The Layout Menu

The layout menu supports you in arranging GUI objects within a dialog.
“Arrange objects” lets you modify the hierarchy of objects in the GUI-tree, by specifying the order in which they are drawn. You may move one or several objects to the front or background and will immediately see the visual effect in the dialog window and the object hierarchy.

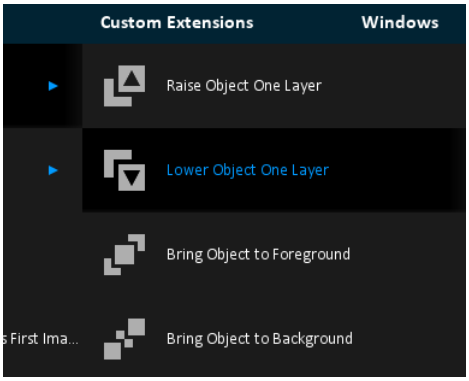


Fig 21 - Layout Menu / Arrange Objects

"Align objects" simplifies the arrangement of several objects in the GUI, by aligning them in one of the pre-defined manners. The first selected object always defines the reference point of the alignment.

Another provided functionality is the even distribution of several objects. The objects are arranged between the two outmost objects, either horizontally or vertically.

Unify object dimensions will adapt the dimensions of the selected objects to the dimensions of the first selected one.

Resize object to dimensions of its first image will resize the object to the original dimensions of the first image in its list of attributes. This is useful to prevent objects from being accidently stretched.

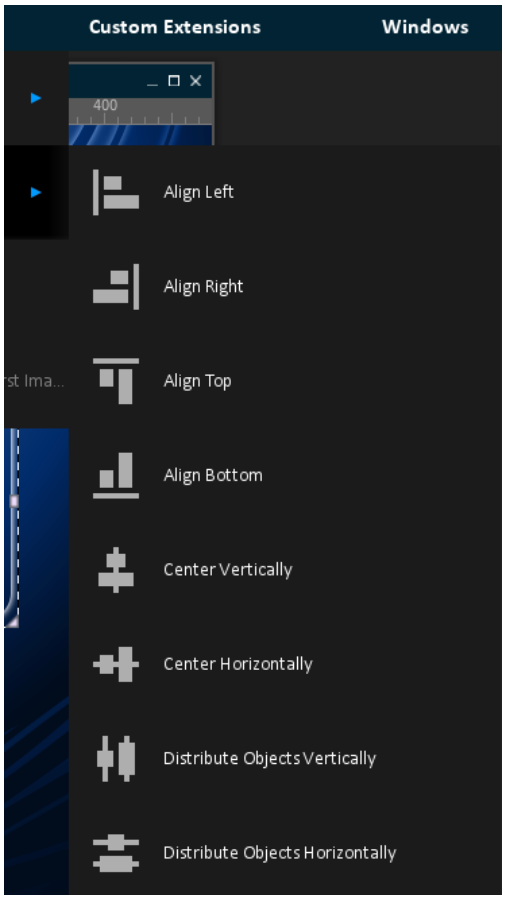
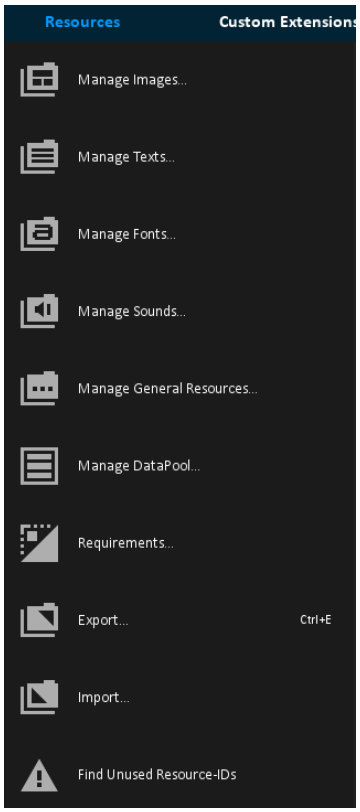


Fig 22 - Layout Menu / Align Objects

3.5 The Resources Menu



Use the resources menu to manage all resources of your GUI, including images, internationalized text, fonts and sounds. Each of those resource types is handled in a dedicated sub-screen.
In addition, you can export your project or import dialogs, too.

NOTE: If you want to search for a certain resource, you can do it by using “Find unused resource-Ids”.

3.5.1 Manage Images

Guiliani is referencing images by an ID, which serves as a symbolic name for the given image. This has two major advantages:

1. You can simply change the actual image behind an ID, thus greatly simplifying skinning for a GUI.
2. Resources can easily be shared among various objects in the GUI and will still only occupy memory once.

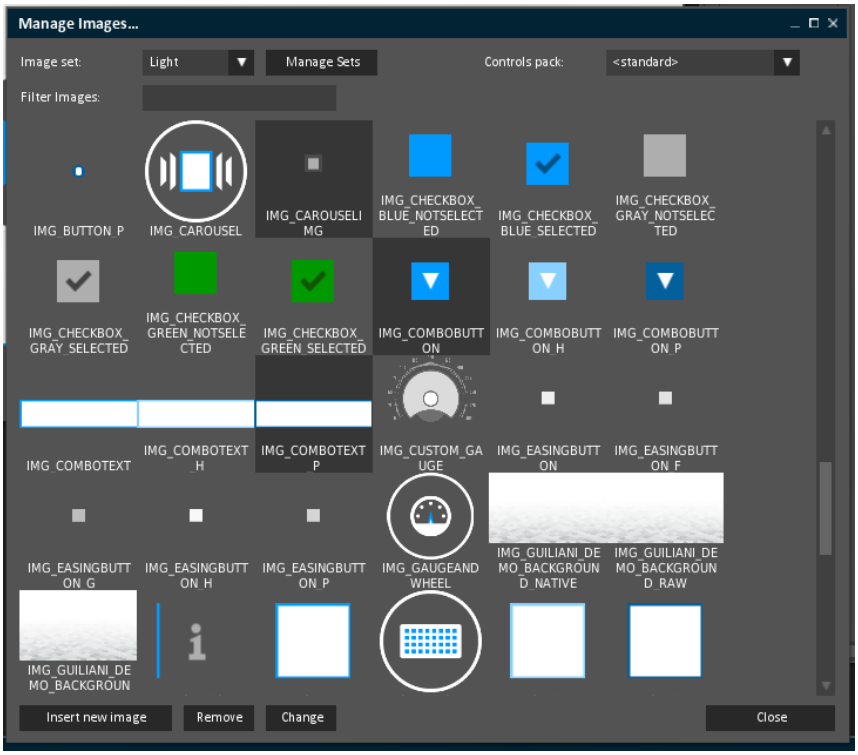


Fig 24 - Manage Images

The “Manage images” screen does allow you to import images of various formats (PNG, JPG and BMP) into the Guiliani streaming editor and to associate them with an ID.

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

You can do so by clicking the button “insert new image”, browsing your disk for the desired image and assigning an ID to it.

Resource-Sets:
The drop down menu “Image set” lets you switch between different image sets. You can click on the button “Manage Sets” (chapter 3.5.6) to create, delete or rename a set.
All sets share the same IDs, but they may map different images onto them.

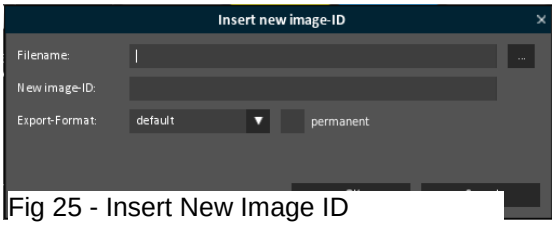


Fig 25 - Insert New Image ID

Permanent Resources:
In the “Insert new image-ID” dialog the “permanent” checkbox allows you to fine-tune the runtime behavior of your GUI. By default, resources in Guiliani will be loaded when required, and unloaded again once they are not used anymore. This reduces memory-consumption but may result in additional loading times for the respective bitmaps. Also Memory-Fragmentation might occur.
Marking an image as “permanent” prevents it from being automatically unloaded again. This will speed up your application (in particular screen switches) but at the cost of increased memory usage.

Export-Formats:
Also the export-format can be set to “default”, “native”, “RLE” or “RAW” to use different memory-consumption-strategies (see Image-Export-Formats for further details)
Once an image is shown within the “Manage images” screen, it can be assigned to an object within the current GUI.

NOTE: The default Guiliani resources cannot be changed or removed.

3.5.2 Manage Texts

While it is possible to use string literals within Guiliani, it is desirable to use Text IDs instead. This enables you an easy migration from a single language to a multi-language GUI by simply translating the strings without recompilation of your application.

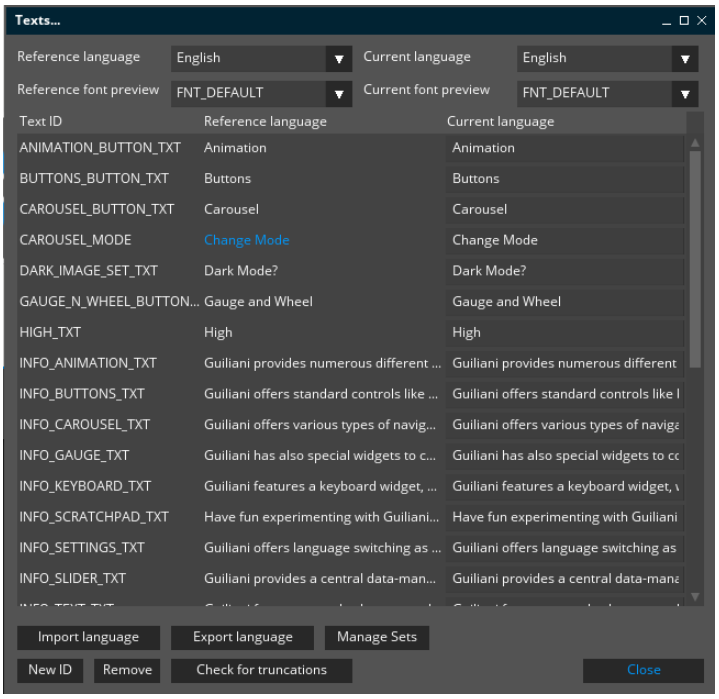


Fig 26 - Manage Texts

In the “Manage Texts…” window, you will find dropdown-boxes to select a reference-language and the currently used language used in the GUI. Additionally you can select the related fonts to see if there will be problems with special characters.

NOTE: You can click onto the button “Manage Sets” (chapter 3.5.6) to switch between different text sets. A new dialog will be opened where you can create, delete or rename a set. All sets share the same IDs, but they may map different texts onto them.

NOTE: The dropdown boxes “Current font preview” and “Reference font preview” only contain fonts that are currently included in the project.

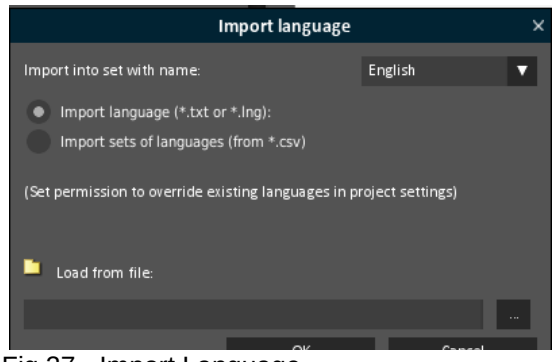


Fig 27 - Import Language

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

Enter your Text in the text input field of the section “current language” and press Enter. Add Text IDs by pressing the “new ID” button. To delete, create or rename a text set, click the “Manage Sets” button.

Text-sets associate Strings in a given language with Text IDs. For example, the Text ID “TXT_EQUALIZER” in the Text Set ‘German’ associates to “Klangeinstellung” while in the Text-set ‘English’ it maps to “Equalizer”. Additionally you can export and import one or all of your language sets.

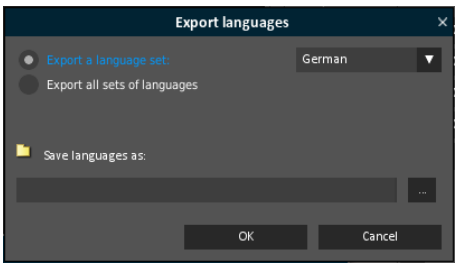


Fig 28 - Export Language

NOTE: Importing already existing languages only works when “Overwrite existiong languages during import” under “File → Settings → Project settings” is set. Otherwise data can only be imported for new language names either by writing a new language name in the “ Import into set with name” drop down menu or by importing a .csv-file with a non-existing language name column. When exporting languages, output files are overwritten without warning.

With the button “search for truncations”, you can easily check all of your texts if they will fit correctly into the objects. But only texts that are not entered directly in the object can be checked, i.e.those entered using “Manage Texts”.

NOTE: This will work for saved dialogs, only, so either save your project before searching for truncations or just click onto the “Yes” button if you will be asked for saving your project before searching for truncations.

3.5.3 Manage Fonts

Fonts in Guiliani are referenced via “Font IDs”. Each font ID represents one specific font, with a defined font-face and size.
For example, the font ID “FNT_ARIAL_HEADLINE” could have the font size 20 and the font face Arial.

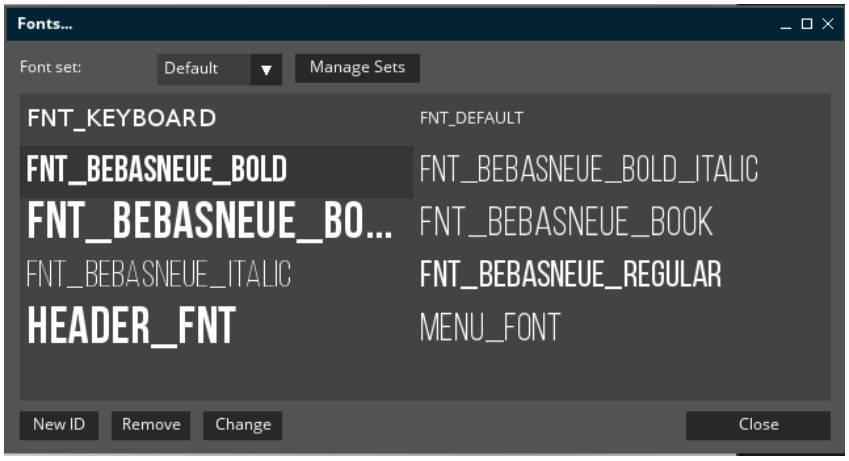


Fig 29 - Manage Fonts

By assigning this font ID to every headline of your GUI, the GUI design can easily be changed by switching the Font set of the project. The active “Font set” is set visible in your GUI.

NOTE: You can click onto the button “Manage Sets” (chapter 3.5.6) to manage different font sets. A new dialog will be opened where you can create, delete or rename a set. All sets share the same IDs, but they may map different fonts onto them.

Add new Font IDs by pressing the “new ID” button. The “insert new Font ID” window is opened and you can enter or browse for the font file. The Font ID will be set automatically. Enter the font size (in pixels).

Fig 30 - Insert New Font ID

If you would like to change the size or source file, press the “change” button.

For deleting a selected font, click onto the “remove” button. (The default Guiliani resource fonts cannot be deleted)

NOTE: The size of a text is set in this font menu. To have a small and a large text you need to create two fonts-resources.

3.5.4 Manage Sounds

Sound IDs are managed in the manage sounds window. Here you will find the play button for preview listening, a “new ID” button, a remove and a change button.
If you press the “new ID” button, the “insert new ID” window will open, which does have a play button, too.
Here you browse the sound file, name its ID and choose between permanent and not permanent.

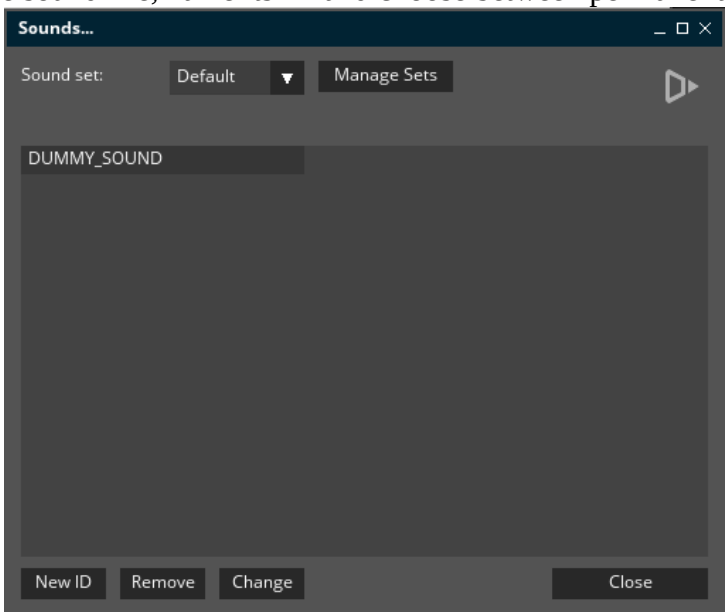


Fig 31 - Manage Sounds

Changing the sound scheme of your GUI is done by changing the “Sound set” in the drop down menu. In the “Manage Sets” dialog, you can create, delete or rename the sets. All sets share the same IDs, but they may map different sounds onto them.

To delete a sound, select it and click onto the “remove” button.

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

3.5.5 Manage General Resources

Within the “Manage General Resources” dialog you can manage your own independent resource IDs. Like images, fonts and sounds, they are assigned to a resource ID, but do not contain specific content. You need to provide your own controls to use these resources, but you are free to embed 3D models or videos into the GSE workflow just like the fixed type resources. They will be exported and imported just like the fixed type resources.

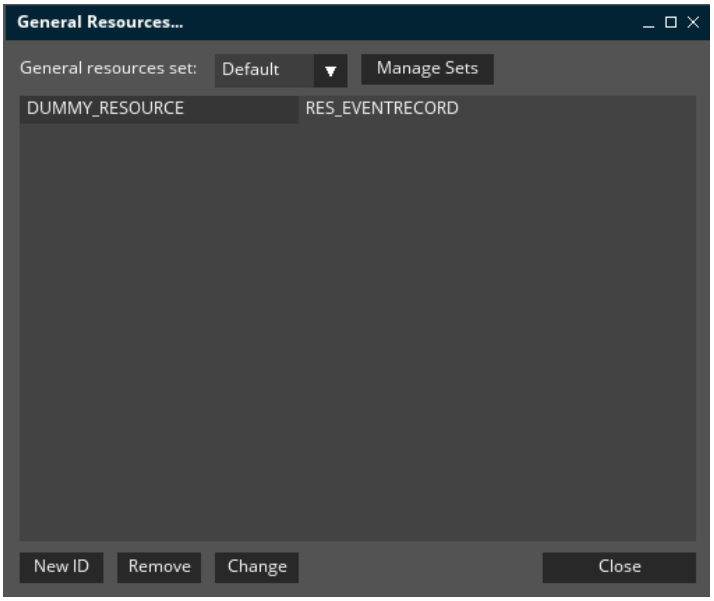


Fig 32 - Manage General Resources

The advantages of using 'general resources' for tracking your resource files are:

- Keep track of your project's resource files in distributed development environments.
- Automatic exports of other resources to the project streaming target directory.
- Application wide access to 'general resource' files using the Guiliani integrated resource manager, i.e. by a unique resource ID.
- Possible encapsulation of other resources to the Guiliani resource file, which contains all resource files of the workspace project after export.

For a new resource ID, or to change or remove an existing one, just click the corresponding button.

3.5.6 Manage Sets

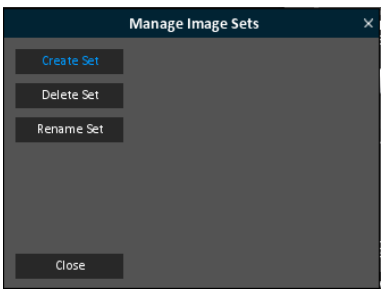


Fig 33 - Manage Sets

The manage sets window allows you, as its own name says, to manage the different sets your Guiliani application might use (i.e. Image, Text, Font and Other resources)

The main view has four buttons, one for each of the options

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

you have for managing a set, and a close button.
The possible options in the managing of a set are: Create set, Delete set and Rename Set.

When you click "**Create Set**", a list of elements is made visible. Those are, a text field for introducing the name of a new set, a check button. If marked, it will make visible a list of possible sets to copy the information from into the newly created set. At the bottom, a button named "Create" is visible; it will create the new set once you have introduced a valid name.

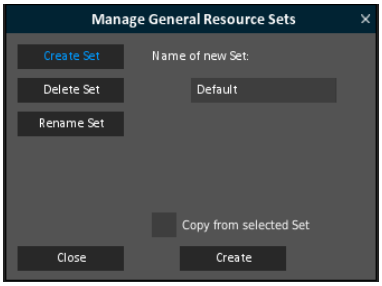


Fig 34 - Manage XXX Sets - Create

The next option of manage sets is to delete a set. That process begins by clicking the option "**Delete Set**". Then, a list of existing sets will be visible, for you to choose which of them to delete. This deletion is executed when you click the button "Delete".

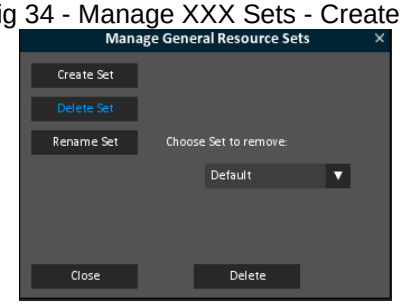


Fig 35 - Manage XXX Sets - Delete

If you click "**Rename Set**", the list of visible elements shown contains a text field for writing the new name of the set, a list of sets from where you have to choose which set you're going to rename, and a button named "Rename", that will rename the chosen set after a valid name has been introduced.

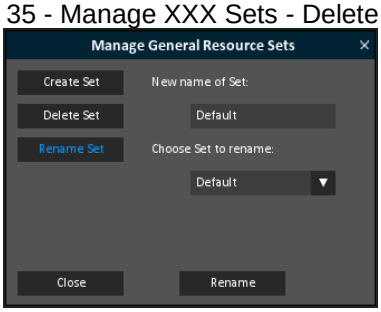


Fig 36 - Manage XXX Sets - Rename

3.5.7 Export

Fig 37 - Export

An export will copy all files related to your project (Images, fonts, language-files, XML files etc.) into a directory of your choice, from where you can copy them onto your target device.

The width and height fields will be forwarded to your target system's StreamRuntime, where they will typically be used as the application's initial size.

The various set names define which resources will be loaded as defaults when starting the application. Note that you can use the "select XYZ sets…" buttons to exclude certain sets from the export, if you wish.

The resource directory is an optional path, which will be used as a prefix when the StreamRuntime searches for resources on the target system. This is useful if the resources will not reside in their standard location right next to the executable.

You may wish to activate a default background color for your application, in case you do not have a default full-screen background image in each of your screens.

If you do not use any of Guiliani's standard resources, because you have your own designs, it is recommended to uncheck the "Export Guiliani Standard resources" checkbox, so that they will not be exported and take up more memory.

The Create resource header / Create resource file checkboxes offer you the possibility to automatically create a Resource File from all resources, which you chose to export. This is particularly useful for systems without a file-system, where you will compile all resources right into the StreamRuntime executable.

When having changed Object-ids which are referenced within the StreamRuntime-application, the option "overwrite Headerfiles" can be used to overwrite the "UserXXXResource.h"-files inside the StreamRuntime to update all IDs properly on export/run simulation.

The "Streaming Mode" radio buttons let you specify whether you wish to export XML or binary streaming files. When using binary-files endianness and alignment are set in the "Settings"-dialog.

When activating the "Show statistics" a small dialog is shown after export where all resources and their total sizes are shown. When using different export-formats the values of input and output can differ and thus the overall percentage of increase/decrease is shown.

The “remove duplicate resources” option can be used to remove files which are identical in content but different in name and would thus consume more memory as really needed. If files with identical content are found during export the first file is taken as the original and all following copies are replaced by references to this original, i.e. only one file is actually exported and the other Resource-IDs just point to the same file.
This option does not affect General Resources and operates only within the same resource-set, so different image-skins may have different duplicates.

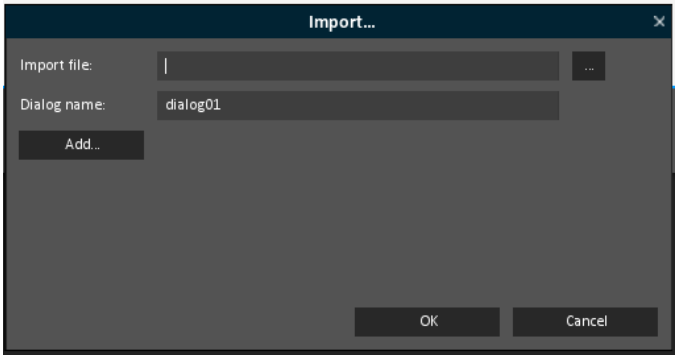
A pratical example would be to use a set of dummy-images during GUI-design each with a different name and ID, but same content. And now part by part the images are replaced with their actual correct image-file. Till this point the export with the activated option will help saving memory on the target-device during evaluation. When the GUI is finished all files are now replaced and the option will not filter out any duplicates and could be de-activated.

Export Scripts
There are input fields to provide scripts to be run if desired. Depending on the OS running the GSE, you will need shell scripts or batch files.
To understand the scripting capabilities it is necessary to know that the export can be split into two phases. First, the export copies all needed data. In a second step, it works on the gathered data like packing it.
This leads to three starting points for the scripts: Before the export starts, before it begins to pack and after it is finished.

There are several possibilities that arise from using one or more scripts. In a pre processing script you might want to update some repositories to ensure the usage of the newest resources. The intermediate script can be used for some expert cases, like converting graphics to platform-specific formats, before the export packs. Finally, a script can be used to copy the export to a special destination, store a release or flash a target, immediately after exporting.
For more convenience, the checkboxes enable/disable script execution.

3.5.8 Import

The import wizard is used to import Guiliani dialogs to the current workspace. This allows you to import GUIs that were created from manually implemented Guiliani applications, or that have been generated through conversion from other file formats.

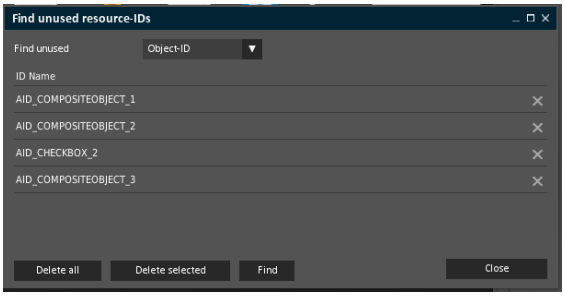


To import one or more dialogs out of another GSE project, pass the dialog .xml files to the "Import" dialog. After clicking the "OK"" button the "prepare resources" dialog appears. If all dialogs to import are valid, the "prepare resources"" dialog fills itself. Afterwards the import process starts.
In case of errors during the import, all resources, which can be loaded, will appear. For all other resources the user will be informed about the error.

Fig 38 - Import

3.5.9 Find Unused Resource-IDs

A simple but very useful tool to identify unused resources, in other words resources that are not used in any dialog of the workspace project. Use "Find unused resource ids" to keep the project clean of resource residues.
First select a resource type to inspect, using the combo box in the first line, afterwards press 'Find'.
The table presents unused resources, so that they are ready for deletion by clicking the red X next to the entries.



NOTE: Depending on the project size, the search procedure can take up to several minutes.

Fig 39 - Find Unused Resource IDs

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

3.5.10 Manage DataPool

This dialog lets you control the data pool of Guiliani. The Guiliani data pool is an object oriented data pool. Therefore you need not to select the type of data that is synchronized. See also the data pool demos that accompany the SDK.

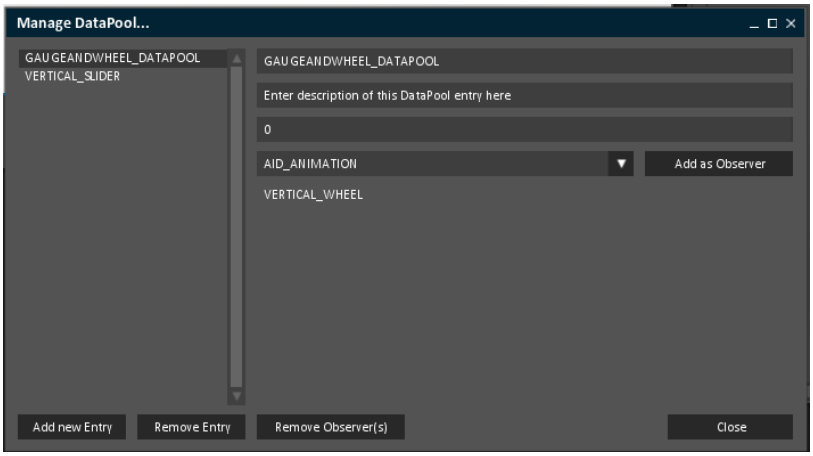


Fig 40 - Manage DataPool

On the left side is a list of all present data pool entries. Add an entry by typing a name in the right field and press the below "Add new Entry" button. Alternatively, remove entries by selecting an entry and clicking the below "Remove entry" button. On the right, it shows details to the selected data pool entry. Rename it instantly by typing a new name and selecting another field.

When a data pool entry is selected, its observers are shown in the right box. Remove an observer by selecting it and clicking on "Remove Observer(s)". Vice versa the drop box allows selecting a new observer and adds it via the "Add as Observer" button.

3.6 Custom Extension

The custom extension wizard generates class stubs for customized controls/ commands/ behaviours and registers them as custom extensions to the GSE and StreamRuntime projects.
A recompilation of the GSE is needed, when using this feature.

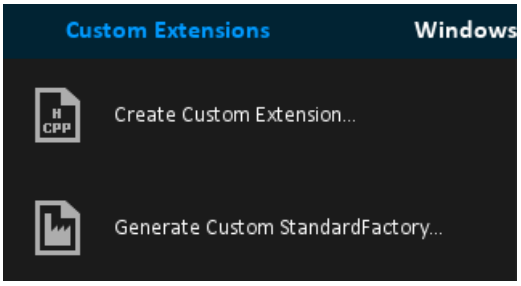


Fig 41 - Custom Extension

NOTE: For more information and an example, please refer to the document "Custom Extensions.pdf".

The generate custom StandardFactory wizard helps to create your own custom standard factory. This factory will only contain entries for controls that are used in the current project. This enables the linker to remove unused code for not used controls.

3.7 The Windows Menu

All editor windows (Controls window, Workspace window, Object Hierarchy window, Attributes window and Dialog window) can be manually moved around or can be hidden. To reopen them, use the windows menu.

With "Auto-Arrange Windows" you can open all available windows and arrange them within the GSE.

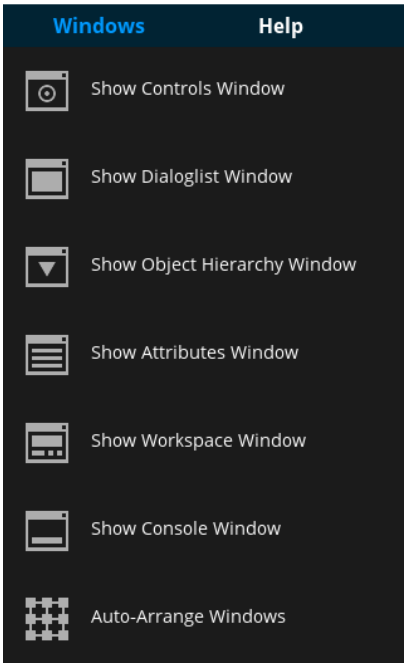


Fig 42 - Windows Menu

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

3.8 The Help Menu

Important information about the editor is present in the help menu.
The “About” will open a window containing the editor version, copyright and contact information. Also the currently used number of resources and the limits if set are shown.

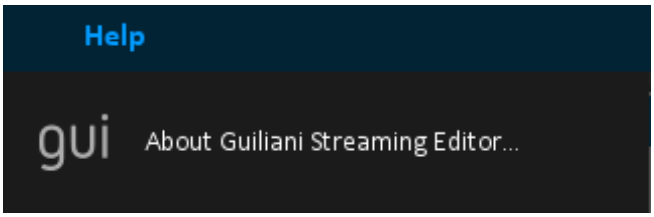


Fig 43 - Help Menu

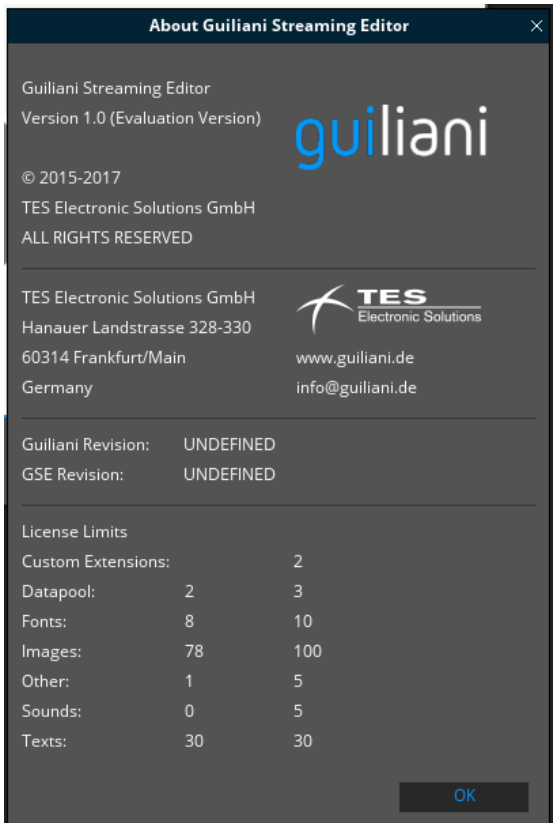


Fig 44 – About Dialog

4 The Dialog-List-Window

The “Dialog-List” window lists all dialogs of your project. The highlighted dialog is currently visible in the Dialog window.

You can switch the dialog currently edited in the Dialog-Editor by selecting one of the listed dialogs.

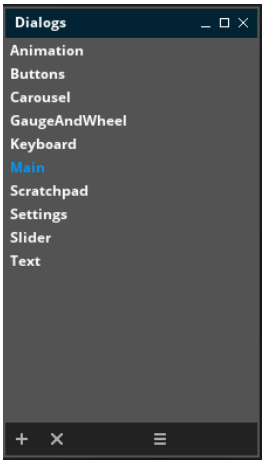


Fig 45 - DialogList Window

At the bottom of the window there are several buttons to operate with the dialogs.

To add a new dialog use the “+” icon.
To delete the selected dialog use the “x” icon (at the window’s bottom).
Switch between text list and thumbnail display with the icon on the right.

5 The Workspace-Window

Use the Workspace window to arrange objects on the screen.
Drag and drop them into or out of containers or resize them (use <CTRL> to select multiple objects).

If an object is selected all of its attributes are listed in

Fig 46 - Dialog Window

The Attributes Window (chapter 0) and it's highlighted in

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

The Object Hierarchy Window (chapter 7).

By expanding the Workspace window's size, you will not influence the dialog's size itself. Nevertheless, you may reveal objects, which were off screen and can edit them now.

If you want to resize the edited dialog, select the root of the tree (first entry) in the hierarchy view and edit the width and height variables in the Attributes window.

5.1 Selecting objects

You can select objects by clicking on them. A selected object is displayed with an outlined moving rectangle. To de-select an object, just press the CTRL-key and click on it.

When you want to select multiple objects you can click on each of them holding down the CTRL-key or hold down the SHIFT-key and draw a frame. This will select all objects which are entirely contained inside the drawn frame.

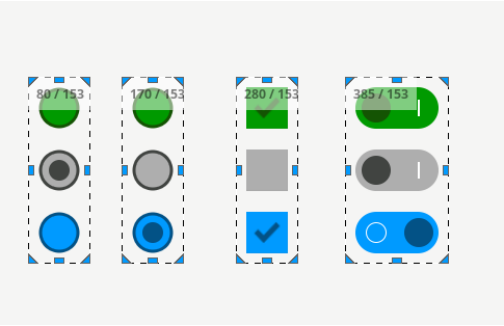


Fig 47 – Multi-Selection

NOTE: when using Multi-Selection be sure that no object is currently selected. Otherwise the currently selected object will be moved instead of drawing the selection-frame.

5.2 Moving and resizing objects

To move an object, just select it and move the mouse while the left button is pressed. While moving the object the numbers which are displayed in the upper/left part tell you the current position of the object relative to its parent.

Fig 48 – Moving/Resizing an object

When you want to resize an object, just select it and drag one of the triangular- or rectangular-shaped and coloured points around the object-frame.

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

Triangular-shaped points will change both width and height of the object into the direction which is used (top-left, bottom-left, top-right or bottom-right)

Rectangular-shaped point will change only width or height depending on the position (top, left, bottom, right)

NOTE: While resizing an object the numbers which are displayed in the upper/left part tell you the current size of the object, not the position.

5.3 Guidelines

The artist positions the objects in the dialog. Guidelines assist here by snapping the objects to a common edge. The snapping feature is provided between the moved object and each other object for the left, right, upper and lower edge, as well as horizontal and vertical center line. The guidelines can be toggled on/off in the bottom bar of the dialog window.

Fig 49 - Assisting Guidelines

6 The Attributes Window

Edit objects in the Attributes window. Add object specific Image-, Text-, Sound- or Font IDs, commands, layouters or behaviours. Assign an ID to an object to identify it in The Object Hierarchy Window and reference it from other objects through commands, behaviours etc.

Fig 50 - Attribute Window

6.1 Images

Objects may have five different states (for example the button). These include standard, highlighted, pressed, grayed out and focused. In the Attributes window, you can add the state-dependent image to the object by clicking on the appropriate image ID button and selecting the desired image in Manage Images window.

6.2 Texts

Objects with texts have a text string that can be edited in the “Text” input field of the Attributes window. (See picture below). For internationalized Text, use the Text ID-Button and select a Text ID. If you do not have a Text ID yet, click on Manage Texts in the dropdown box, which will open the Manage Texts window. Here you can define Text IDs and translate your texts in different languages.

The position of an object’s text is per default fixed in the left corner. If you resize the object, its text remains as is, it will neither reposition nor resize. You can thus freely position a text inside its associated object by changing its TextXPos / TextYPos and TextWidth/TextHeight attributes. Additionally you can set the texts alignment by changing the VerticalAlignment- or HorizontalAlignment attributes.

6.3 Fonts

While you can make changes to the text colors in the Attributes window itself, you will need the Manage Fonts for changing the font face or the font size. Press the TextFontID button in the attributes window to select the object's Font ID.

6.4 Commands

Commands encapsulate specific actions. In this way they can easily be reused and attached to objects within the GUI. Commands can cover calls to functions of the underlying application log (e.g. Start Playback of an MP3 File) or trigger actions inside the GUI itself (e.g. Switching screens).
They can be directly assigned to certain widgets that support them, for instance buttons or input fields. Commands are typically executed by the widget when certain conditions are met or specific events occur. For instance, the button executes its command when clicked on. The input field executes its command when the user finishes input by pressing ENTER.

If you wish to execute commands from widgets that do not have ready-to-use Command-slots, you do so by using the SingleCmdBehaviour. This predefined standard behaviour enables you to execute commands for any object in reaction to various events (e.g. Clicking, Dragging, Getting the focus…).

There are predefined commands that you can use, like the Quit command, it provides a simple way to create a 'quit application' button. Another command that Guiliani offers by default is the load dialog command, it loads another XML file at runtime. This is useful to create screen switches or pop-ups. For real applications, you can extend this list with your own application specific commands and comfortably use them within GSE.

Commands can link to another command to execute more than one action at once, when building a command chain. To do this, look for the attribute 'AdditionalCmdCount' in the attributes list and click the button 'Add more' next to it.

6.5 Behaviours

Behaviours are used for adding functionality to Guiliani's event slots. Each widget has numerous event slots that are called by the framework when specific events occur, like key presses, mouse clicks, mouse drags and so on. Please refer to Guiliani's documentation for in detail information on this concept.

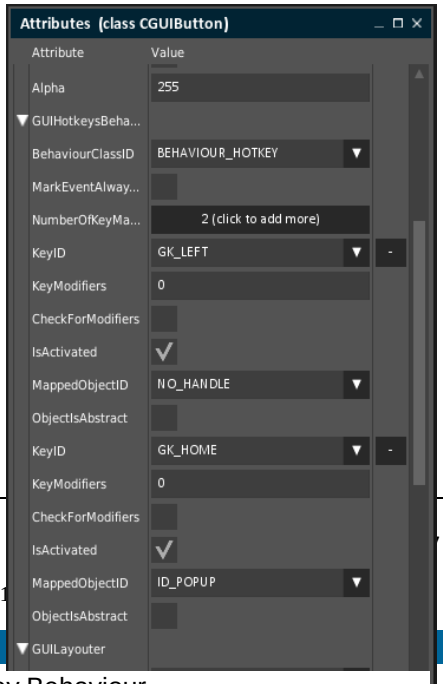


Fig 51 - GUI Hotkey Behaviour

A powerful aid when creating your GUI will be the behaviours related to hotkey-handling and command execution, these are explained now.

Behaviours and commands can be tied together by using two special behaviours, the MultiCommand and SingleCommand behaviours.

Select the multiple commands behaviour for assigning commands to each of the event slots, or use the single command behaviour to choose one event slot only and attach a command to it.

The Hotkeys behaviour reacts to key presses by the user. It is triggered whenever a key press occurs and is not handled in any way, for instance by a widget. Keys map to object IDs: When a defined key press is detected, the mapped object is searched and, if it was found, is 'clicked'. In other words, you can use this behaviour to simulate clicking on objects through keyboard keys.
The Hotkeys behaviour should be added to a parent object (the root of an object group). Add the ASCII key code in the KeyContent text input field. You can remove the key mapping by pressing the X-icon on the right side. Map the behaviour to an Object ID by selecting it through the appropriate button. If the user presses the key, the mapped object will react like it was clicked, for example, a check box will switch from selected to unselected when the key is pressed.

6.6 Basic Object Specific Attributes

Each GUIObject has some basic attributes. These are explained below.

6.6.1 Position

Every object has an X and Y Pixel-Position in your GUI. If you add a new control to your GUI it will be displayed in the left corner (zero-point) of your GUI, meaning its coordinates will be X=0 and Y=0. Mind that Guiliani can position objects sub-pixel accurate, so X or YPos inputs like 0.5 are possible.

Positions are relative to object's parents, so an object that seems to be located in the middle of a dialog does not necessarily have coordinates that are close to the dialog's center. Instead, the coordinates express an offset to the object's parent (the container that contains this object – look for the object that is one level up in the hierarchy view).

6.6.2 Width and Height

An object's width or height is given in pixels and can be changed in the Attributes window, or by dragging the corners of the object in the preview window. .

6.6.3 Focusable, Grayed Out, Disabled and Invisible

A focusable object can receive the focus in the running application. This is necessary if the object is supposed to be selectable via the keyboard / cursor-keys.

A grayed out or disabled object will not react to user input in the running application, so it cannot be clicked and will not be focusable either.

An invisible object will not be displayed in your application and will therefore not react to any events.

6.6.4 BehaviourClassID

Choose behaviour by clicking the button and selecting a behaviour class ID from the list. The chosen behaviour's specific attributes become visible in the attribute list in the rows following the class ID. See also Commands and Behaviours.

6.6.5 LayouterClassID

Layouters are used to automatically influence the position and/or size of child objects within a composite object. For instance, the LAYOUTER_ANCHOR can be used to 'fix' the edges of a widget to its parent. Try this by creating a composite object, putting an image inside it, assigning the LAYOUTER_ANCHOR to the image, activating the "AnchorBottom" and "AnchorRight" check boxes and then resizing the composite object. You will see how the distance of the image's bottom and right edges to the respective parent's edges are always kept the same, moving the image around as you resize its parent.

There are further layouters, for example for automatically arranging child objects in a list container. Please refer to the Guiliani documentation for details.

6.6.6 Object ID

Object IDs are symbolic names. Assign an object ID to your objects to find them in the Object Hierarchy window. Do this by adding a new Object ID in the Attributes window.

Note that Object IDs are also used by many commands and behaviours in order to reference specific objects inside the UI. Renaming an object will not automatically update these references and might cause unexpected behavior.

The option “search object-id” can be used to search for object-ids meeting certain specific requirements like objects in the current dialog or child-objects of the currently selected object.

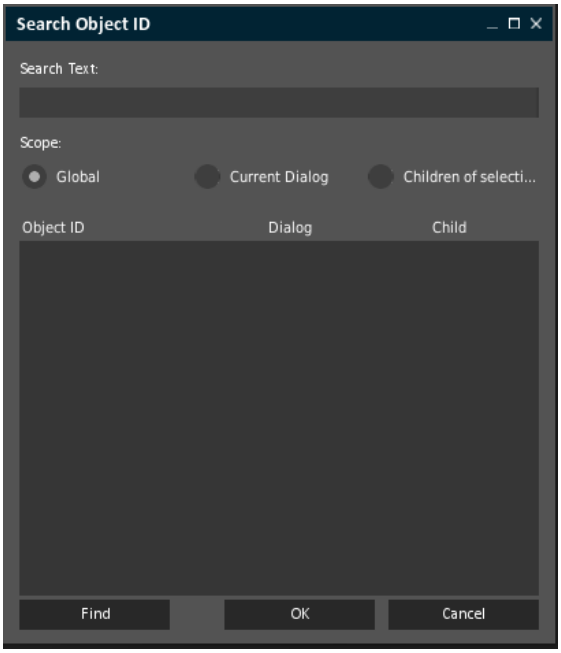


Fig 52 – Search object-id

6.6.7 Colors

If an object has a color which can be changed (e.g. TextColor or BackgroundColor) the color-selection dialog can be used to easily pick a color or choose from various favorite colors.
The dialog is opened by clicking on the button displaying the hexadecimal color-code next to the attribute-name. The favorite-colors are saved with the project and can be selected in the upper list.

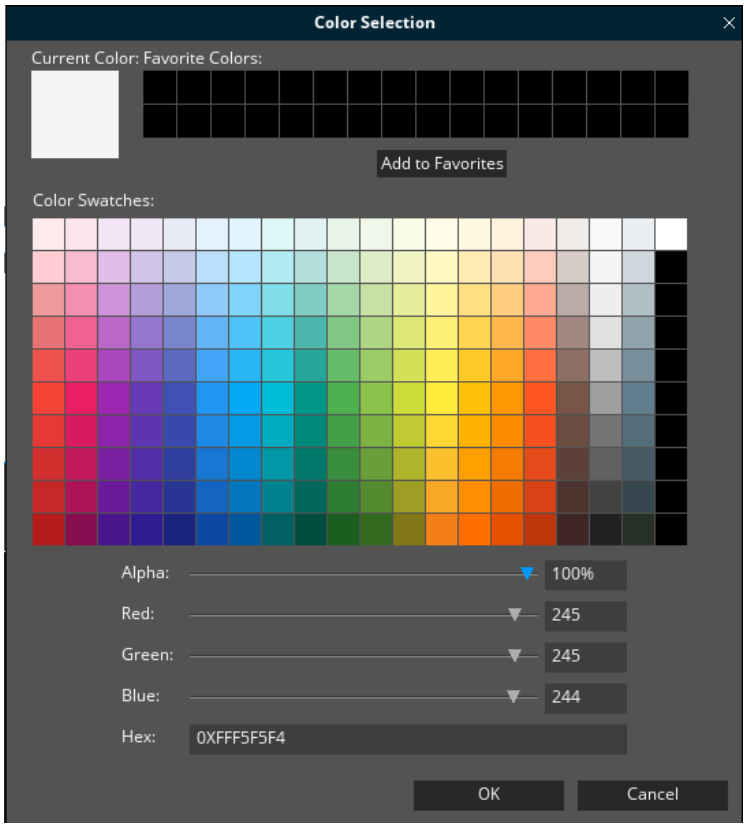
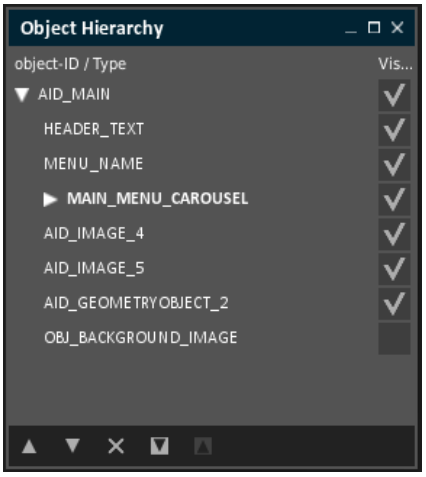


Fig 53 – Color Selection

TES Electronic Solutions GmbH Hanauer Landstraße 328-330 60314 Frankfurt am Main Germany
www.guiliani.de

7 The Object Hierarchy Window



The Object Hierarchy window displays all GUI objects contained within the selected dialog including their hierarchical structure. Objects that cannot be seen on the Dialog window because they are below other GUI elements can be selected in the Object Hierarchy window instead.

If an object is selected in the object hierarchy, it is selected as well in

The Workspace-Window and in

The Attributes Window.

For changing the hierarchical order of a GUI Element, use the arrow icons at the bottom of the window. Mind that the only possibility to extract a group's sub-object is to drag and drop it out of the group on the Dialog window.

Fig 54 - Object Hierarchy

Delete objects by clicking on the delete icon in the bottom bar.
Shift it with the up/down arrows.

You cannot change names (object IDs) of GUI-objects and group names in this window, but in the Attributes window.

Declutter the workbench by temporarily hiding GUI objects. Do this by clicking on the visibility check boxes.

Some composite-objects (such as ScrollViews) may contain content which is larger than the area covered by the object itself. In such cases you may wish to "zoom-in" on such an object by double clicking on it. This way you will be able to reach child objects which were previously not visible.

8 The Controls Window

You will find all Guiliani Standard Controls in the Controls window. If you click on one of the icons, a new instance of that control will appear in the upper left corner of the current dialog, or the currently selected composite object.

A detailed description of every single attribute of each control can be found in the document “GSE Control Attributes”

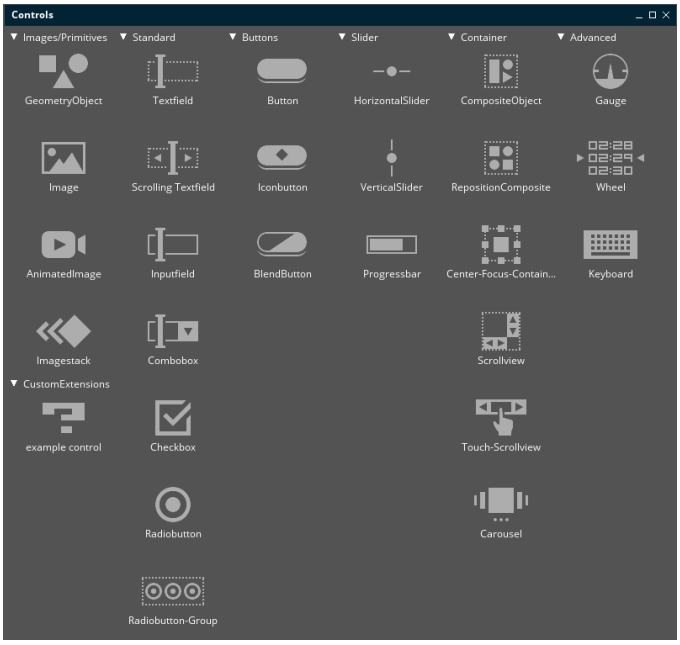


Fig 55 - Controls

Many Controls need image IDs to be displayed. There are up to five states that need to be provided with image IDs: standard, grayed out, highlighted, focused, and pressed. Controls that need one or more image are marked with an * after the name of the control in the following list.

8.1 Images/Primitives

In this group all controls representing images or graphical primitives can be found

8.1.1 Geometry Object

The Geometry Object offers a simple way to add geometrical primitives, such as lines or rectangles, to your user-interface without writing customized drawing code.

8.1.2 Image*

It displays an arbitrary image in the dialog. By default, the image stretches to the size of the widget. If you do not wish this to happen, deselect the Check box StretchBlit in the Attributes Window. This leads to the image being drawn with its original size and being centered in the widget rectangle.

The Alpha Channel influences the transparency of your image. The value 0 means that this image is invisible, 255 means it is fully visible. It can be adjusted in size and layout. It has the abilities to be focusable, grayed out, disabled or invisible and therefore can be equipped with a behaviour or commands.

By selecting an Image ID in the Attributes window, the imaged stretches to the size of the widget. If you do not wish this to happen, deselect the Check box StretchBlit in the Attributes Window. This leads to the image being drawn with its original size and being centered in the widget rectangle.

In order not to use any standard resources, the image itself must be provided and given an ID.

NOTE: It is easy to resize an image (and any other object which uses images) to its innate size by setting the image ID and afterwards clicking in the bottom bar of

The Workspace-Window on the button “Resize object to dimensions of its first image”  .

8.1.3 Animated Image

Add as many images to the animated image icon as you wish via the”Images” attribute by clicking on the “NumberOfImages” Button and choosing an “ImageID”. Choose the FrameDelay, which is the time after which the next image is displayed. Select the check box. Repeat, if you want the animated image to run in a loop.

8.1.4 Image Stack

Image stack will be used for showing pseudo 3D transitions: one image fades out into the background while getting smaller; another one comes in from the front, getting smaller and fading in. The opposite direction works similarly: the images appear to come from the back to the front while fading. In order to see the effect you must run a simulation of the GUI, focus the image stack object and cycle through it using the PageUp / PageDown keys. (Of course, it is possible to map these control keys onto other inputs for the target hardware.)

8.2 Standard

This group contains all controls which are found in a standard GUI-application for selecting or displaying information.

8.2.1 Text Field

Use text fields for displaying texts inside your GUI. Either the text can be set hard coded or via text ID, we recommend the latter one, so the GUI is ready for internationalization. It is adjusted by selecting an appropriate font ID and colors for the different states.

For some extra convenience, the text field provides the option to set a background image.

8.2.2 Scrolling Text Field

With this control, text can be scrolled from the right to the left or vice versa, as well as from top to bottom or vice versa. In addition, you can define the scrolling speed.

8.2.3 Input Field*

The input field consists of an editable text and a background object, the parent of the text. The background object has standard attributes such as Position or height only. To edit the attributes of the text input, select it in the Object Hierarchy window or in the Dialog window.

The input field can be used for entering passwords (PasswordMode).

Decide if the input may be numbers, only, or if all characters shall be allowed (AcceptedCharSet).

Choose colors for the text states (standard, highlighted, pressed, grayed out) and for the text background highlight (SelectionColor). Set the Font ID, the Font Spacing and the text alignment (top, left, right, bottom, and center).

Note: This input only supports single line input. The SingleLine check box should always be selected.

Position the text within the background object with XPos and YPos or within its border. Do this by changing the TextXPos and TextYPos value. If the TextHeight has the same value as the Height value, the text will be positioned in the height's middle.

8.2.4 Combo Box*

The ComboBox offers a Drop-down list of entries from which users can make a selection. Advanced features include the possibility to search for entries by typing substrings into the header and to add new entries at runtime in the same way.
Please be aware that even though you can create and edit the ComboBox inside GSE, you will currently not be able to fill it from there. Nevertheless, you can access the ComboBox from source-code via its ObjectID and dynamically fill it at runtime.

8.2.5 Check Box*

Check boxes provide a binary option, either check or not. They are useful for checklists and optional choices. A full-customized checkbox has 10 images due to five states when checked and the same five states when unchecked.

Check boxes allow action methods, commands and behaviours. In this way their usage is more flexible.

They can be customized by setting a text that is displayed inside the check box. Image and text can be aligned independently to satisfy individual needs.

8.2.6 Radio Button*

Radio buttons present a choice between several mutual exclusive options. Therefore, they come in groups, from which exactly one is checked at a time.
Note that Radio buttons HAVE to be attached to Radio button groups in order to work correctly.

8.2.7 Radio Button Group

The radio button group is a specific composite object in which you can align radio buttons. In a radio button group, only one radio button can be selected. All others are unselected, even if all radio buttons have the selected checkbox in the Attributes window selected.

8.3 Buttons

In this group all sorts of buttons can be found

8.3.1 Button*

Buttons are one of the main control elements. They trigger some action, like open, export, ok or cancel.

Buttons allow commands and behaviours, which makes their usage more flexible.

The buttons can be altered in their visual appearance by providing image IDs for the different states. It has all five states that are mentioned above. For each of the states a different image ID can be set.

8.3.2 Icon Button*

The icon button is a standard button with an additional image as overlay whose position can be changed independently. Both the main button image and the icon have five states (standard, grayed out, highlighted, focused and pressed).

8.3.3 Blend Button*

The Blend Button is a specialization of the standard Button, which softly blends between the various state images instead of simply switching them. Be aware that alpha-blending can be time consuming on low end platforms, in particular if blending is done using a software-renderer.

8.4 Slider

8.4.1 Horizontal and Vertical Slider*

A slider has these images: A background image and a knob that can be normal, highlighted or pressed.

Change the slider's orientation from horizontal to vertical or vice versa by selecting the appropriate slider SLD_HORIZONTAL or SLD_VERTICAL.

In the BackgroundMargin input field, you can shorten the length of the background image without influencing the span in which the knob can be moved. The margin may range from 0 to 100.

By default, the knob can be moved in the span that is as long as the background image. If you would like to reduce the span-size, edit the width input field of the horizontal slider or the height input field of the vertical slider.

To change the sliders starting point (from left to right or from up to down), switch from BASE_AT_MINIMUM to BASE_AT_MAXIMUM.

Edit the value range in the MinValue and MaxValue input field and influence the StepSize as desired. You may choose 100 as a MaxValue and StepSize 10 for example if you want to display 100 percent and the slider showing every 10 percent of it.

Value represents the current value within the defined range. It does thus also define the position of the knob within the slider.

8.4.2 Progress Bar*

The progress bar is quite similar to the slider's attributes. It has less images, one for the background, the other one for the bar itself.

The bar can be positioned relative to the left corner of the background image (edit the BarX and BarY value). The width and height of the bar image can be changed, too (BarWidth and BarHeight).

The progress bar has two ProgressBarTypes. The first type is designed for full control of the fill state via the application. The second type is used for representation of unpredictable duration of operations. For this type you can choose between three different loop modes. The first will fill up the bar until full and starts repeatedly again with an empty bar. The second one will repeatedly move the bar from start to end. The last loop mode will move the bar from start to end and back to start again.

Edit the value range in the MinValue and MaxValue input field and influence the StepSize just as you wish. You may choose 100 as a MaxValue and StepSize 10 for example, if you want to display 100 percent and the progress bar showing every 10 percent of it.

The ProgressBar can show a timed animation. The animationInterval attribute influences the delay between animation steps. It defaults to 30 milliseconds.

You can change the fill direction to right-to-left with the "Base"-ComboBox. Vertical ProgressBars can be created by checking the DrawVertical check box.

8.5 Container

Here you will find all container-controls available for grouping or dynamic placement and interaction.

8.5.1 CompositeObject

CompositeObjects serve as containers, which you should use to group other objects. This has several advantages:

- Moving the container will move all child objects, as well
- Resizing the container can resize the children (when using Layouters)
- Marking a container as Invisible will render its children invisible, too

- You can use the container to clip its content. (That means everything contained inside the Composite object, but positioned outside its dimensions will not be visible)

8.5.2 Reposition Composite

A reposition composite object is a composite object that will automatically reposition its child objects. You can align all children at the top, bottom, left or right of the composite and define the space between the child objects and the space to the composite-border. You can even have a composite object in a reposition composite and vice versa.
Note: The repositioning effect will only take place when resizing the container. Therefore, drag its lower right corner to see the effect.

8.5.3 Center Focus Container

The center focus container changes its own position with an animation so that the focused child object is centered on a specific point (try it; it is much easier to see than to describe). You can define this "center point" with the CenterX and CenterY coordinates that are given relative to the Center Focus Container's parent.

8.5.4 Scroll View*

A scroll view object is a specific composite object that represents a view window onto a larger scrollable area contained therein. This is useful when you need to make a larger amount of information or widgets accessible on limited display space.

Take a look at the Object Hierarchy window. You will see that the scroll view object consists of several objects. Two of these are a vertical and a horizontal scroll bar. An additional composite object keeps the actual content that shall be visible in the scroll view window. As long as the content is smaller than the scroll view window, the scroll bars are set to invisible. If the content's height (or width) increases, the scroll bars will be automatically visible so that you can scroll the content. The scroll bar's visibility policy may also be changed so that they are always visible (see

The Object Hierarchy Window for information on how to “zoom in” on the content of Scroll Views). All objects that should be displayed in the scroll view object must be child objects of the “ScrolledContainer”.

8.5.5 Touch Scroll View

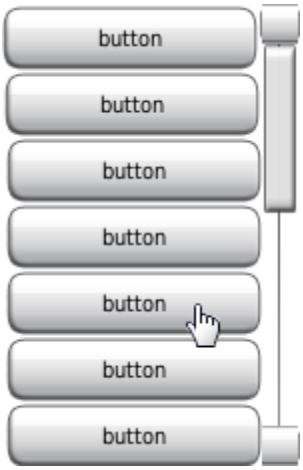


Fig 56 - Touch Scroll View

The "touch scroll view" is a specialized scroll view that is optimized for touch screens and supports scrolling by dragging as well as kinetic effects

When dragging, the content of the scroll view follows the finger/mouse. Besides that, it is possible to activate kinetic scrolling which will trigger a scroll animation after the drag that slowly fades out with time. The animation speed depends on the drag speed, which means that quick fling gestures will result in faster scrolling. Optionally it is possible to activate a bounce back factor, which will invert the animation direction when the edge of the scroll view is reached.

It is also possible to define a destination position (relative to the scroll view). When this is done, the scroll view will always choose one object that is the 'active object' and align its center to this position. The destination position is only defined for one direction. Depending on the destination direction (horizontal or vertical), the target position is used for the XPosCenter or the YPosCenter of the active object. When clicking on the scroll view, the nearest object to the click position (in destination direction) is chosen as the active object. When dragging, the nearest object to the target position is chosen as active object when the drag is finished. When kinetic scrolling is activated, the nearest object to the target position is chosen when the kinetic animation is about to fade out.

8.5.6 Carousel

Objects you add to the carousel will automatically be positioned in a 3D rotating wheel. The focused child is always moved to the 'front' of the carousel with an animation. The carousel's tilt angle and radius can be set with the respective attributes. The radius should be large enough to allow children to be positioned far enough apart from each other. However, keep it small enough to avoid clipping of the children at the carousel's edges. The size of the children cannot be changed.

8.6 Advanced

8.6.1 Gauge

The gauge control is used to visualize values on a meter using a needle. The visualization can be customized by either supplying images, for both the background and the needle, or by drawing the needle with a line of customizable color and length. The usage of the gauge control is very easy, as only minimal value, maximal value and maximal rotation have to be defined. The current

position of the needle will be calculated automatically through the given value. The "gauge" control can be used in a wide range of use-cases e.g. to present an analog clock or a speedometer.

8.6.2 Wheel

The wheel-control can be used to display different numeric entries in a horizontal or vertical way. An entry is selected by dragging the wheel to this entry. Font and color can be adjusted for normal entries as well as for the selected entry.

8.6.3 Keyboard

The on-screen keyboard is mainly used for entering text into inputfields when you don’t have a physical keyboard available on the target.

8.7 Custom Extensions

Here all available CustomExtension you created are listed.

8.7.1 Example Control

This control is an example CGUIObject implementation that draws a rectangle with configurable border width and configurable colors. This is available as source code and you can use it as an example for creating your own controls (see Custom extension).

9 Debugging and Trouble Shooting

This chapter contains advice on how to debug an application that is built using GSE.

9.1 The Console Window

The console window contains all the log output generated by GSE, Guiliani or its widgets.

For developers the console is very useful, since it can contain valuable information on your newly developed custom widget that simply refuses to do what it is supposed to do. It will for instance let you know if you are trying to access an illegal image resource, or if you are trying to find an ObjectID, which does not exist.
For users of GSE that are non-developers the console can still prove helpful, for example when for some reason one of the project's XML files got corrupted. In this case the console will yield information telling you which file was corrupted, what GSE expected to read, what it actually did read, and likely also the line-number within that file.

Overall, it is recommended to have a look at the console whenever something goes wrong and you need more details on the cause of the problem.

NOTE: The GSE writes the same information into its .log file.

9.2 Dealing with corrupted XML Files

If you are opening a project and a MessageBox pops up, telling you that a file is corrupted, then this is usually caused by an erroneous XML-File, or by a version conflict. In such a case, open the console window to find out more about the cause of the issue.

Usually the output will point you to the file causing the problem, and likely even to the precise line number.
Have a look at the following example log output:

```
ERROR: GUICompositeObject::ReadFromStream: Incompatible class version! Current version is 3, read version is 42.
ERROR: CGUIFactoryManager::LoadDialogFromFile: Caught streaming control exception. Problem encountered while
streaming file Unknown, user file (Line 6)
ERROR: DialogManager::CreateDialogViaFactory: Dialog
C:\Projects\GUILIANI_Workspace\apps\GSE_clean\examples\EditorDemo\240x320\Main.xml is corrupt.
ERROR: DialogManager::CreateDialogViaFactory: Caught an exception while loading dialog from file.
```

You can see that the problem was a version conflict. The first line tells us that the file contained GUICompositeObject of version 42, while the version of Guiliani/GSE works with a GUICompositeObject of version 3. The next line points us the line-number within the xml file

where the problem was encountered (Line 6). The third line finally gives the name of the corrupted file, which is "Main.xml" in this case.

Now that you know the root of the problem, you can either fix it manually in the XML file, or contact the person from which you received the file and ask for an update.

9.3 Debugging StreamRuntime

The executable of your StreamRuntime-Application typically resides in the subdirectory /temp of your GSE-Project's directory. How to debug it, depends heavily on your IDE. For VisualStudio we can recommend the following two approaches:

Approach 1:

1. Set your StreamRuntime application as the "StratUp Project" by right clicking on it.
2. Set the application's working directory (Project Properties->Debugging->Working Directory) to the directory which contains the project's resources (e.g. the GSE-Project's /temp directory)
3. Start Debugging

Alternatively…

Approach 2:

1. Start the application directly from GSE via "File->Run Simulation..."
2. Use VisualStudio's "Tools->Attach to Process" to attach the debugger to your application (typically named StreamRuntime.exe)

10 Tool Tips

The GSE provides many tool tips. You can find tool tips when you rest the mouse pointer above an icon (e.g. the “Bring object to foreground” button in the dialog window). This will give you short information about the function.

11 Contacts

If you have any questions on Guiliani or the GSE, please feel free to contact:

support@guiliani.de
www.guiliani.de

This project is supported by the Federal Ministry of Economics and Technology by a resolution of the German Parliament.