

G-Core SDK Documentation

General guide for development, testing and certification

Author: Ewenz Michael

Content

1 Introduction	4
1.1 Terms and abbreviations	4
2 G-Core Client SDK.....	4
2.1 Supported development platforms	4
2.2 Files and directory structure	4
2.3 Setting up a virtual test environment.....	5
2.3.1 Assign local policy “Lock pages in memory”	5
2.3.2 Setup the Microsoft SQL Server 2014	6
2.3.3 Start the G-Core Server	6
2.3.4 Import the test setup	7
2.3.5 Media Channel Simulator (MCS plugin)	7
2.3.6 View live/recorded video and backup video in G-View	7
2.3.7 Monitor and testtool GngPLCSimulator	8
2.4 Overview of the interfaces in the ClientSDK	8
2.4.1 Introduction	8
2.4.2 Building blocks of functionality	8
2.5 Guidelines and hints	9
2.5.1 Introduction	9
2.5.2 General hints.....	9
2.5.3 Hints for C++ development.....	9
2.5.4 Hints for .Net development.....	9
2.5.5 .Net wrapper dependancies	10
2.5.6 Working with handles and instances (C++)	11
2.5.7 Interaction between DBI and MediaPlayer	12
2.5.8 Enumeration of setup data	16
2.5.9 PLC, actions and events	17
2.5.10 Media channel IDs	19
2.5.11 Handling connection collapses	19
2.6 Using the SDK with .NET.....	19
2.6.1 Deploying a custom solution based on the .NET wrapper	19
3 Request of a G-Cert.....	20
3.1 G-Cert Introduction	20
3.2 Creating a self-signed certificate.....	20
3.3 Sign your application	20

3.3.1 Using the developer command prompt.....	21
3.3.2 Sign automatically with each Build	21
3.3.3 Request your G-Cert	23

Tables

Table 1: Folder contents	5
Table 2: Methods of a server object instance	12
Table 3: Methods of a viewer object instance.....	13
Table 4: Methods of a PLC object instance	17

Figures

Figure 1: Setting the local policy "Lock pages in memory"	6
Figure 2: G-Core server in console mode	7
Figure 3: Open project properties.....	21
Figure 4: Access Post-build event command line	22
Figure 5: Login to SDK website.....	23
Figure 6: My Solutions on the SDK website	24
Figure 7: Add a new solution.....	24
Figure 8: Register a new solution.....	25
Figure 9: G-Cert Generator tool	26
Figure 10: Upload the signed executable or dll	26
Figure 11: Download the certificate	27

Listings

Listing 1: Working with handles.....	12
Listing 2: Example for using a viewer	13
Listing 3: Example of using the OffscreenViewer	15
Listing 4: Enumerating G-Core settings.....	16
Listing 5: Example for using the PLC	18

GEUTEBRÜCK

1 Introduction

The G-Core SDK consists of a collection of free software interfaces for the GEUTEBRÜCK DVR G-Scope. It can be used to integrate these devices in custom applications and for linking not yet supported peripherals as well.

In G-Core systems text based messages (called “actions”) are used to communicate between the G-Core server and any client application. All available actions can be divided into three groups: Notification actions (for example “User Login”), command actions (for example “Viewer connect live”) and logical actions (these actions are not directly created by the G- Core server and they don't directly result in any reaction in the G-Core server, for example “Custom action”). All actions are grouped into different categories. The category “Viewer actions” e.g. contains all actions that are relevant for remote controlling G-View. To get notifications about G-View activities, one of the options “Send notification actions” in the profile manager of G-View has to be set. All possible notification actions are collected in the action category “Viewer notifications”.

More detailed information about all available actions can be found in the topic “Action documentation” (especially Viewer actions and Viewer notifications). Please be aware of the fact that G-View is working in an asynchronous mode. If a custom application sends an action, that depends on the result of the previous sent action there may be the need for inserting a pause time before sending the second action (e.g. send action “Viewer connect live”, wait one second, send action “Viewer print picture”). G-View does not operate an input queue for remote control actions.

1.1 Terms and abbreviations

DVR	Digital video recorder
G-Scope	Geutebrück DVR
G-Core	Server software running on a G-Scope
G-Set	Setup user interface for G-Core
G-View	User interface for displaying live and recorded videos
G-PLC	Geutebrück logic for client/server communication using actions and events
SDK	Software development kit
Action	Text base G-PLC command
Event	Action triggered message to be stored in the database. Events can be started and stopped to mark a timerange.
DBI	Database interface
PLC	Programmable logic controller

2 G-Core Client SDK

2.1 Supported development platforms

- Microsoft Visual Studio 2013 with C++ and MFC
- Microsoft Visual Studio 2013 / 2015 / 2017 / 2019 with .Net Framework 4.5, 4.5.1 and 4.6.1
- OS Windows 8.1, Windows 10, Windows Server 2012 R2 and higher

2.2 Files and directory structure

During the installation the system variable %GNGSDKPATH% will be set to the root directory of the SDK (default: C:/G-Core-SDK). This path is used in the SDK examples.

Folder	Description
Bin\x64	64 bit libraries, G-Core demo binaries and destination for 64 bit compilations of the examples
Bin\x32	32 bit libraries and destination for 32 bit compilations of the examples
Documentation	SDK and G-Core action documentation
Examples\VS2013CPP	Examples for MS Visual Studio 2013 C++ compiler
Examples\VS2013NET	Examples for MS Visual Studio 2013 C# .Net compiler
Examples\VS2015CPP	Examples for MS Visual Studio 2015 C++ compiler
Examples\VS2015NET	Examples for MS Visual Studio 2015 C# .Net compiler
INCLUDE	C++ header and special source files
Lib	Libraries for C++ projects
PluginSDK\Documentation	Documentation of the Plugin SDK
PluginSDK\Examples	Examples for developing an own plug-in for G-Core
PluginSDK\INCLUDE	Needed headers and sources for an own G-Core plug-in
GBF	Geutebrück backup file – A proprietary file format containing compressed video of one or multiple channels

Table 1: Folder contents

2.3 Setting up a virtual test environment

All required components for setting up a virtual G-Core server are included in the SDK. So an independent development of custom solutions can be archived without any special hardware required.

After starting up the G-Core server the software can be used with full functionality for six hours. Exceeding this time the server stops automatically but can be restarted for the next six hours.

2.3.1 Assign local policy "Lock pages in memory"

To run G-Core server on your local machine, the local policy "Lock pages in memory" needs to be assigned to the user account under which G-Core server should work. Please open the "Local Security Policy" dialog in the control panel "Administrative Tools (or run "gpedit.msc") as shown in Figure 1.

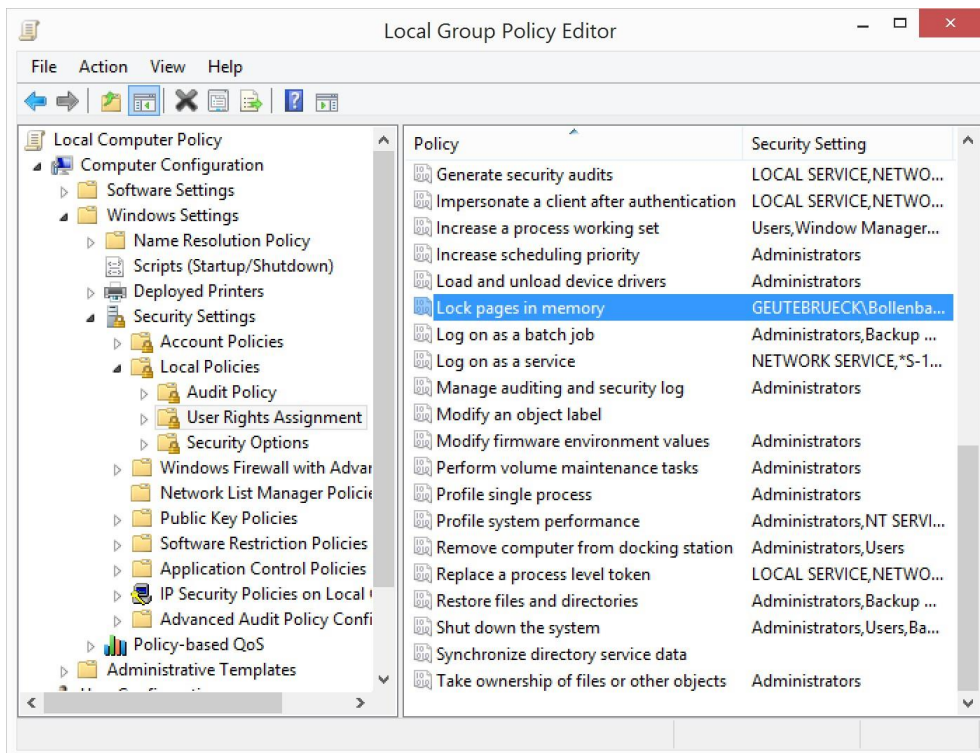


Figure 1: Setting the local policy "Lock pages in memory"



The user has to be member of the local Administrator group.
The user has to logout and login again to let the setting take effect.

2.3.2 Setup the Microsoft SQL Server 2014

It is highly recommended to use the Microsoft SQL Server 2014 installer provided by GEUTEBRÜCK (G-Core_ - SQLServerInstaller.exe). This installer generates an instance with the name "GCoreSQL" and sets connection parameters.

2.3.3 Start the G-Core Server

The G-Core Server can be started using the "GCoreServer.bat" stored in the root folder of the G-Core SDK or using the console and the command "gcoreserver.exe console demo". Figure 2 shows the G-Core server started in console mode.

After starting the G-Core server the database will be generated automatically. If the G-Core Server doesn't respond anymore, please restart the server once.

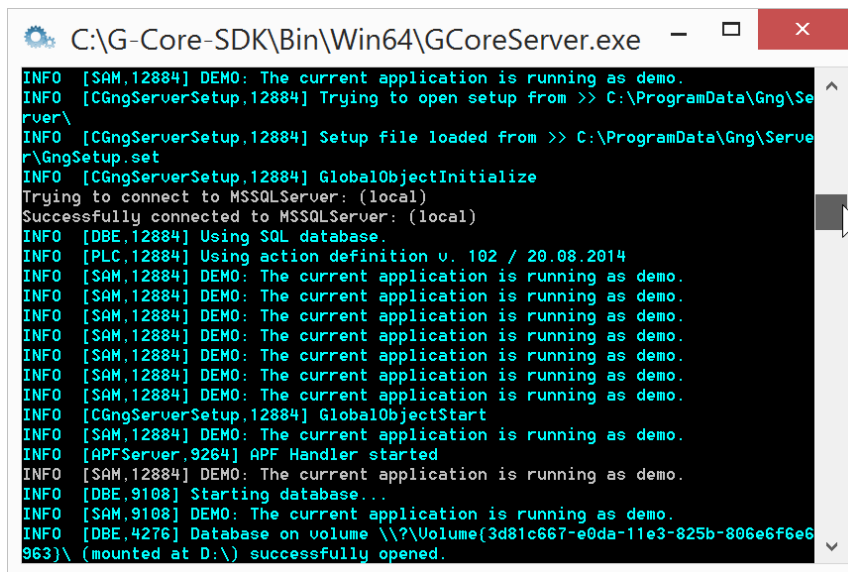


Figure 2: G-Core server in console mode

2.3.4 Import the test setup

By default the setup of the G-Core Server is empty. G-Set (GSet.exe) is used to alter the configuration of the G-Core Server. The default username is “sysadmin” and the password “masterkey”.



A test setup (“GCoreSDK.set”) is located in the root folder of the SDK and can be imported.



The setup has to be sent to the server by tapping on the  button!

2.3.5 Media Channel Simulator (MCS plugin)

To provide a test environment with full functionality the G-Core media plugin “MCS” (Media Channel Simulator) is used. It simulates real video media channels by channeling test pictures into the G-Core Server. 16 media channels can be used as live channels or can be recorded into the test database. Furthermore, the channels create messages (actions) that allow using them as base for developing video analysis software.

In the test setup the MCS plugin is already configured.

2.3.6 View live/recorded video and backup video in G-View

The default viewer for live and recorded video in G-Core is G-View.

G-View can be connected to G-Core Server using username “sysadmin” and password “masterkey”. A media channel can be watched by simply dragging it to one viewer of G-View.

Backup video can be watched by opening a “.gbf” file with G-View. The media channels included in that file can also be watched by dragging them to one viewer.



A sample backup video is contained in the G-Core SDK and stored in the bin folder (DatabaseBackup.gpf).

2.3.7 Monitor and testtool GngPLCSimulator

The GngPLCSimulator (GngPLCSimulator.exe) serves as a monitoring tool for all messages (actions) and events that are transported inside the complete system. Furthermore actions can be triggered and events can be started and stopped. After building up a connection to the local server all action traffic is displayed in a list. This tool is extremely helpful for testing of custom applications based on the SDK and for analyzing the message flow in the complete system.

2.4 Overview of the interfaces in the ClientSDK

2.4.1 Introduction

This document gives a short overview over the different interfaces provided by the SDK.



All interfaces include class declarations to access the exported functions of the dynamic link libraries. To use them in C++, the matching cpp files and the lib files corresponding to the libraries have to be added to the custom project.

2.4.2 Building blocks of functionality

- DBI (DataBase Interface)
 - Low level server and database interface
 - Connection handling, GBF access, raw database access (no video display!), media export functionality, backup functions, access to raw live media (no video display!), setup data access
 - Supports basic functionality for building blocks “PLC” and “MediaPlayer”
 - Main binary file: GngDBI.DLL
 - Main include files (C++): GngDBI.h, GngDBI.cpp
 - C# wrapper files:
 - GngDBINET_x.y.z.dll (for dedicated .Net framework version)
 - GngDBINET_SDK.dll (for the latest supported .Net Framework version)
- PLC (Process Logic Controller)
 - Complex notification, action and event processing
 - Listens to, dispatches, creates and sends actions
 - Listens to, starts and stops events and system notifications
 - Allows controlling and monitoring the system
 - Main binary file: GngActions.DLL
 - Main include files (C++): GngActions.h
 - C# wrapper files:
 - G-ActionsNET_x.y.z.dll (for dedicated .Net framework version)
 - G-ActionsNET_SDK.dll (for the latest supported .Net Framework version)
- MediaPlayer
 - High level server and database interface including media presentation
 - Display video, play audio (live and backup)
 - Integrated export functionality (GBF, MPEG, Single picture)
 - Search media data by time or corresponding to event data
 - Main binary file: GngMediaPlayer.DLL
 - Main include files (C++): GngMediaPlayer.h, GngMediaPlayer.cpp
 - C# wrapper files:
 - GngMediaPlayerNET_x.y.z.dll (for dedicated .Net framework version)
 - GngMediaPlayerNET_SDK.dll (for the latest supported .Net Framework version)

- OffscreenViewer
 - Part of building block “MediaPlayer”
 - Same functionality as MediaPlayer, but: no rendering, only decompressing
 - Class TGngOffscreenViewer can be used analogous to TGngViewer
- Media plugin (G-Core server plugins)
 - G-Core server plugins allow integrating custom peripherals in G-Core systems
 - Channeling of video and/or audio media into the server
 - Including full access to PLC
 - Plugins run as In-Process-DLLs in G-Core Server
- G-View data filter plugin
 - G-View plugins allow integrating custom data filter frontends in G-View software
 - Plugins run as In-Process-DLLs in G-View software
- G-View data presentation plugin
 - G-View plugins allow customized presentation of event data in G-View software, especially of event data presented in viewed pictures
 - Plugins run as In-Process-DLLs in G-View software

2.5 Guidelines and hints

2.5.1 Introduction

It is recommended to be familiar with the G-Core system and the possibilities of modern video surveillance systems and video management systems. Before starting programming the developer should know the basics of video formats, video compression, G-Core events, G-Core actions and the principles of a client - server network communication. The following sections supports with some suggestions and hints about using the SDK interfaces.

2.5.2 General hints

If the application needs to listen to events and actions please use the GngPLCSimulator. This software can send actions and start/stop events used by the application. Before you start you should do some tests with a real G-Core device or with the virtual test environment provided with the SDK. Create some events and actions, start them with GngPLCSimulator. Typing Ctrl-Alt-U in G-Set will offer the possibility to open DBITest to discover the database structure and to evaluate and test select statements against the database. Additionally this tool offers the possibility to start the registry editor to evaluate the internal structure of the G-Core Setup.

2.5.3 Hints for C++ development

All objects that are created inside of DLLs should be deleted after use. The objects themselves should always offer a Destroy() or Free() method for releasing the used memory. Callback functions, which are called out of the SDK DLLs, are called from threads, which were created inside the DLLs. Variables and pointers that are passed as arguments of the callback must not be used outside the callback context. They are only valid for the duration of the callback call. Structures that are used as arguments for SDK functions should always be initialized by the function memset(). After setting all the structure elements to zero, the size or structsize element has to be initialized with the sizeof() function. MPEG-2 files that are created by SDK functions can possibly not be played with the windows media player. The reason is a missing MPEG-2 decoder. We recommend using VLC Media Player software.

2.5.4 Hints for .Net development

Some objects provided by the libraries must be disposed explicitly after use to release occupied memory.

All resources used by video export must be disposed in the very same thread as they are instantiated. Otherwise the SDK application may hang or crash. This is because of the MFC usage inside the library.

If the SDK application shall be developed against the most recent .Net Framework which is supported by the SDK, it is highly recommended to reference the ...NET_SDK.dll versions. This enables the developer to easily update the application to a higher .Net Framework version if supported by the SDK without changing the references.



For the currently supported .Net Framework versions please refer to 2.1

2.5.5 .Net wrapper dependancies

If the SDK application shall be developed in C# .Net, the particular wrapper have to be referenced in the projects. The wrapper libraries are dependant of several unmanaged libraries which must be present in the execution path of the application. The following list shows the basic dependancies:

- GngDBINET_SDK.dll
 - GngActions.dll
 - GngDBI.dll
 - GngMediaProcessor.dll
 - GngActions.dll
- G-ActionsNET_SDK.dll
 - no dependancies of other Gng libraries
- GngMediaPlayerNET_SDK.dll
 - GngDBI.dll
 - GngMediaProcessor.dll
 - GngActions.dll
 - GngMediaPlayer.dll
 - GngMediaProcessor.dll
 - GngDBI.dll
 - GngActions.dll
 - GscDBI.dll
 - GngMediaProcessor.dll
 - GngMediaRenderer.dll
 - GngVCA.dll
 - GngVCAPrimitives.dll
 - GngMediaProcessor.dll
 - GngActions.dll
 - GscDBI.dll
- GngExceptionsNET_SDK.dll
 - no dependancies of other Gng libraries

Even if no dependency is listed, there may be dynamically loaded libraries which cannot be detected by DependencyWalker.

The following files should always be present in the executable path of the SDK application:

- G-ActionsNET_SDK.dll
- GLibClassLib.dll
- GngActions.dll

- GngActionsNET_SDK.dll
- GngDBI.dll
- GngDBINET_SDK.dll
- GngExceptionsNET_SDK.dll
- GngMediaPlayer.dll
- GngMediaPlayerNET_SDK.dll
- GngMediaProcessor.dll
- GscDBI.dll
- GscActions.dll
- Libmfxsw64.dll / libmfxsw32.dll
- ffmpeg.exe (if video export is needed in mpeg format)

2.5.6 Working with handles and instances (C++)

Integral part of the SDK are units that give the user a comfortable access to the plain functions of the DLL, e.g. GngDBI.h/cpp. In these units classes encapsulate access to instances of objects which are created inside the DLL. To have access from outside the DLL (custom application) to the inside residing instances, handles are used. The units have to be added to the project respectively to the solution to avoid linker errors. After the work with instances is finished, the instances have to be deleted by calling their destroy() or free() method. Otherwise there will be memory leaks.

Using the plain exported functions of the DLL is not recommended. To get access to full functionality one should use the units instead (h/cpp files).

Listing 1 shows how to cope with handles.

```

1  // define a handle to a server object
2  HGngServer MyServer;
3
4  // create a server object instance inside the DLL and
5  // get a handle to it
6  MyServer = DBICreateRemoteserver( );
7  ...
8
9  // work with the object instance with the help of the handle
10 MyServer->Connect( );
11 ...
12
13 ...
14 // define a handle to a PLC object
15 HGngPLC PLC;
16
17 // create a PLC object instance inside the DLL and
18 // get a handle to it
19 PLC = MyServer.CreatePLC( );
20 ...
21
22 // work with the object instance with the help of the handle
23 PLC->OpenPushCallback (...);
24 ...
25
26 // destroy PLC object
27 PLC->Destroy( );
28 ...
29
30 // destroy server object
31 MyServer->Destroy( );

```

Listing 1: Working with handles

2.5.7 Interaction between DBI and MediaPlayer

The DBI interface gives access to G-Core Server functionality. After creating an instance with the function `DBICreateRemoteserver()` a connection to the server can be established by calling the method `Connect()` of the server object instance.

Table 2 shows methods of a server object instance which can be called to get access to different kinds of functions (not a complete list).

Method	Function
<code>CreateDataSet()</code>	Fetch data from server database
<code>CreateDataPacket()</code>	
<code>CreateLiveStream()</code>	Fetch live data from server
<code>CreatRegistry()</code>	Fetch setup data from server (media channel information, event information, ...)
<code>CreatePLC()</code>	Listen to, create and send actions

Table 2: Methods of a server object instance

The MediaPlayer interface offers simple to use objects to display live and recorded video in windows controls. A viewer object instance needs to be created by calling `GMPCreateViewer()`. The viewer requires a handle for a Windows control and a handle for a server object instance. It automatically retrieves and decompresses data and displays videos in the linked Windows control. Table 3 shows the methods of a viewer object instance which can be called to get access to different kinds of functions (not a complete list).

Method	Function
<code>ConnectDB()</code>	Fetch video data from the database and display it in any play mode required. Filter and search criteria can optionally be defined.
<code>SetPlayMode(...)</code>	Display pictures in the particular play mode (live, recorded, event related, ...)

Table 3: Methods of a viewer object instance

```

1  // define a handle to a viewer object
2  HGngViewer MyViewer ;
3
4  // create a viewer object instance inside the DLL and
5  // get a handle to it
6  MyViewer = GMPCreateViewer ( WindowHandle , . . . ) ;
7
8  // define a structure with data needed to link
9  // the viewer to a media channel in the server
10 TMPConnectData MyViewerConnectData ;
11
12 // handle to the server object instance
13 MyViewerConnectData.Connection = MyServer ;
14 MyViewerConnectData.Server Type = ctGng Server ;
15 MyViewerConnectData.MediaType = mt Server ;
16
17 // ID o f the media channel that should be displayed
18 MyViewerConnectData.MediaChID = ...
19
20 // link the viewer to a media channel and display live data
21 MyViewer->ConnectDB(MyViewerConnectData, pmPlayStream, ...);
22
23 // destroy viewer object
24 MyViewer->Destroy ( ) ;

```

Listing 2: Example for using a viewer

Beside the viewer object class there is another class in the MediaPlayer interface: The offscreen viewer object class. The offscreen viewer is used if the media should be decompressed but not displayed using the viewer object. An instance can be created with the function `GMPCreateOffscreenViewer()`. The offscreen viewer object instance provides nearly the same functionality as the viewer object class does. The video footage is not rendered in a window, it is decompressed in a special `DecompBuffer` object instance. After the decompression is done inside the offscreen viewer, the hosting application can be notified with the help of a callback function. Inside the callback the decompressed image can be accessed.

GEUTEBRÜCK

The `DecompBuffer` class encapsulates special functions for effective decompressing. So it is recommended to use it. You can create an instance of the buffer by calling the function `GMPCreateDecompBuffer()`. The instance can be used for as many decompressions as needed. The method `GetBufPointer()` gives access to the raw picture data inside the buffer.

GEUTEBRÜCK

```
1 // define a handle to a DecompBuffer object
2 HGngDecompBuffer MyDecompBuffer;
3
4 // createa DecompBuffer object instance inside the DLL and
5 // get a handle to it
6 MyDecompBuffer = GMPCreateDecompBuffer( ) ;
7
8 // define a handle to a offscreen viewer object
9 HGngViewer MyOffscreenViewer;
10
11 // create an offscreen viewer object instance inside the DLL and
12 // get a handle to it
13 MyOffscreenViewer = GMPCreateOffscreenViewer(MyDecompBuffer);
14
15 // set callback of the offscreen viewer object
16 MyOffscreenViewer.SetNewOffscreenImageCallBack(NewOffscreenImageCallback);
17
18 // define a structure with data needed to link
19 // the offscreen viewer to a media channel in the server
20 TMPConnectData MyOffscreenViewerConnectData;
21
22 // handle to the server object instance
23 MyOffscreenViewerConnectData.Connection = MyServer;
24 MyOffscreenViewerConnectData.ServerType = ctGngServer;
25 MyOffscreenViewerConnectData.MediaType = mtServer ;
26
27 // ID of the media channel that should be decompressed
28 MyOffscreenViewerConnectData.MediaChID = ...
29
30 // link the offscreen viewer to a media channel and decompress live data
31 MyOffscreenViewer->ConnectDB(MyOffscreenViewerConnectData,pmPlayStream,...);
32 ...
33
34 // destroy offscreen viewer object
35 MyOffscreenViewer->Destroy();
36
37 // destroy DecompBuffer object
38 MyDecompBuffer->Destroy();
39 ...
40
41 // callback function, that is called after images have been decompressed
42 ...
43
44 // get a raw pointer to the picture in the DecompBuffer
45 // object
46 MyDecompBuffer->GetBufPointer(BufferPointer, ...);
47
48 // copy the picture into a windows bitmap resource
49 // for example
50 SetDIBits(..., BitmapHandle, ..., BufferPointer, ..., DIB_RGB_COLORS);
51 ...
```

Listing 3: Example of using the OffscreenViewer

2.5.8 Enumeration of setup data

G-Core Server resources can be enumerated by custom applications. The setup object, which can be instantiated by calling the server method `CreateRegistry()`, offers functionality for this.

Enumeration of resources normally is done in four steps:

1. Define an array of type `GngSetupReadRequest` with the only element `"/"`. This causes the method `ReadNodes()` to transfer the whole setup from the server to the custom application.
2. Call the method `ReadNodes()` of the setup object to get the whole setup from the server.
3. Call one of the `Get...()` methods of the setup object to get an array of GUIDs representing the list of resources. There are different `Get...()` methods, e. g. `GetMediaChannels()` or `GetEvents()`.
4. Use the GUID array to receive the resources data by calling `Get...Settings()` methods, e. g. `GetMediaChannelSettings()` or `GetEventSettings()`.

```
1 // connect to the server
2 MyServer->Connect();
3 ...
4 // define a handle to a setup object
5 HGngRegistry MySetup ;
6 // create a setup object instance inside the DLL and
7 // get a handle to it
8 MySetup = MyServer->CreateRegistry();
9
10 // define a array for the setup read request
11 GngSetupReadRequest SetupReadRequest[1];
12 SetupReadRequest[0].NodeName = "/" ;
13
14 // read the setup data from the server
15 MySetup->ReadNodes(&SetupReadRequest, ...);
16
17 // define a GUID array for the GUIDs of the
18 // existing media channels
19 GuidDynArray MediaChannels;
20
21 // get the GUID array out of the setup data
22 MySetup->GetMediaChannels(MediaChannels);
23
24 // get the data of each single media channel
25 foreach MediaChannelGUID in MediaChannels
26 MySetup->GetMediaChannelSettings(MediaChannelGUID,
27 MediaChannelID,
28 GlobalNumber,
29 ...);
30 ...
31
```

Listing 4: Enumerating G-Core settings



The Media channels can be enumerated by using the global function `MPQueryMediaChannelList()` of the `MediaPlayer` interface as well.

2.5.9 PLC, actions and events

The PLC (Process Logic Control) object supports you with functionality for handling notifications, actions and events. The method `CreatePLC()` of the server object class creates a handle to a PLC object inside the DBI DLL.

Table 4 shows the methods of a PLC object instance which can be called to get access to different kinds of functions (not a complete list).

Method	Function
<code>SendAction()</code>	Send an action to the connected server
<code>StartEvent()</code>	Start an event of the connected server
<code>SubscribeActions()</code>	Subscribe a list of actions that should be notified by a registered callback function
<code>OpenPushCallback()</code>	Register a callback function, that is called if a notification arrives or an event starts/stops or if one of the subscribed actions arrives

Table 4: Methods of a PLC object instance

To receive notifications and actions a callback function can be registered with the method `OpenPushCallback()`. After receiving an action, the action should be decoded and dispatched by an instance of the class `GngActionDispatcher`. The action dispatcher is a simple way to react on specific actions.

GEUTEBRÜCK

```
1    // initialization code:
2    ...
3
4    // connect to the server
5    MyServer->Connect();
6    ...
7
8    // define a handle to a PLC object
9    HGngPLC PLC;
10
11   // create a PLC object instance inside the DLL and
12   // get a handle to it
13   PLC = MyServer.CreatePLC();
14   ...
15
16   // link your callback function for a custom action
17   // to the action dispatcher, so that the callback function
18   // is called automatically if a custom action arrives
19   ActionDispatcher->OnCustomAction = this->MyCustomActionHandler;
20
21   // register a callback function for notifications,
22   // events and actions (this callback function dispatches
23   // all received actions with the help of the
24   // Gng Action Dispatcher)
25   PLC->OpenPushCallback(...);
26   ...
27
28   // destroy PLC object
29   PLC->Destroy();
30   ...
31
32   // destroy server object
33   MyServer->Destroy();
34
35   // callback function for all notifications, events and
36   // subscribed actions:
37   ...
38
39   // dispatch the received action to the linked
40   // callback functions
41   ActionDispatcher->Dispatch(Action Handle);
42   ...
```

Listing 5: Example for using the PLC

2.5.10 Media channel IDs

The existing media channels can be displayed by the viewer objects of the MediaPlayer interface. Normally this is done with the method ConnectDB(). This method needs the media channel ID to identify the media channel (camera) that should be displayed.

The media channel IDs are generated automatically by the G-Core Server. Every created media channel gets an ID which is always unique.



If a media channel is removed from the setup and added again, it will receive a new ID.

For that reason media channels should not be accessed by constant IDs. It is recommend using global numbers instead, because they can be changed in the setup. To find the fitting media channel ID for a given global number, the media channels should be enumerated from the server setup. Please refer to chapter 2.5.8 to see how this is done.

There is a similar difficulty with events, digital inputs and outputs. Events don't have global numbers. In this case the event name should be used instead.

2.5.11 Handling connection collapses

The callback OpenPushCallback() of the PLC object enables to listen to different kinds of notifications from the PLC object. One is the "plcnPushCallbackLost" notification. This notification is thrown if a connection is internally detected as collapsed. As a reaction on this event you should destroy or free all objects that were created inside the DLLs and start reconnecting. Reconnect attempts should begin every 30 seconds. Additionally your application can listen to UDP broadcasts that are sent by the G-Core Server. After your application received this broadcast it can directly try to reconnect to the server.



Please be aware of the fact, that broadcasts only work in LAN - routers normally block broadcasts.

2.6 Using the SDK with .NET

To make the usage of the native Win32/Win64 DLLs easier in .NET languages like C# or VB.NET, the SDK contains some wrapper assemblies around the plain SDK DLLs.

These wrapper assemblies are developed in C++/CLI and published with the SDK. The assemblies can be found in the G-Core SDK binary folder "Bin/x86" / "Bin/x64". Please refer to 2.1 to check which .Net Framework versions are currently supported.

The .Net wrappers must be used together with the unmanaged SDK dlls. For the dependancies please refer to 2.5.5.



Wrapper assemblies AND native SDK DLLs are needed in the same folder as the application!

2.6.1 Deploying a custom solution based on the .NET wrapper

To successfully deploy a custom application that uses the .NET wrapper contained in the SDK, the following prerequisites have to be fulfilled:

GEUTEBRÜCK

- Microsoft Visual C++ Redistributable Packages need to be installed.
The wrapper assemblies are developed in C++/CLI. For executing them on a none development machine, the Microsoft Visual C++ Redistributable Packages are needed. These packages exist in a debug or a release version. On productive machines the release version needs to be installed. For .Net applications developed with MS Visual Studio 2013 the C++ Redistributable Package 2013 is needed, for those developed with MS Visual Studio 2015 you need the Package 2015.
- .NET Framework Version 4.0 or newer has to be installed (depending on which framework was used for development).
- Wrapper assemblies, native SDK DLLs and the custom application need to be stored in the same folder.

3 Request of a G-Cert

3.1 G-Cert Introduction

When the development process of an SDK based application is finished a G-Cert is required which will qualify your application for connections to GEUTEBRÜCK systems. Otherwise it will work only with servers running in demo mode. To request a G-Cert for an application it needs to be signed with a certificate. For a signed application a matching G-Cert can be requested and downloaded from the GEUTEBRÜCK SDK portal at <http://sdk.geutebrueck.com>. This G-Cert needs to be placed in the directory where the signed application will be stored or in the folder where the GngDBI.dll is located.

The following chapters describe:

1. How to create a self-signed certificate, which can be used to sign an application if no registered certificate is available.
2. How to sign an application with tools included in the Microsoft Visual Studio installation.
3. Request the G-Cert for a signed application.

3.2 Creating a self-signed certificate

If there is no registered certificate available it's necessary to create a self-signed certificate to sign the application after the release. If there is already a certificate, this topic can be skipped.

To create a self-signed certificate, Visual Studio offers a function to be accessed via the project properties.

Open the properties page of the project. Go to "Signing" and put the hook on "Sign ClickOnce-Manifest". Now click the button "Create test certificate" and type a password twice. The certificate file starts with the project name and ends with .pfx. It is stored in the source path of the project.

3.3 Sign your application

There are two ways to sign an application with a certificate. Both need the Microsoft SignTool.exe which is part of the Microsoft Visual Studio installation. An application can be signed manually after each build using the SignTool in the developer command prompt. The other way is to add the sign tool command to the post-build events in the Visual Studio project. If this option is used, the application will be signed with each build automatically.

GEUTEBRÜCK

3.3.1 Using the developer command prompt

In the developer command prompt the signtool.exe has to be executed with some parameters.
signtool.exe sign /f "c:\Example\GLog.pfx" /p test1234 c:\Example\GCoreLogServer.exe

- /f <file> Specify the signing cert in a file.
- /p <pass.> Specify a password to use when opening the PFX file.

3.3.2 Sign automatically with each Build

To sign an application automatically with each build, the command from chapter 3.3.1 can be used more or less as a post build event to your project in Visual Studio. It's important to use absolute paths to all files. In this example the new command will be:

"C:\Program Files (x86)\Windows Kits\8.0\bin\x64\signtool.exe" sign /f "c:\Example\GLog.pfx" /p test1234 c:\Example\GCoreLogServer.exe

1. To add the post build event, open the project properties in Visual Studio by opening the context menu on the project in the SOLUTION EXPLORER and selecting PROPERTIES.
2. Now please select BUILD EVENTS and paste the signtool command to the field POST-BUILD EVENT COMMAND LINE

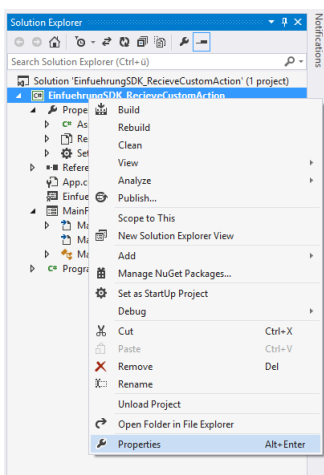


Figure 3: Open project properties

GEUTEBRÜCK

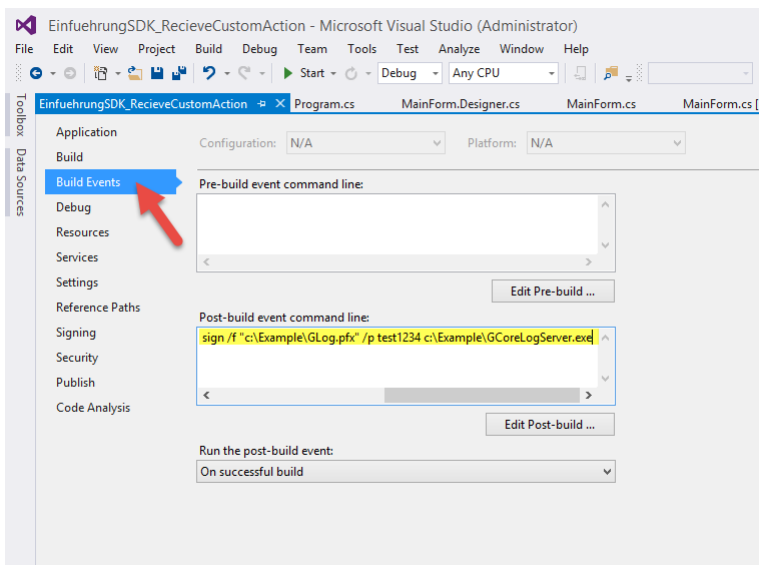


Figure 4: Access Post-build event command line

3. Please save the project and try to build it. If this was successfully each new build will automatically be signed.

GEUTEBRÜCK

3.3.3 Request your G-Cert

If there is a signed version of the application available, you can directly continue to request a G-Cert for this application.

Open the Geutebrück SDK portal and login with your credentials.

The screenshot displays the Geutebrück SDK web interface. At the top, the 'GEUTEBRÜCK' logo is visible. Below it, a code editor shows C# code for initializing a PLC and setting up event handlers. To the right of the code editor, a sidebar contains the text 'GEUTEBRÜCK SDK' and 'INTERFACE DEVELOPMENT - MADE SIMPLE!'. Below the code editor, a login form is presented with the heading 'Are you already registered as developer?' and the instruction 'Then log on with your credentials to access the desired information.' The login form includes fields for 'Username' and 'Password', a link for 'Forgot your password?', and a 'Login' button.

```
/// <summary>
/// Initializes the PLC, event handler and subscribes actions, events and blocking filters
/// </summary>
private void CreatePLC()
{
    DestroyPLC();
    m_GngPLC = m_GngServer.CreatePLC();
    // using PLC push callback with event notification
    m_GngPLC.PLCCallback += new PLCCallbackEventHandler(GngPLC_PLCCallback);
    m_GngPLC.OpenPushCallback();

    // Initialize ActionDispatcher
    m_GngActionDispatcher = new Dispatcher();

    // Register action event handler
    m_GngActionDispatcher.CustomAction += m_GngActionDispatcher_CustomAction;
    m_GngActionDispatcher.DigitalInput += m_GngActionDispatcher_DigitalInput;
    // further action event handler ...
}
```

Are you already registered as developer?

Then log on with your credentials to access the desired information.

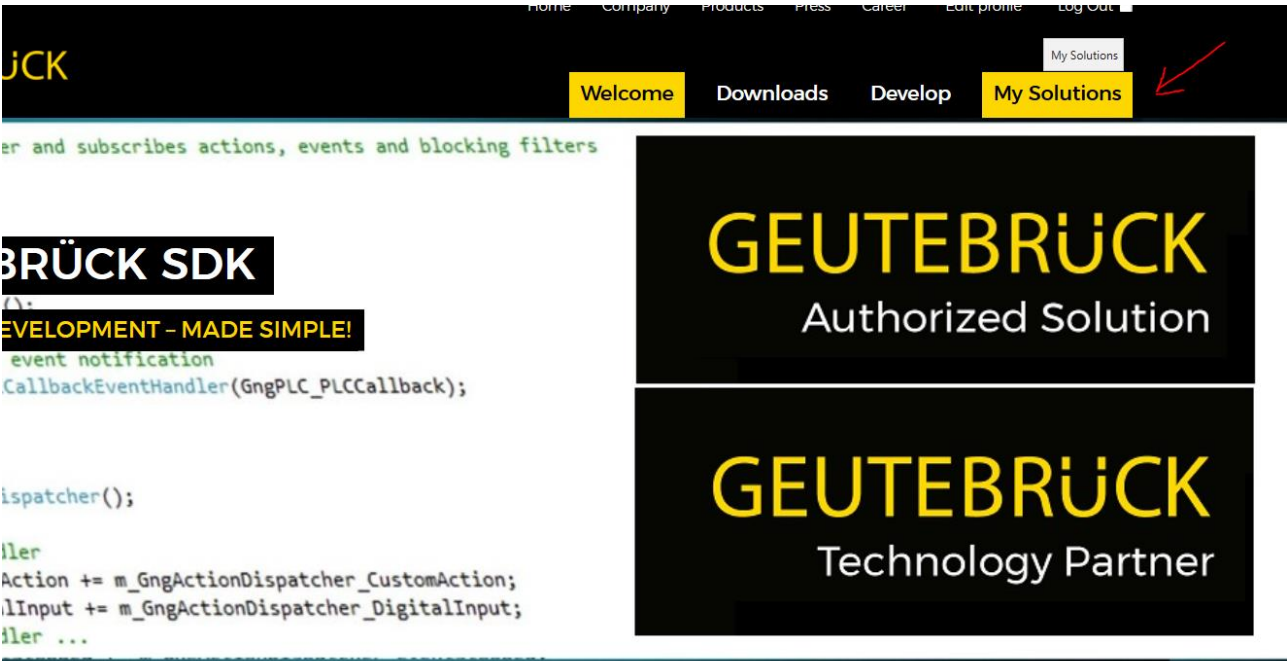
Username

Password

[Forgot your password?](#) [Login](#)

Figure 5: Login to SDK website

Navigate to the My Solutions area.




The GEUTEBRÜCK SDK site

Present in our SDK and products. With our brand new G-Core software the G-World is completed with our G-Core SDK. The G-Core SDK can be used in third-party software, to send or receive actions or to write plugins for our G-Core server.

Figure 6: My Solutions on the SDK website

Now there is an overview of all already registered solutions. At the bottom of this list you can add a new solution to the account.

EDIT	AUTHORIZED	VIEW EDIT
Edit	Authorized	View Edit
Edit	Authorized	View Edit



Add new Solution

Figure 7: Add a new solution

Please fill out the fields in the General Info and the Solution info tab and click save and request cert to go to the next step.

Register a new solution

Please fill out the form as detailed as possible. We need this information to provide you with the best possible support.

General info	Solution info
Solution Software which will be developed.	
Name *	
Description *	
Geutebrück version *	
Programming Language *	
.NET Framework *	
Interface	
Type of solution	
Planned Functions*	
<input type="checkbox"/> Seamless Integration of live and recorded video in the graphical user interface of the third party system	
<input type="checkbox"/> PTZ control out of the graphical user interface of the third party system	
<input checked="" type="checkbox"/> Manual Recording	
<input type="checkbox"/> Exchanging alarm information (sync signal lost, motion detection, start/stop of event recordings, ...)	
<input type="checkbox"/> Export Snapshots	
<input type="checkbox"/> Export Video Sequences	
<input type="checkbox"/> Displaying belonging video of an alarm handled by the third party system	
Other:	
Back	Save Save and Request Cert

Figure 8: Register a new solution

In the next step fill out the web form in the CERTIFICATION tab and go to the G-Cert tab. Here you have two possibilities to add the required data for the G-Cert. The easiest way is to upload the signed application. The other way is to select MANUAL INPUT and enter the filename of the application and the subject which was set in the certificate. The tool "G-Core GCertRequest.exe" can help you with that. It can be downloaded from the SDK website.

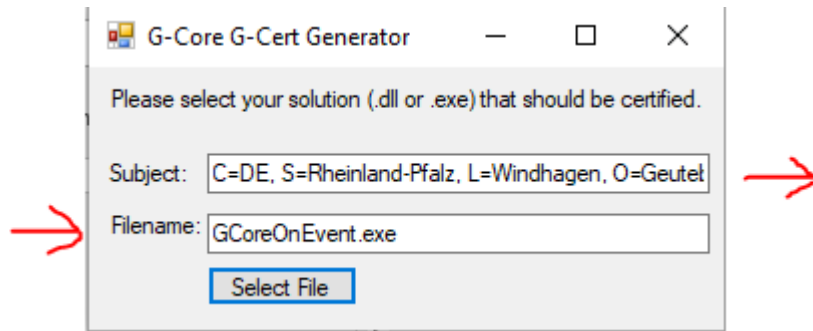


Figure 9: G-Cert Generator tool

The resulting subject must be copied to the subject field in the web form.



To get a working G-Cert, it's important that filename and subject are matching exactly.

As this is in most cases the problem, that a G-Cert is not working, we suggest to upload the application. In this case the filename and the subject of the certificate will be readout during the generation of the G-Cert. If everything is filled out, the request can be submitted by using the request button. An email confirmation will be send if the G-Cert was approved and published.

Geutebrück | My Solutions | SimpleClient

Request GCERT and "GEUTEBRÜCK Authorized"-Logo

This is the last step to use your solution in production mode. Please fill out the form as detailed as possible. If important information is missing we will refuse your request.

Please be aware if you tick "Show solution in interface list" we will add your solution to our interface list on our GEUTEBRÜCK website.

Certificate for solution: **SimpleClient**

Certification

G-Cert

API key

To verify that you have a valid GCERT we need the filename and the subject of your signed solution.

If you use TACI to integrate your third party application you can request a GCERT for TACI.

If you use our libraries to integrate you will need to upload your solution or manually input filename and subject of your Digital Signature. To do that you can use our G-Cert Request tool to get this information.

☐ TACI

☒ Upload

Durchsuchen...

Current file: **SimpleClient.exe**

☐ Manual input

File name

SimpleClient.exe

Subject (set in Digital Signature of .exe/dll)

CN=GEUTEBRUECK\Ewenz1311

Back

Save

Figure 10: Upload the signed executable or dll

In the MY SOLUTIONS area of the GEUTEBRÜCK SDK Website you can see Status of the submitted requests. If a G-Cert was approved and created it's indicated with the status **AUTHORIZED**. To download the G-Cert please click **VIEW** in the **CERTIFICATE** area (see Figure 7). Under the G-Cert tab just click on „certificate here“.

Home
Company
Products
Press
Career
Edit profile
Log Out

GEUTEBRÜCK

Welcome
Downloads
Develop
My Solutions

Geutebrück | My Solutions | SimpleClient

Certificate for solution: SimpleClient

Certification

G-Cert

API key
gWrHDHUhVz7gep3AFztxFnzfQo/P7aOn06xmoc
CncporJVUIRS6YoXR2tWqzRnE0Elvj/qI7ZKWJ2
VloiqGo/RvmFypJA7jZlBvs/xmdBsENfDq51oo+M
yBsjTciQz1Uxku6hvxWEkdjyNHylZ074nehbecIJx
f6eqIFWctc=

Download your
certificate here.
Download authorized partner logo.

G-Cert version
4.1.1.625

Solution subject
CN=GEUTEBRUECK\Ewenz1311

Solution filename
SimpleClient.exe

Issuer name
SimpleClient

Issuer ID
686

Validity
Not-Before
28.03.2019 - 06:54
Not-After
28.03.2020 - 06:54

Back

Figure 11: Download the certificate



The certificate must be stored in the same folder as the application.



Currently it is not possible to certify a .Net dll because the testing mechanism of the SDK is not able to recognise .Net dlls. In case you don't want to certify the executable for whatever reason and the library file which handles the G-Core connection is a managed .Net dll, you have to request a GngDBI.dll G-Cert. In such a case please contact the SDK team. We will ask you to sign an NDA.