guiliani.de

How to 6 - using CallApplication APIs in GSE

A guide to enlarge your GSE project with CallApplication APIs

Product:	Guiliani Streaming Editor (GSE)
Release version:	2.2
Release date:	December 05, 2017

©2017 TES Electronic Solutions GmbH

Disclaimer

The information in this document is subject to change without notice and does not represent a commitment on the part of TES Electronic Solutions GmbH. The software described by this document is subject to a License Agreement and may not be copied to other media. No part of this publication may be copied, reproduced or otherwise transmitted or recorded, for any purpose, without prior written permission by TES Electronic Solutions GmbH.

"Guiliani.de", "Guiliani", "Guiliani Streaming Editor", "GSE" and associated logos are (registered) trademarks of TES Electronic Solutions GmbH.

Windows, Visual Studio and Visual C++ are registered trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks are the property of their respective owners and use of them does not imply any affiliation with or endorsement by them.

Guiliani, guiliani.de and GSE are products of

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

Email: support@guiliani.de
Website: http://www.guiliani.de
Forum: https://www.guiliani.de/forum/
HelpDesk: https://guiliani.on.spiceworks.com/portal

©2017 TES Electronic Solutions GmbH

Table of contents

1. Introduction 5
1.1. Assumed knowledge 5
1.2. Prerequisites 5
1.3. Documentation conventions 5
1.4. Short cuts 6
1.5. This document's goal 7
2. Using a CallApplication API with programming the GSE 8
2.1. Load project "step by step" 8
2.2. Adding CallApplication API in GSE 8
2.2.1. Adding a command to burner 1 8
2.2.2. Adding a command to burner 2 10
2.3. Adding CallApplication API in source code 11
2.3.1. Loading our project to the IDE 11
2.3.2. Add CallApplication API "SetPotNumber" 14
2.4. How to continue? 26
2.4.1. Sample solution 26
2.4.2. Continuing How Tos 26
3. Index 27

©2017 TES Electronic Solutions GmbH

Table of figures

Fig. 1 This “How To” result....7
Fig. 2 ApplicationAPI entered....9
Fig. 3 Start IDE....11
Fig. 4 StreamRuntime as StartUp Project....12
Fig. 5 First start of GSE....13
Fig. 6 CallApplicationAPI in Output window....13
Fig. 7 Open MyGUI_SR.cpp....14
Fig. 8 Function CallApplicationAPI....15
Fig. 9 Added our if-case....15
Fig. 10 our ApplicationAPI in Output....16
Fig. 11 Open MyGUI_SR.h....17
Fig. 12 Added SetCookingPotNumber to MyGUI_SR.h....18
Fig. 13 Added SetCookingPotNumber to MyGUI_SR.cpp....19
Fig. 14 Open UserObjectResource.h....20
Fig. 15 Select .h files from GSE....21
Fig. 16.h files copied to StreamRuntime....22
Fig. 17. UserObjectResource .h....23
Fig. 18. Include GUITextField.h....24
Fig. 19. Add TextField toSetCoockingPotNumber function....25

©2017 TES Electronic Solutions GmbH

1. Introduction

This document explains step by step how to use CallApplication APIs in a GSE project.

1.1. Assumed knowledge

- Basic handling of GSE
- It is recommended to read “How to 1 - build a project step by step” and “How to 5 - using of CMake and an IDE for GSE”, first.
- For a better understanding on how the project works, we suggest reading “How to 2 - using Commands in GSE”, “How to 3 - using a DataPool in GSE” and “How to 4 - using a Behaviour in GSE”, too.

1.2. Prerequisites

- Unzipped GSE SDK
- Your project “step by step” created in “How to 4” (or the “step by step” project inside the folder called “*How to 4 - sample solution*” inside the documentation folder)

1.3. Documentation conventions

Whenever you can use keys from your computer’s keyboard, these will be displayed in square brackets (e.g.,” To run your project press [Ctrl] + [r].”).

Menu commands or file path used in this document will be shown in *italic*.

Text that appears in the software on controls will be printed in **bold and blue**.

Whenever the reader of this document has to do something in his project, the text will start with this triangle.

Results will be shown using this arrow.

©2017 TES Electronic Solutions GmbH

In this document, we use icons whenever we will warn the user or will give him additional or important information.

The speech bubble icon will show additional helpful information.

Whenever a text begins with an exclamation mark icon, it contains important information that is essential for the current chapter.

A warning sign icon signals serious issues and potential risks that require your full attention.

1.4. Short cuts

In the documents, we often select a command from the window. These can be selected by the following short cuts, too:

File ➔ *New Project…*	[Ctrl] + [Shift] + [n]
File ➔ *New Dialog…*	[Ctrl] + [n]
File ➔ *Save Project…*	[Ctrl] + [s]
File ➔ *Run Simulation…*	[Ctrl] + [r]

©2017 TES Electronic Solutions GmbH

1.5. This document's goal

At the end of this How To you will have learned

- how to use CallApplication APIs
- send information between different controls
- and more

The main dialog will look like this:

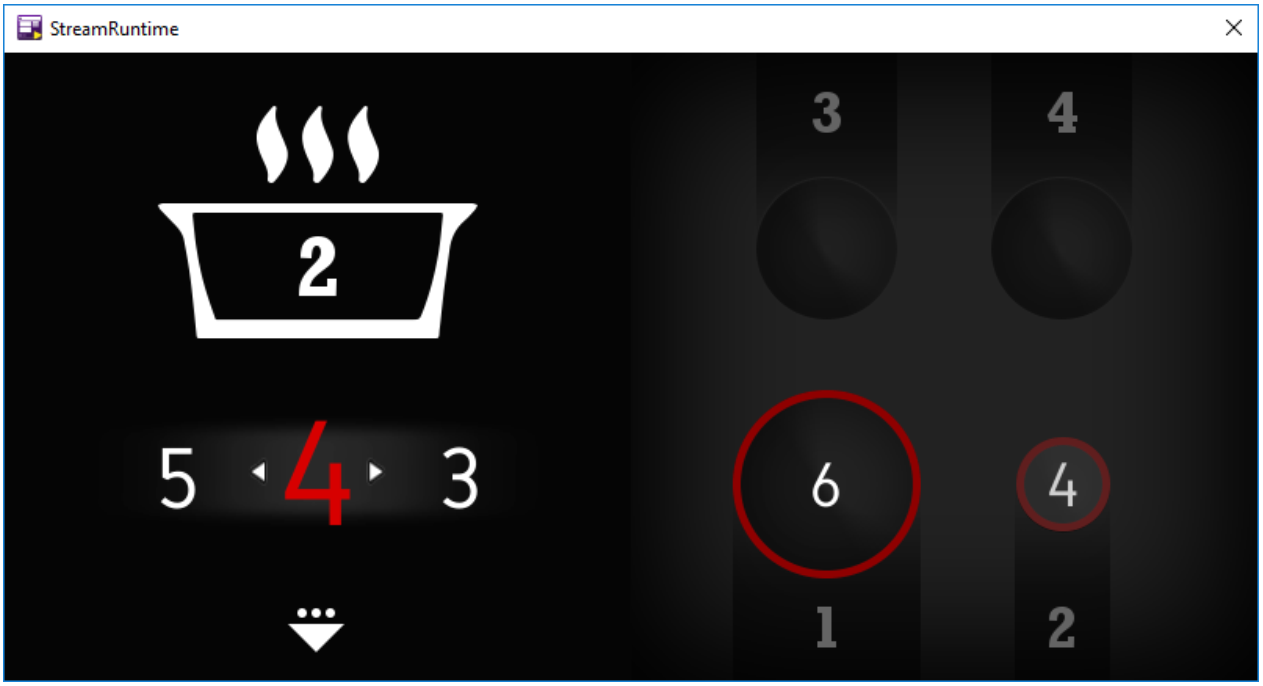


Fig. 1 This "How To" result

©2017 TES Electronic Solutions GmbH

2. Using a CallApplication API with programming the GSE

2.1. Load project “step by step”

- Start the GSE and load the created Step by Step project from HowTo 4 (step_by_step.gpr).

2.2. Adding CallApplication API in GSE

Whenever a user clicks onto a burner, the appropriate number has to be shown in the pot.
In the pot, there is a text field. When the user clicks into a burner, the burner’s number has to be send to the text field.
We will realize this by using the command “**CMD_APPLICATIONAPI**” out of the image we laid above the burner 1 (in HowTo 4).

2.2.1. Adding a command to burner 1

For some controls (like a button) the attribute “**GUICommand**” exist. There we could use directly the “**CMD_CALLAPPLICATIONAPI**” command.
Unfortunately an image has no such attribute. But we can have one making a small detour – using a behaviour.

- Select **AID_IMAGE_5** in the “**Object Hierarchy**” window.
- Look for the attribute **GUISingleCmdBehaviour**.

There you will find the commands we added in HowTo4.

- Click onto the button “**(click to add more)**” behind one of the four attributes “**AdditionalCmdCount**” and choose “**CMD_CALLAPPLICATIONAPI**”.
- In the attributes list, the attribute “**CallApplicationAPICmd**” has been add (below the last **GUISetObjectStateCmd**).

Now we have to enter the name of the **ApplicationAPI**. Additionally we can enter an optional parameter value.

©2017 TES Electronic Solutions GmbH

Let us call our CallApplication API “SetPotNumber”. As parameter, we will send the burner number.

- Enter “SetPotNumber” as **ApplicationAPI** and “**1**” as Parameter.

- The attributes should look like this:

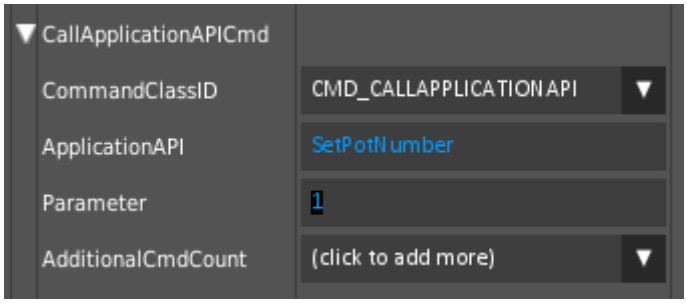


Fig. 2 ApplicationAPI entered

To see a change, we have to add the command to burner 2, too.

©2017 TES Electronic Solutions GmbH

2.2.2. Adding a command to burner 2

- Select **AID_IMAGE_6** in the “Object Hierarchy” window.
- Look for the attribute **GUISingleCmdBehaviour**.

There you will find the commands we added in HowTo4.

- Click onto the button “**(click to add more)**” behind one of the four attributes “**AdditionalCmdCount**” and choose “**CMD_CALLAPPLICATIONAPI**”.
- In the attributes list, the attribute “**CallApplicationAPICmd**” has been add (below the last **GUISetObjectStateCmd**).
- Enter “**SetPotNumber**” as **ApplicationAPI** and “**2**” as Parameter.
- Save your project and run it.

Is there something new what happens, while you are clicking onto burner 1 or 2?
No, because the CallApplication API command will be handled in the source code.

©2017 TES Electronic Solutions GmbH

2.3. Adding CallApplication API in source code

- Close the GSE.

2.3.1. Loading our project to the IDE

Look for “GSE.sln” in you Build folder.

> For this HowTo, we will use the Guiliani SDK for Microsoft VS 2015 and therefore we created the folder “Build_GSE_2015_HowTo” in “How to 5 - using of CMake and an IDE for GSE”. If you are using the Guiliani SDK for Microsoft VS 2012, you might like to create and use a folder named “Build_GSE_2012_HowTo”.

> If you do not like long path names, just rename the source path BEFORE starting CMake. We have done it and renamed “Guiliani_2.1_SDK_including_GSE_Windows-Desktop_VS2015” to “Gui_SDK_VS2015”.

- Double click onto “GSE.sln” to start your development environment (or load it to your IDE).

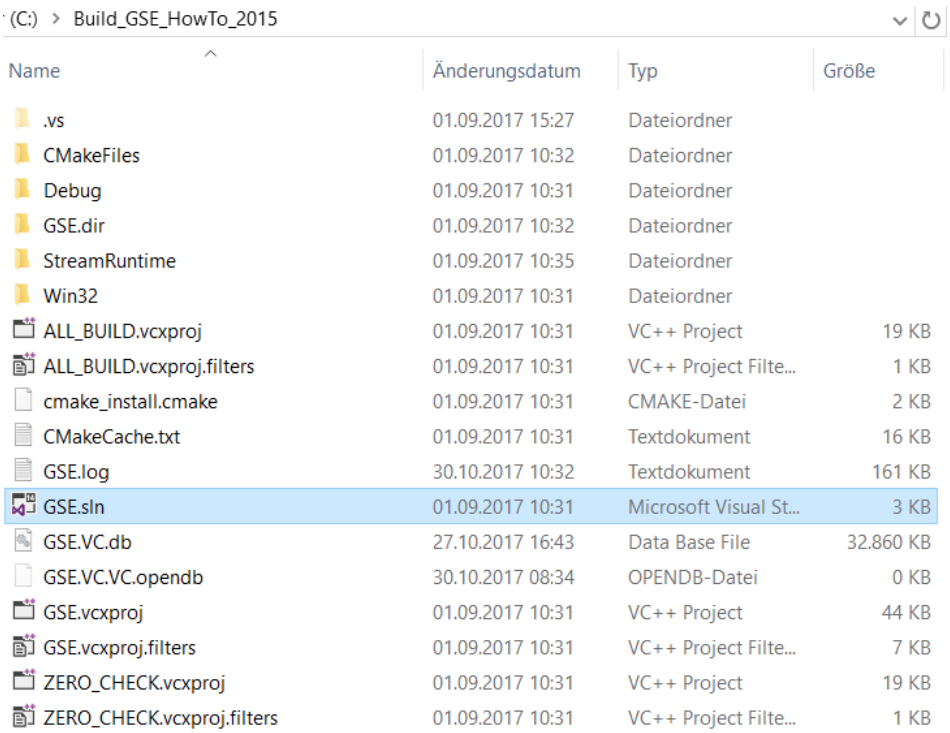


Fig. 3 Start IDE

©2017 TES Electronic Solutions GmbH

- Inside your IDE set "**StreamRuntime**" as your StartUp Project.

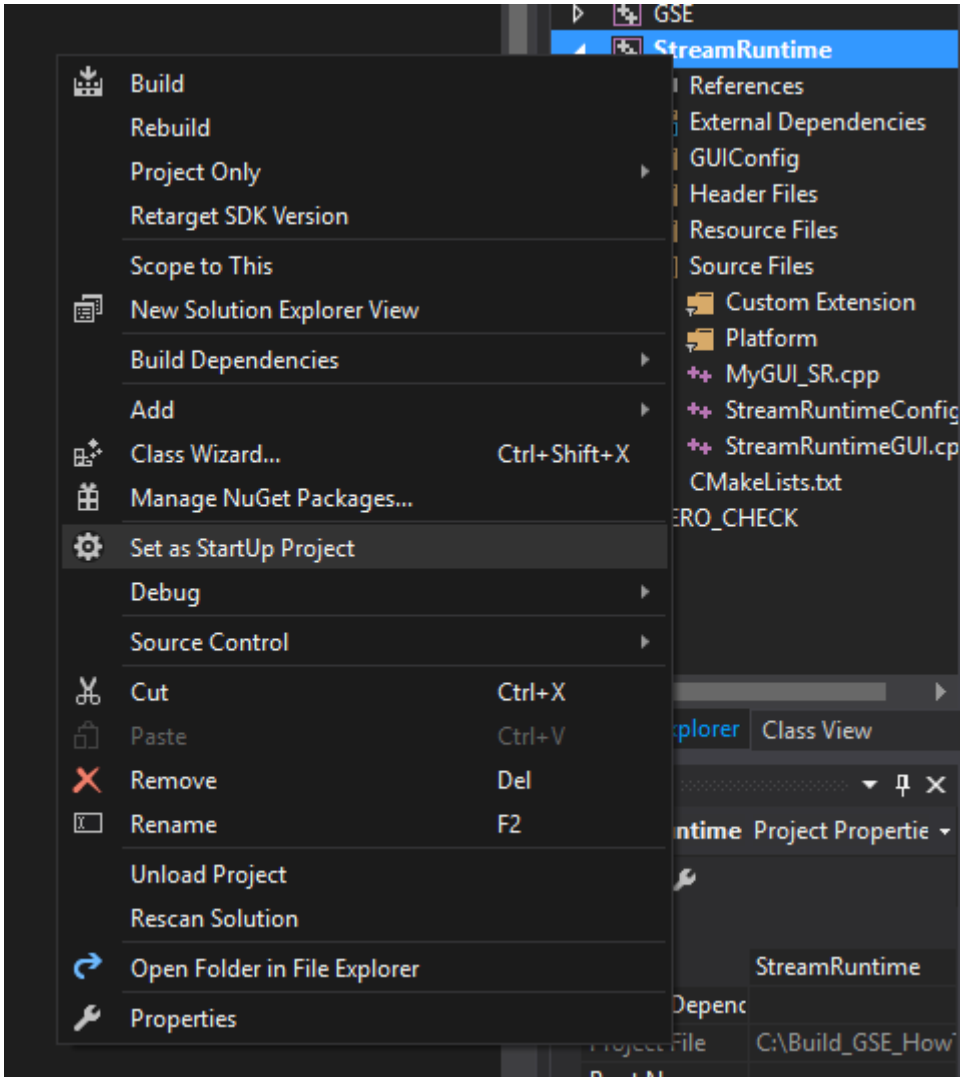


Fig. 4 StreamRuntime as StartUp Project

- When you start debugging (menu *Debug/Start Debugging* or F5) now, your project will be started.

©2017 TES Electronic Solutions GmbH

The result should look like this:

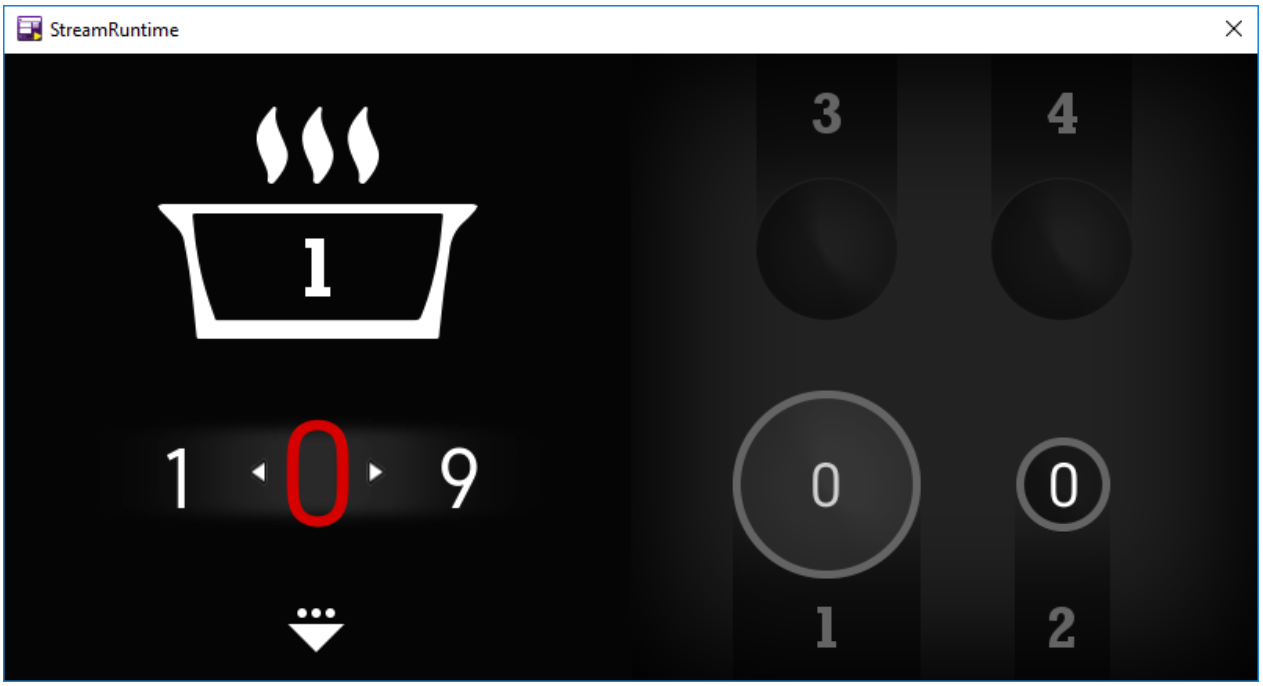


Fig. 5 First start of GSE

Click once onto burner 1 and close the application.

Look at your IDE's Output window. Scroll some rows up until you see a line starting with "INFO: CallAplicationAPI …".

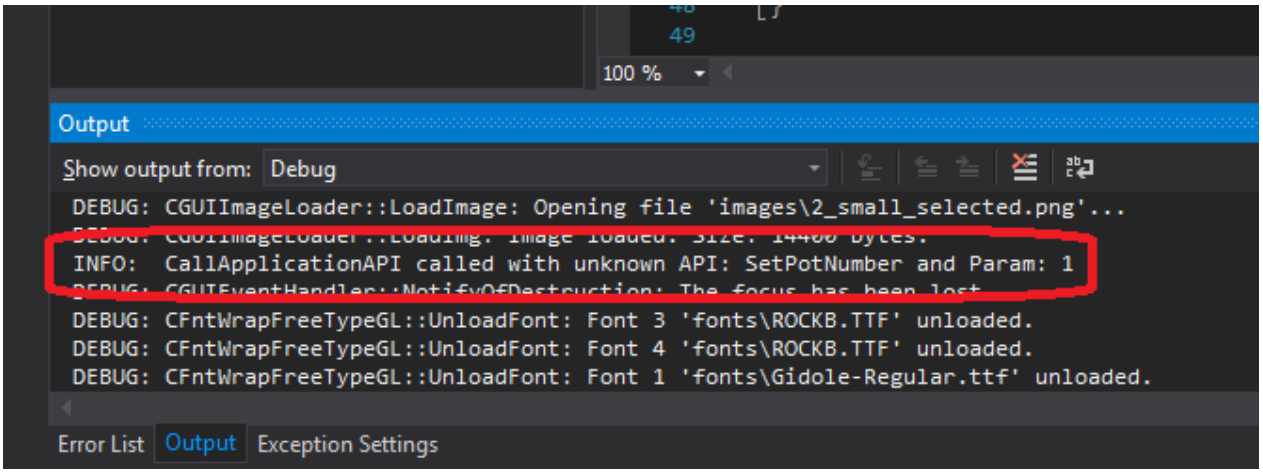


Fig. 6 CallApplicationAPI in Output window

This is the call we send from our GSE application (burner 1)!
But at the moment, it is unknown to the GSE. We will change this now.

©2017 TES Electronic Solutions GmbH

2.3.2. Add CallApplication API “SetPotNumber”

The CallApplication API command will be handled in the file “MyGUI_SR.cpp”, where you can add all the programming stuff you need to run your application the way you want it.

- Open the file “MyGUI_SR.cpp”. You will find it in the folder “*C:\Gui_SDK_VS2015\StreamRuntime\src*”.

For Microsoft VS 2015/2012 users: you will find the file in the „StreamRuntime“ project, inside the folder “Source Files.”.

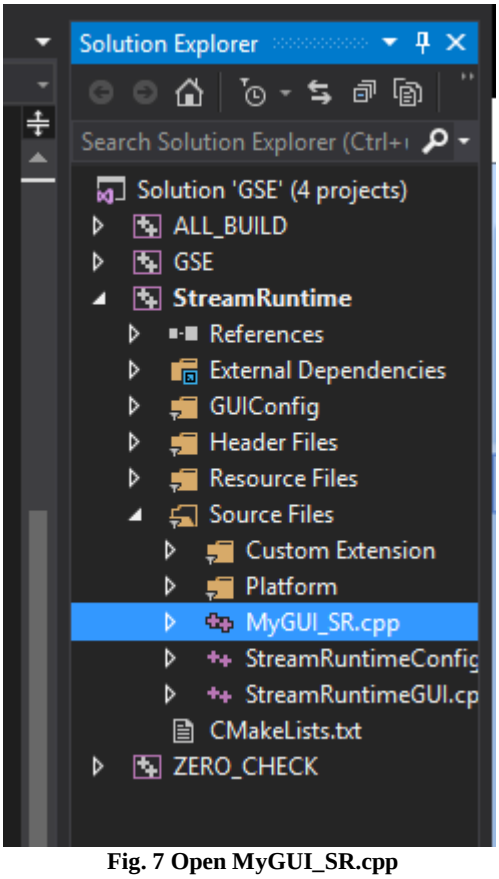


Fig. 7 Open MyGUI_SR.cpp

©2017 TES Electronic Solutions GmbH

Here you will find the function "**CallApplicationAPI**" where we have to add the call for our function which should handle the call of "SetPotNumber".

```
49
50  CMyGUI::~CMyGUI()
51  {
52      // Add application specific de-initialization here if necessary
53  }
54
55  eC_Bool CMyGUI::CallApplicationAPI( const eC_String& kAPI, const eC_String& kParam)
56  {
57      // Add application API handlers here
58      if( kAPI == "Example" )
59      {
60          GUILOGMESSAGE(" CallApplicationAPI called with API: Example \n");
61      }
62      else
63      {
64          GUILOGMESSAGE(" CallApplicationAPI called with unknown API: "+kAPI+" and Param: "+kParam+"\n");
65          return false;
66      }
67      // Successfully handled call
68      return true;
69  }
70
```

Fig. 8 Function CallApplicationAPI

- Enter the if-case for our CallApplicationAPI handling. At the moment we only want a debug message, that our CallApplicationAPI has been called.

```
55  eC_Bool CMyGUI::CallApplicationAPI( const eC_String& kAPI, const eC_String& kParam)
56  {
57      // Add application API handlers here
58      if( kAPI == "Example" )
59      {
60          GUILOGMESSAGE(" CallApplicationAPI called with API: Example \n");
61      }
62      else if (kAPI == "SetPotNumber")
63      {
64          GUILOGMESSAGE(" CallApplicationAPI called with API: SetPotNumber and Param: "+kParam+"\n");
65      }
66      else
67      {
68          GUILOGMESSAGE(" CallApplicationAPI called with unknown API: "+kAPI+" and Param: "+kParam+"\n");
69          return false;
70      }
71      // Successfully handled call
72      return true;
73  }
```

Fig. 9 Added our if-case

- Enter the following code:

```
else if (kAPI == "SetPotNumber")
{
    GUILOGMESSAGE(" CallApplicationAPI called with API: SetPotNumber and Param: " + kParam
+ "\n");
}
```


©2017 TES Electronic Solutions GmbH

- Compile GSE and start our application. Click once onto burner 1 and close the GSE.

- The output should include this line:

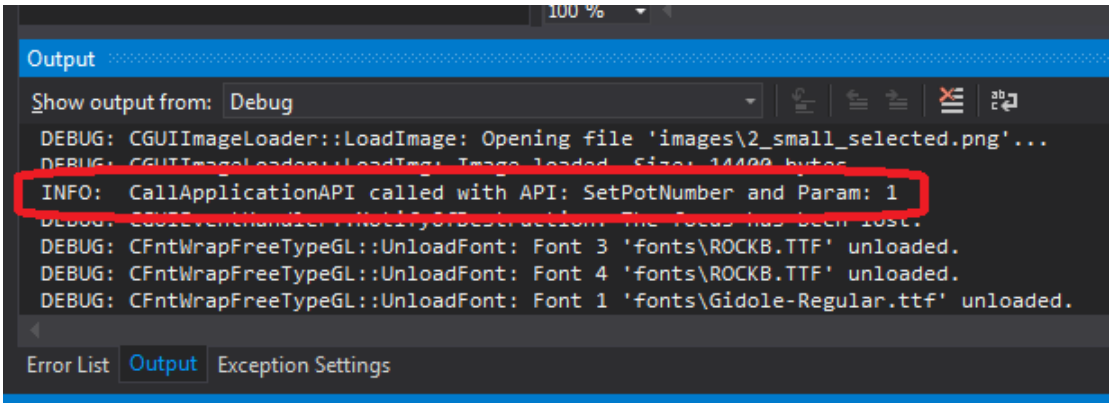


Fig. 10 our ApplicationAPI in Output

Our CallApplicationAPI should be handled in the function "**SetCookingPotNumber**". So we have to declare it in the ".h" file and add the function to the ".cpp" file.

- Open the file "MyGUI_SR.h". You will find it in the folder "*C:\Gui_SDK_VS2015\StreamRuntime\include*".

©2017 TES Electronic Solutions GmbH

For Microsoft VS 2015/2012 users: you will find the file in the „StreamRuntime“ project inside the folder “Header Files.”.

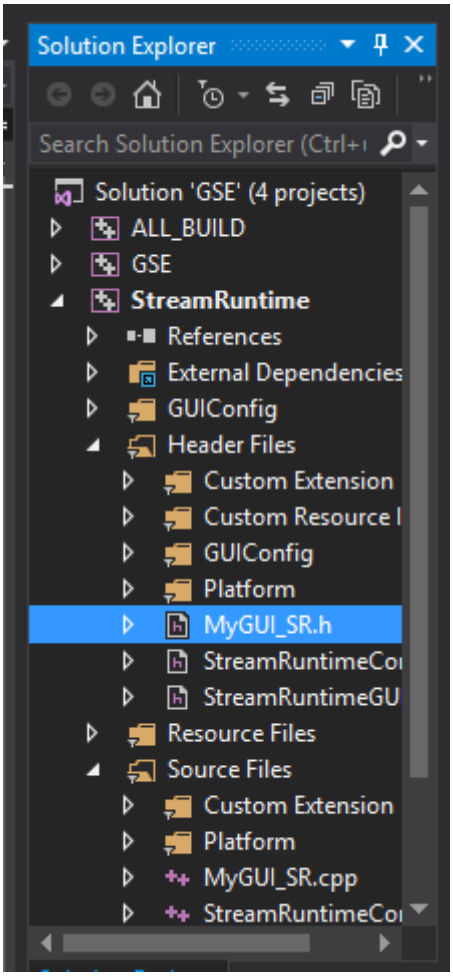


Fig. 11 Open MyGUI_SR.h

©2017 TES Electronic Solutions GmbH

In the MyGUI_SR.h file we will declare the function "**SetCookingPotNumber**" as private with one parameter.

MyGUI_SR.h* | MyGUI_SR.cpp*

StreamRuntime | CMyGUI

```
1   /*
2   * Copyright (C) TES Electronic Solutions GmbH,
3   * All Rights Reserved.
4   * Contact: info@guiliani.de
5   *
6   * This file is part of the Guiliani HMI framework
7   * for the development of graphical user interfaces on embedded systems.
8   */
9
10  #ifndef _MYGUI_SR_H_
11  #define _MYGUI_SR_H_
12
13  #include "StreamRuntimeGUI.h"
14
15  #define GETMYGUI    static_cast<CMyGUI&>(GETGUI)
16
17  // Application specific CGUI instance. Implemented by customer.
18  class CMyGUI
19      :public NStreamRuntime::CStreamRuntimeGUI
20  {
21  public:
22      CMyGUI(eC_Value x, eC_Value y, eC_Value width, eC_Value height, ObjectHandle_t eID);
23      ~CMyGUI();
24
25      /// Example implementation for simple Application <-> GUI communication
26      eC_Bool CallApplicationAPI( const eC_String& kAPI, const eC_String& kParam);
27
28  private:
29      void SetCookingPotNumber (const eC_String& kParam);
30  };
31
32  #endif //#ifndef _MYGUI_SR_H_
33
```

Fig. 12 Added SetCookingPotNumber to MyGUI_SR.h

▶ Enter the following code:

```
private:
    void SetCookingPotNumber(const eC_String& kParam);
```


©2017 TES Electronic Solutions GmbH

- Add the function "**SetCookingPotNumber**" and the call inside "**CallApplicationAPI**" to MyGUI_SR.cPP".

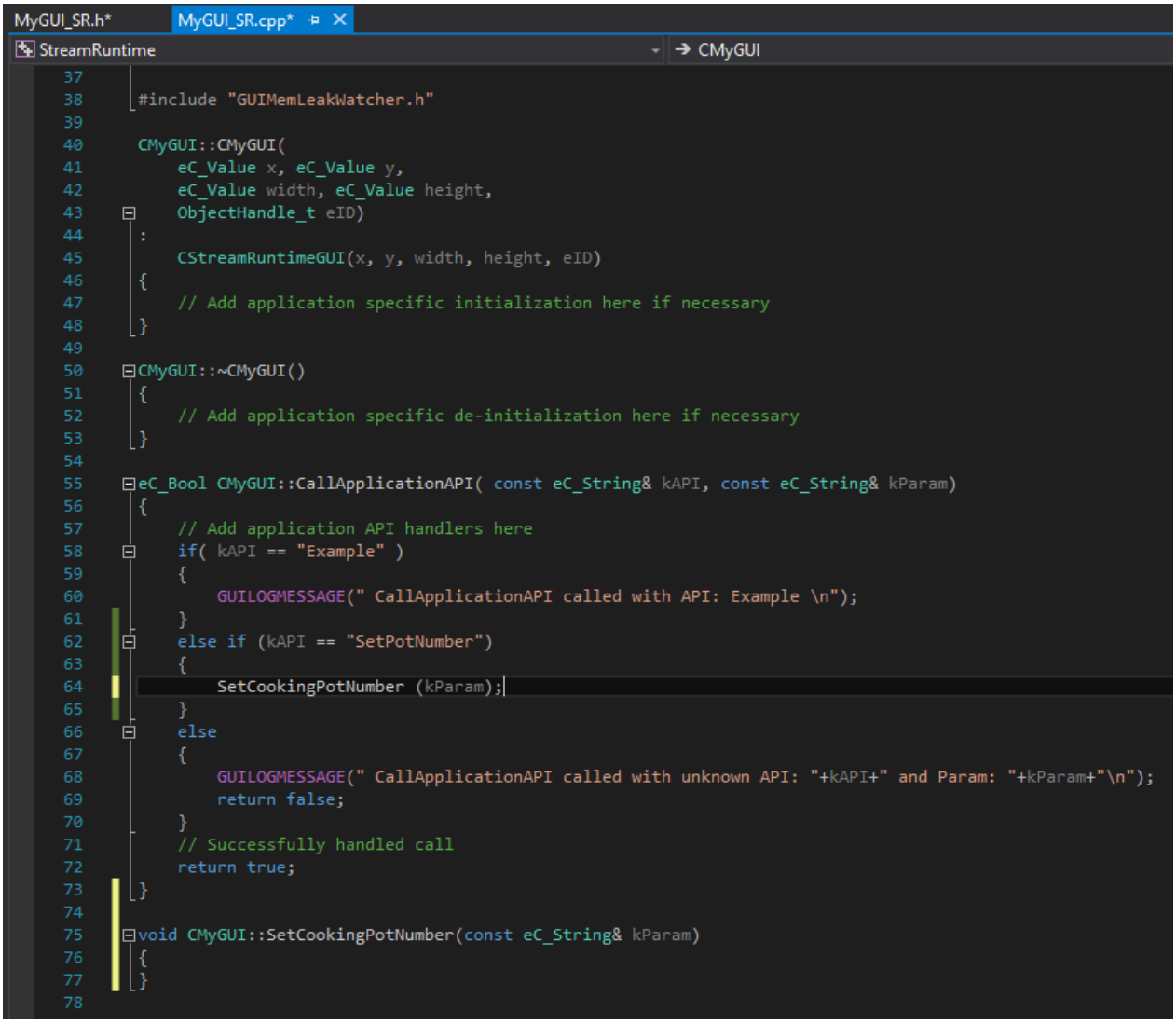


Fig. 13 Added SetCookingPotNumber to MyGUI_SR.cpp

- In the function `CallApplicationAPI` replace:

```
    GUILOGMESSAGE(" CallApplicationAPI called with API: SetPotNumber and Param: " + kParam
+ "\n");
```

- with:

```
SetCookingPotNumber (kParam);
```

- Enter the following code:

```
void CMyGUI::SetCookingPotNumber(const eC_String & kParam)
{
}
```


©2017 TES Electronic Solutions GmbH

What should happen in our function?
The text field from the pot should be updated with the given parameter.

To set a pointer to our text field, we need to know the ID of the control.
All GSE control IDs will be stored in the file named “UserObjectResource.h”. You will find the file in the folder “*C:\Gui_SDK_VS2015\StreamRuntime\include\guiconfig*”.

- Open “UserObjectResource.h” and take a look into it (you will find it in MS VS under: StreamRuntime/HeaderFile/GUI_Config).

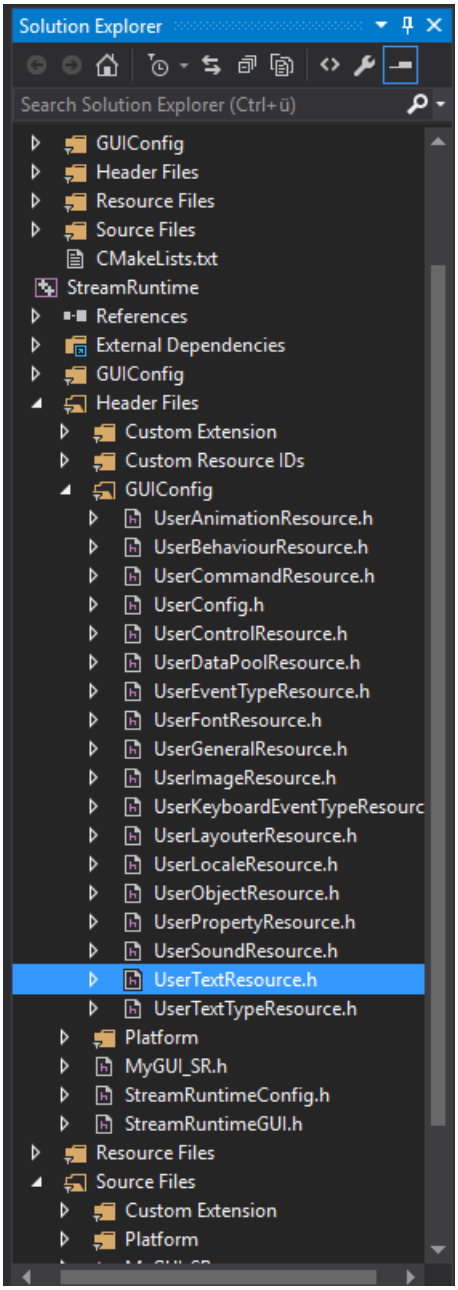


Fig. 14 Open UserObjectResource.h

©2017 TES Electronic Solutions GmbH

There are no object resources defined in this file at the moment!
That is because we stored our GSE project into the folder "*C:\step_by_step*".
So we have to copy the generated .h files from the GSE into our project.

- Open the folder "*C:\step_by_step\temp*" and select all ".h" files.

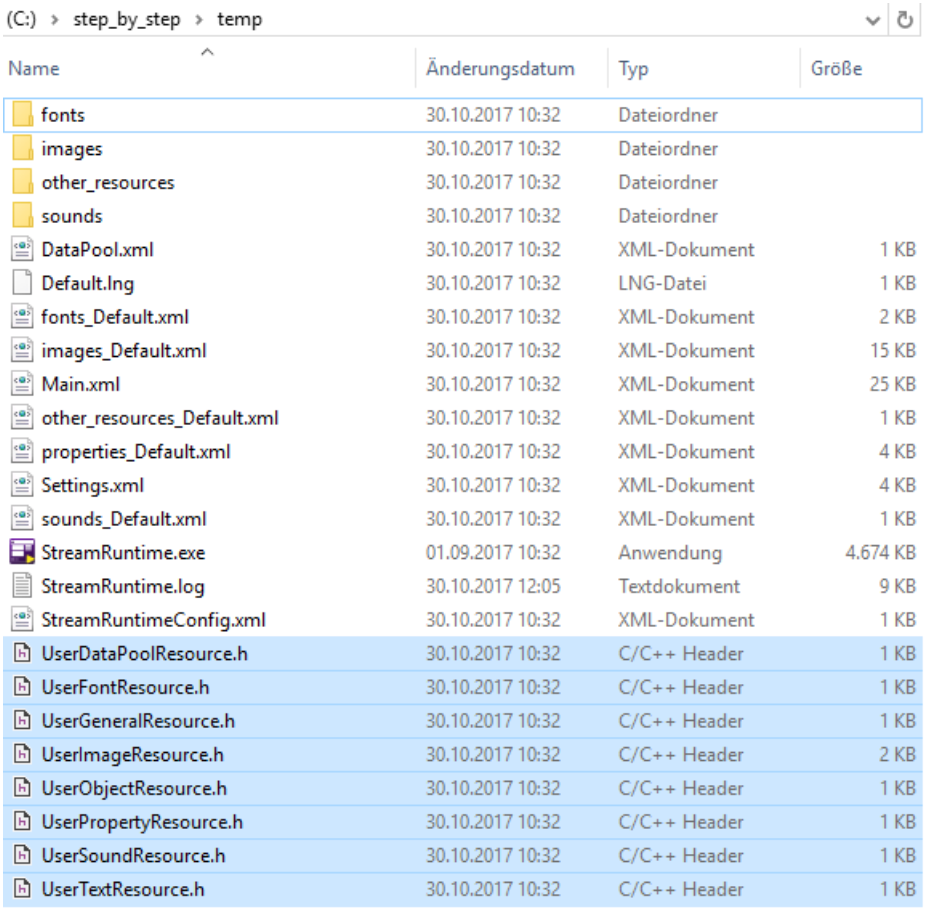


Fig. 15 Select .h files from GSE

- Copy the selected ".h" files to our GSE, into the folder "*C:\Gui_SDK_VS2015\StreamRuntime\include\guiconfig*".

©2017 TES Electronic Solutions GmbH

- Replace existing files.

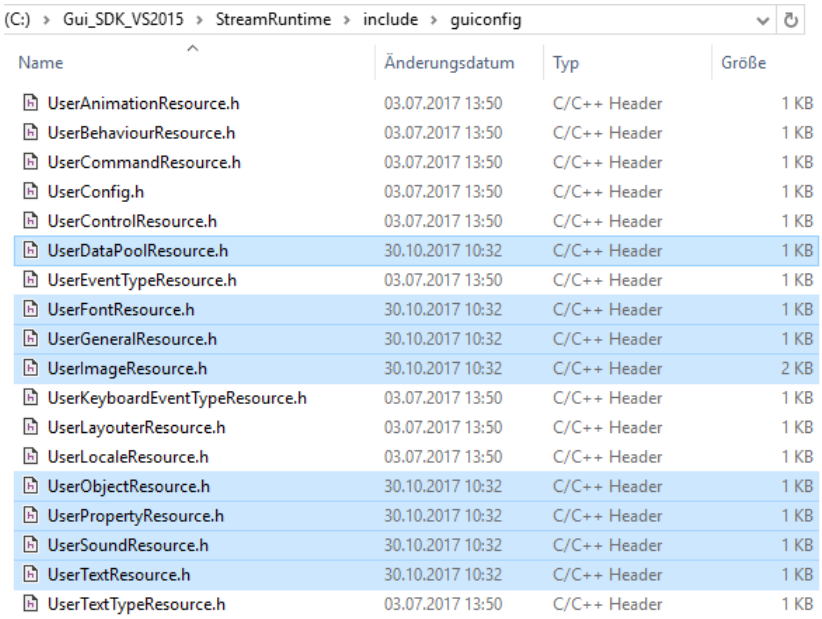
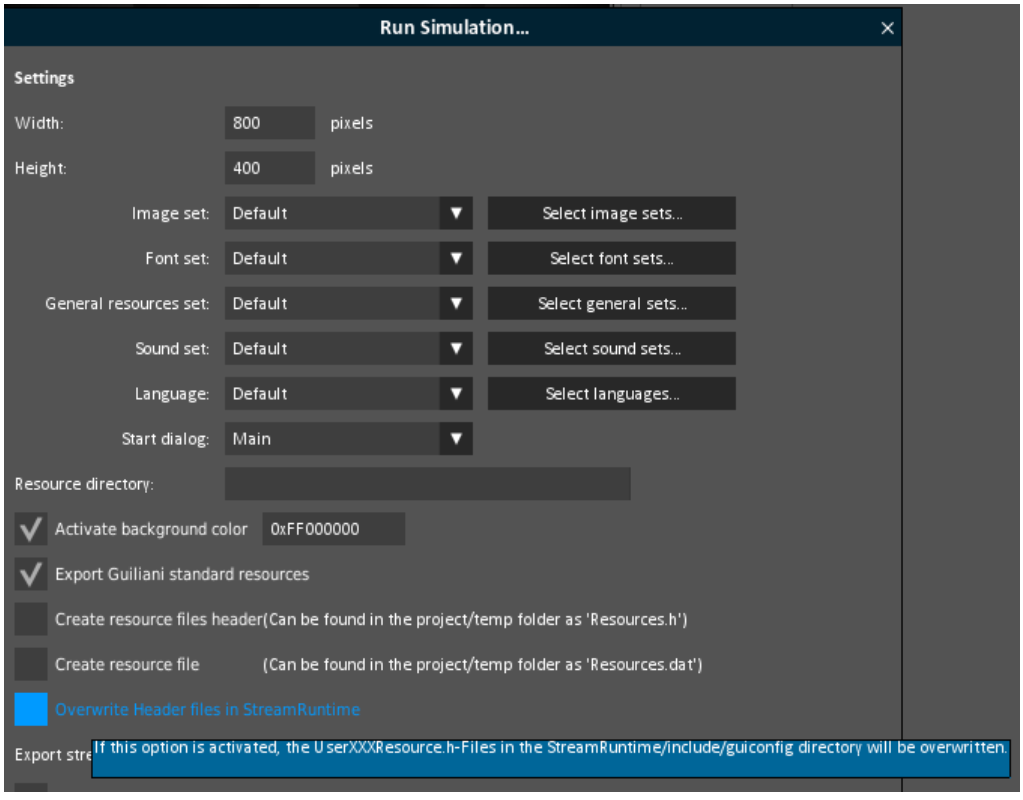


Fig. 16.h files copied to StreamRuntime

If you use the StreamRuntime folder you have specified in CMAKE, you have not to copy the files. Just mark the checkbox “Overwrite Header files in StreamRuntime” when you run the simulation.



©2017 TES Electronic Solutions GmbH

- Open your project in your IDE and look into “UserObjectResource.h” (you will find it in MS VS under: StreamRuntime/HeaderFile/GUI_Config).

You will have a list with all your objects you have defined in the step-by-step project, yet.

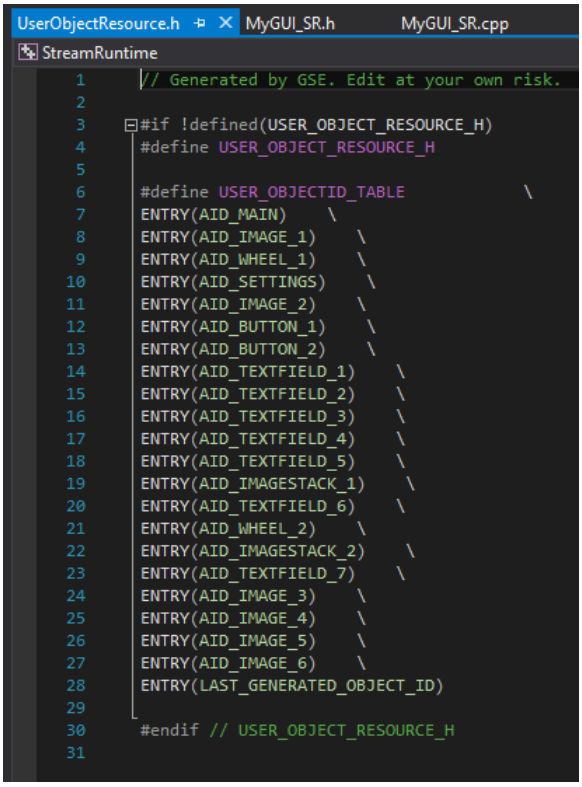


Fig. 17. UserObjectResource .h

Now we are able to address the objects we need.

©2017 TES Electronic Solutions GmbH

We want to change the text field inside the pot which is **AID_TEXTFIELD_1**.

The first step is to tell the application that we want to use the text field. Therefore we need to include the definition of a text field, which will be done in “GUITextField”.

- Add an #include for “GUITextField” at the beginning of the file “MyGUI_SR.cpp”.

Be sure not to add a INCLUDE below the existing #include "GUIMemLeakWatcher.h". This include has to be the last include!

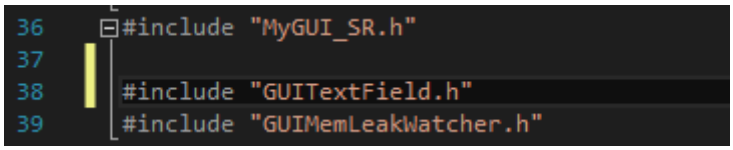


Fig. 18. Include GUITextField.h

- Enter the following code:

```
#include "GUITextField.h"
```


©2017 TES Electronic Solutions GmbH

What do we have to program in our SetCoockingPotNumber function?

```
76  void CMyGUI::SetCookingPotNumber(const eC_String& kParam)
77  {
78      if ((kParam >= "1") && (kParam <= "4"))
79      {
80          CGUITextField* pkTextField = static_cast<CGUITextField*>(GETGUI, GetObjectByID(AID_TEXTFIELD_1));
81
82          if (pkTextField)
83          {
84              pkTextField->SetValue(kParam);
85          }
86      }
87  }
```

Fig. 19. Add TextField toSetCoockingPotNumber function

▶ Enter the following code between the two brackets in our function `SetCookingPotNumber`:

```
if ((kParam >= "1") && (kParam <= "4"))
{
    CGUITextField* pkTextField = static_cast<CGUITextField*>(GETGUI, GetObjectByID
(AID_TEXTFIELD_1));

    if (pkTextField)
    {
        pkTextField->SetValue(kParam);
    }
}
```

Let us go through the function line by line:
78 - When do we have to do something? Only when we get a number from 1 to 4 (our burners)
80 - Declaration of the pointer of the guilliani CGUITextField class and getting the pointer to our text field object
82 - Check if we have a valid pointer
84 - Set the given parameter into the text field's label (its value)

That was it.

▶ Save your application and start it out of your IDE.

Now the pot number changes according to the chosen burner.
And this is the goal of this documentation.

©2017 TES Electronic Solutions GmbH

2.4. How to continue?

2.4.1. Sample solution

Now it should be no problem to use an CallApplication API in GSE anymore.
If you encountered problems or wish to have the solution without creating the project on your own, we have added the sample solution into the folder called "*How to 6 - sample solution*" inside the documentation folder. Here you will find the GSE Project (step_by_step.gpr).

For this HowTo there are no additional resources.

For Windows user inside the folder "temp" there is an executable StreamRuntime.exe.

2.4.2. Continuing How Tos

You will find an overview of continuing HowTos in the document "How to 0 - an overview of building GSE projects".

Don't forget to visit our homepage www.guiliani.de to get more information, demos, help, videos and the latest news about guiliani and GSE.

©2017 TES Electronic Solutions GmbH

3. Index

A

Add CallApplication API “SetPotNumber” 15
Adding a command to burner 1 8
Adding a command to burner 2 11
Adding CallApplication API 8
Adding CallApplication API in source code 12
Assumed knowledge 5

C

CallApplication API
  Add “SetPotNumber” 15
  Adding 8
  Adding in source code 12
  Function 16
  Using 8
Command
  Adding 8, 11
Continuing How Tos 27

D

Documentation conventions 5

F

Function CallApplication API 16

G

generated .h files 22
guiconfig 22
GUITextField 25

H

How to continue? 27

I

ID of the control 21
include 25

L

Load project 8
Loading our project to the IDE 12

M

MyGUI_SR.cpp 15
MyGUI_SR.h 17

O

object resources 22

P

Prerequisites 5
program SetCoockingPotNumber function 26
Project
  Load 8

S

Sample solution 27
Short cuts 6

T

temp folder 22

U

UserObjectResource.h 21
Using a CallApplication API 8

©2017 TES Electronic Solutions GmbH