guiliani.de

Call Application API

A simple example on how to call application APIs from a GUI designed in GSE

Product:	Guiliani Streaming Editor (GSE)
Release version:	2.3
Release date:	March 26, 2019

Table of contents

1. Introduction / Intended audience

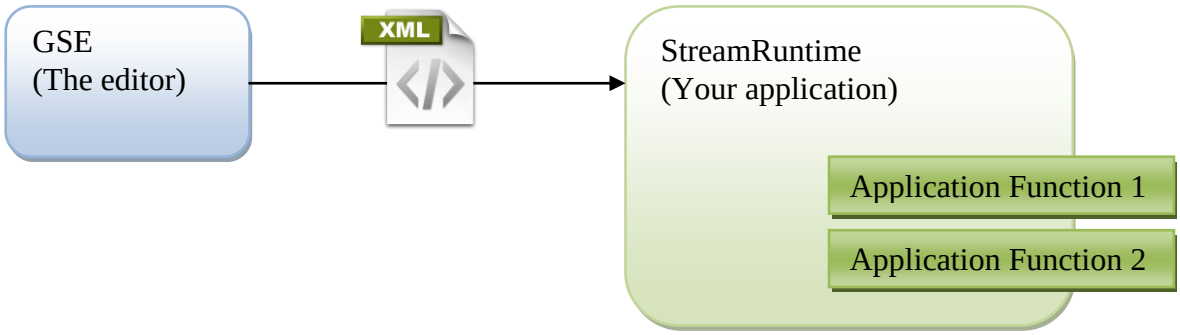
This manual explains how to call application specific code from a user interface which has been designed within GSE. This shall serve as a starting point for developers new to GSE, that search for a quick and easy way to call their own application code in response to an event in the user interface.

2. Preparation and compilation

In order to compile, link and run your own StreamRuntime-application please refer to the *Custom Extensions.pdf* document, which gives detailed instructions on this matter.

3. Overview

The typical setup of an application built with GSE is as follows:



There is the GSE on one side, in which you graphically design your user-interface by placing controls, assigning bitmaps, changing texts etc.

The resulting user interface is then exported as a set of XML-Files (plus resources such as bitmaps, fonts…)

Ultimately, there is the actual application – which we refer to as StreamRuntime, because it loads and interprets the XML description of the user interface. The StreamRuntime that comes as source-code with GSE is merely a generic skeleton, which does all the necessary initialization of

the framework and loads the initial dialog. You are free to extend it with all functionality required for your individual application.

The question is now: How can I call a function of my application, if GSE does not know anything about the application itself?

4. Calling an application API from a command

Within the subdirectory *StreamRuntime/Source/CustomExtension* of your GSE package you will find a file named *CallApplicationAPICmd.cpp* which demonstrates how you can call code within StreamRuntime.

Essentially, all the magic happens within the tiny method called Do()

```
void CallApplicationAPICmd::Do()
{
#if defined(STREAMRUNTIME_APPLICATION)
    // This is the CallApplicationAPI method in MyGUI.
    // Feel free to extend CallApplicationAPI as required.
    GETMYGUI.CallApplicationAPI( m_kAPI, m_kParam);
#endif
}
```

Let's go through the code line by line. The method Do() is predefined by the base class interface (which is *CGUICommand*) and will be automatically called whenever a command gets executed. E.g. if a command is attached to a button, it will be executed if the button gets clicked.

The second line makes sure that the code within Do() does only get compiled within your application (=StreamRuntime) and NOT within GSE. This is necessary since GSE obviously does not know anything about the application's internal interfaces.

The final line calls an API of the application, which is called *CallApplicationAPI* and hands over two parameters. The next chapters explain where these come from.

5. Editing parameters within GSE

The parameters which are handed over to the application can be configured within GSE. This is done via the code in ReadFromStream() and WriteToStream().

```
m_kAPI = GETINPUTSTREAM.READ_STRING( "ApplicationAPI");
m_kParam = GETINPUTSTREAM.READ_STRING( "Parameter");
```

As you can see, the parameters are expected to be of type STRING and they will be shown in GSE with the labels *ApplicationAPI* and *Parameter*.

6. The application-side

We already had a look at the GSE-side of things and saw that the command calls an API named *CallApplicationAPI*(). You will find this in the directory *StreamRuntime/Source* in the File *MyGUI_SR.cpp*.

```
eC_Bool CMyGUI::CallApplicationAPI( const eC_String& kAPI, const eC_String& kParam)
```

This is the method which gets called by the aforementioned command. Feel free to add further handling with regard to the passed parameters as required.