GeViScope SDK

Dokumentation | Documentation | Documentation | Documentatión

GEUTEBRÜCK knowledge base

Version 04.2013

GeViScope Software Development Kit (SDK)

Introduction

The GeViScope SDK consists of a collection of free software interfaces for the GEUTEBRÜCK DVRs GeViScope and RePorter. It can be used to integrate these devices in custom applications and although for linking not yet supported peripherals.

The interfaces are based on native Win32 DLLs. So they can be used with various development platforms of the Windows OS.

To support the .NET technology the SDK examples contain wrapper classes based on C++/CLI. These wrapper examples can be freely used, modified and extended by the SDK users. The C# examples included in the SDK demonstrate, how the wrappers can be used by custom applications.

Contents

Documentation-History Version 3.9 / PME

Files and directory structure of the SDK

During the installation of the SDK the environment variable %GSCSDKPATH% which points to the root directory of the SDK is set. This reference path is used in all examples.

%GSCSDKPATH%\Bin	Contains all dynamic link libraries and is the target directory for the compiled examples
%GSCSDKPATH%\include	Contains all Delphi import units, C++ header and cppfiles
%GSCSDKPATH%\lib	Contains all lib files for Borland C++ Builder and Microsoft Visual C++

The matching interface units between C++ and Delphi have the same name but compiler specific file extensions.

Setting up a virtual test environment

Introduction

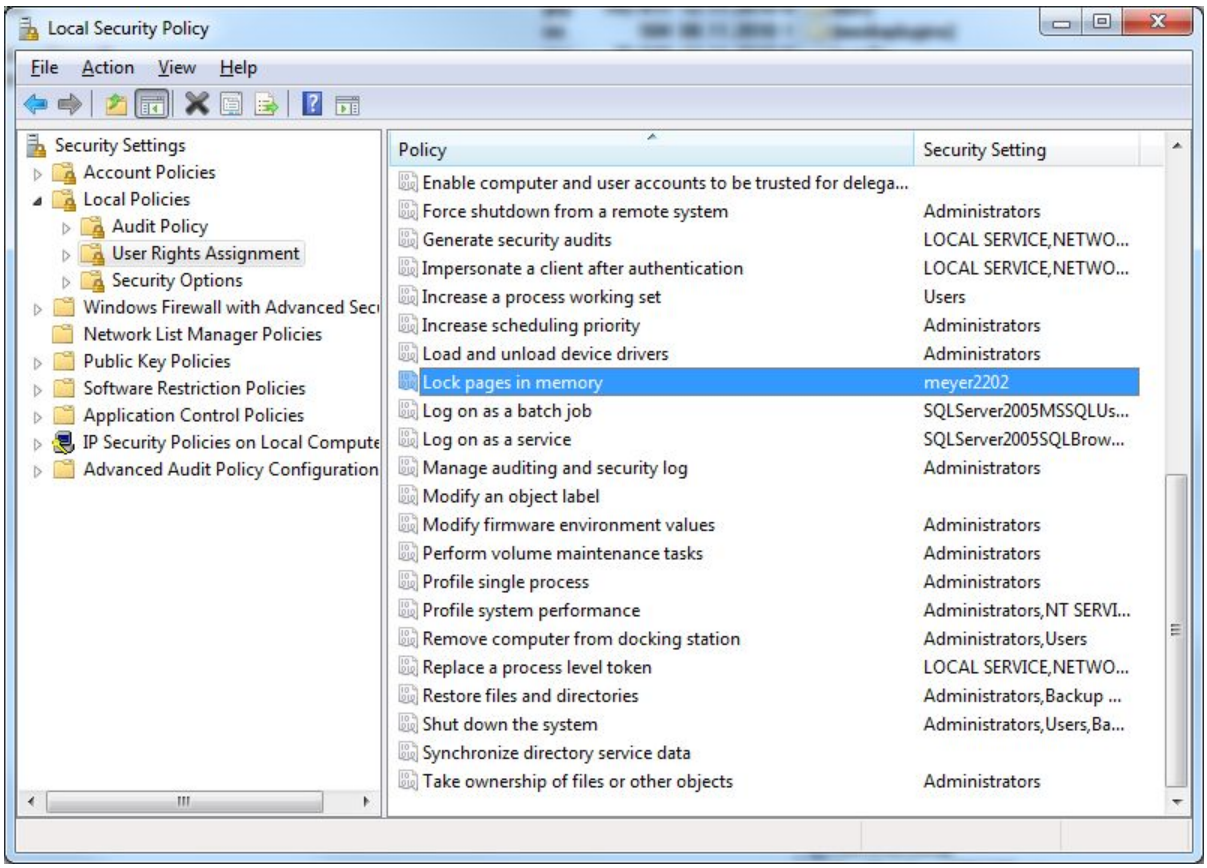
All required components for setting up a virtual GeViScope device are included in the SDK. So an independent development of custom solutions can be achieved without any special hardware required.
After starting up the GeViScopeserver (part of the virtual GeViScope device) GeViScope software can be used with full function for two hours. After that time the functionality is limited. After stop and restart of the server full functionality is offered for two hours again.

Step by step

After the successful installation of the SDK all necessary files exist in the installation folder (normally "%HOMEPATH%\My Documents\GeViScopeSDK").

Step 1: Assign local policy "Lock pages in memory"

To run GeViScopeserver on your local machine, a local policy needs to be assigned to the user account under which GeViScope server should work.
Please open the "Local Security Policy" dialog in the control panel – Administrative Tools.



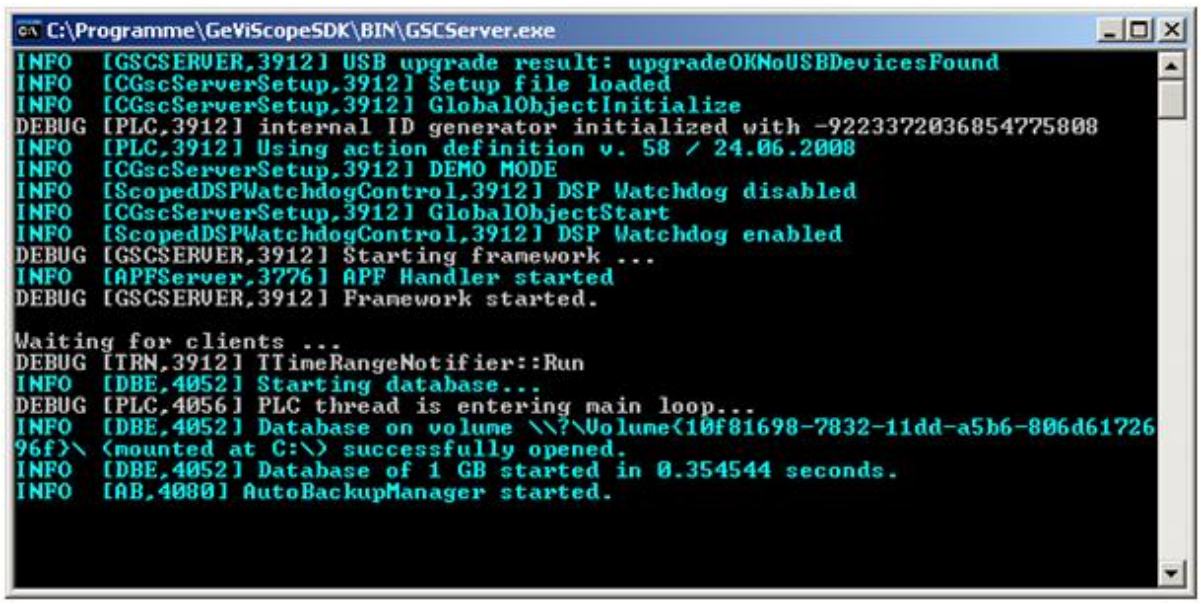
With "Security Settings / Local Policies / User Rights Assignment" the privilege "Lock pages in memory" has to be assigned to the user account under which GeViScope server should run.
The user has to be a member of the local Administrators group.
The user has to logout and login again to let the setting take effect.

Step 2: unpack the test files

Unpack the file "\BIN\GeViScope.Database.zip" to the root directory of your system drive (normally "C:"). Afterwards the file "C:\GeViScope.Database" should exist. Please note that the file is not seen in the windows explorer if hidden files and folders are masked out.
Unpack the file "\BIN\DatabaseBackup.zip" to the sub folder "\BIN" of the GeViScope SDK base directory (normally "%HOMEPATH%\My Documents\GeViScopeSDK"). After that the file "\BIN\DatabaseBackup.gpf", which contains a test backup file in GBF format ("GEUTEBRÜCK Backup File") should exist.

Step 3: start the GeViScopeserver

Start the server by double clicking on file "\BIN\GSCServer.exe". Now a console application should start.



Step 4: import the test setup

Start the GSCSetupsoftware (file "\BIN\GSCSetup.exe") and establish a connection to the local server. Use the following login information:
Username = sysadmin
Password = masterkey
Send the setup once to the server by using the menu entry "Send setup to server".
The test setup "*\BIN\GeViScopeSDKSetup.set*" can be imported into the server with the help of the menu entry "Import setup from file". Afterwards it should be send to the server once again.

Step 5: view live video and backup video in GSCView

Now the correct setup of the test environment should be tested. For that purpose the GSCViewsoftware (file "\BIN\GSCView.exe") can be started and again a connection to the local server should be established. After a successful connection media channels are available and can be viewed. Simply drag the media channels on the viewers of GSCView.

The menu entry “Open backup file…” allows opening the test backup file “\BIN\DatabaseBackup.gpf“, which also contains media channels that can be displayed. Please check the correct function of the backup by play back the video material.

Step 6: Use of tool “\BIN\ GSCPLCSimulator.exe”

The software “\BIN\ GSCPLCSimulator.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 message flow in the complete system.

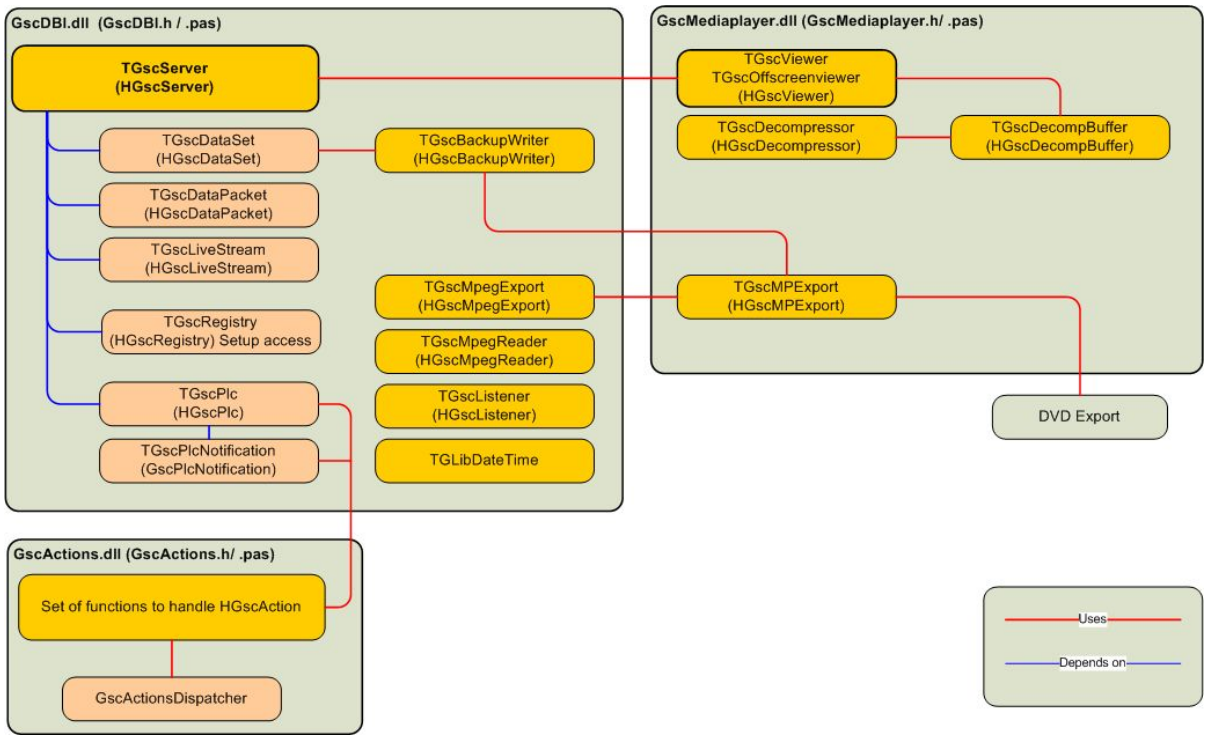
Background information

To provide a test environment with full functionality the GeViScope media plugin“MCS” (Media Channel Simulator) is used. It simulates real video media channels by channeling test pictures into the GeViScopeserver. 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.
The media plugin“MCS” is part of the SDK including source code (development platform Borland C++ Builder 6) and documentation (please see topic “Examples overview” for more information).

Overview of the interfaces in the SDK

Introduction

This document gives a short overview of the different interfaces that belong to the SDK. Please note, that 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 DLLs have to be added to the custom project.



Building blocks of functionality

DBI

- 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: GSCDBI.DLL
- Main include files (C++): GSCDBI.h, GSCDBI.cpp
- Main include files (Pascal): GSCDBI.pas

PLC

- Complex notification, action and event processing
- Listen to, dispatch, create and send actions
- Listen to events and system notifications
- Allows controlling and monitoring the system
- Main binary file: GSCActions.DLL
- Main include files (C++): GSCActions.h
- Main include files (Pascal): GSCActions.pas

TACI

- Telnet Action Command Interface
- Simple ASCII-Format communication based on Telnet
- Allows controlling and monitoring the system
- Received actions need to be parsed

- To use that interface, the media plugin “GSCTelnetActionCommand” needs to be installed

MediaPlayer

- High level server and database interface including media presentation
- Display video, play audio (live and backup)
- Integrated export functionality (GBF, MPEG, Video-DVD, Single picture)
- Search media data by time or corresponding to event data
- Main binary file: GSCMediaPlayer.DLL
- Main include files (C++): GSCMediaPlayer.h, GSCMediaPlayer.cpp
- Main include files (Pascal): GSCMediaPlayer.pas

OffscreenViewer

- Part of building block “MediaPlayer”
- Same functionality as MediaPlayer, but: no rendering, only decompressing
- Class TGSCOffscreenViewer can be used analogous to TGSCViewer

Media plugin (GeViScope server plugins)

- GeViScope server plugins allow integrating custom peripherals in GeViScope systems
- Channeling of video and/or audio media into the server
- Including full access to PLC
- Plugins run as In-Process-DLLs in GeViScope server software

GSCView data filter plugin

- GSCView plugins allow integrating custom data filter frontends in GSCView software
- Plugins run as In-Process-DLLs in GSCView software

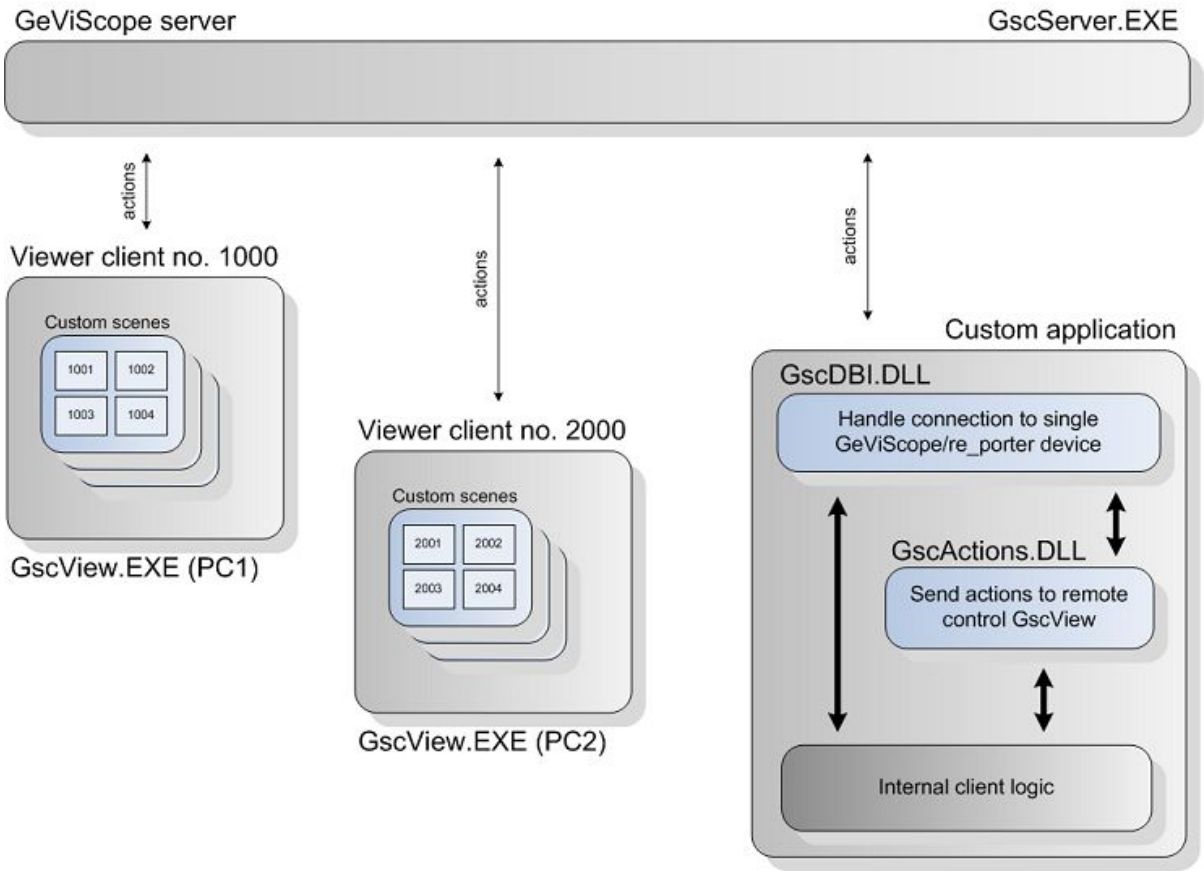
GSCView data presentation plugin

- GSCView plugins allow customized presentation of event data in GSCView software, especially of event data presented in viewed pictures
- Plugins run as In-Process-DLLs in GSCView software

Remote control GSCView by actions

Introduction

The simplest approach to view and browse live and recorded video of one or more GeViScopes is to remote control GSCView out of custom solutions.
GSCView can be used in a special mode so that it can be controlled by actions that are sent from a GeViScope server. The actions can be channeled into the system using the SDK (GSCDBI.DLL and GSCActions.DLL) in custom applications. As an alternative the actions can be sent to the TACI interface of the GeViScope server. The TACI interface is a media plugin of the GeViScope server, which can receive actions as ASCII text commands similar to a TELNET communication. The TACI plugin has to be licensed.

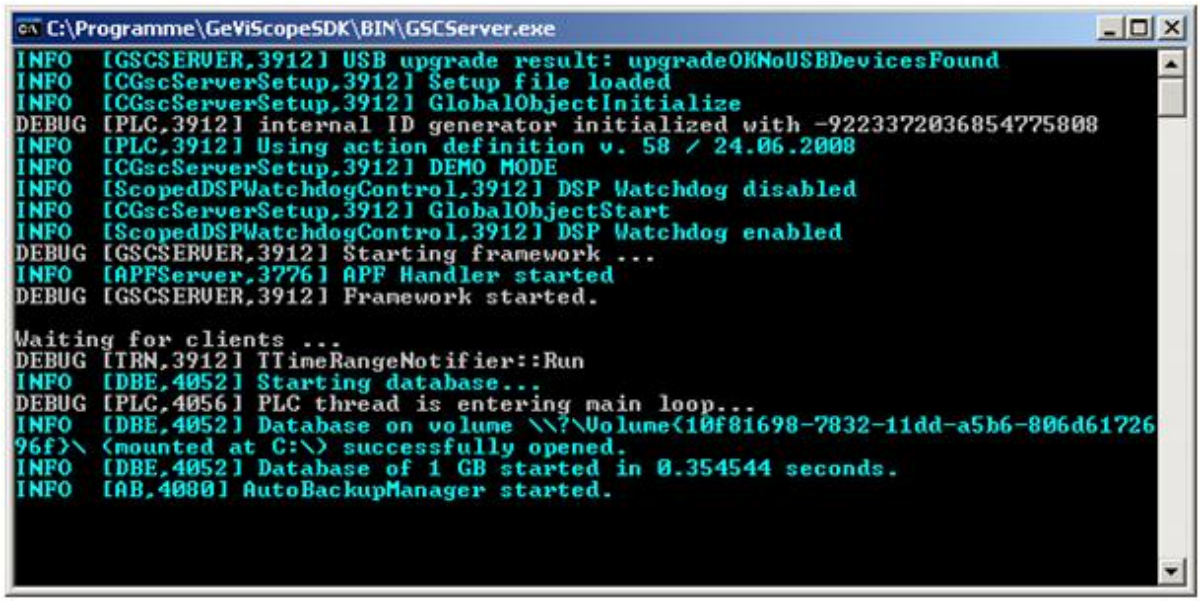


Step by step

The following step by step instructions show how to configure a simple system to demonstrate remote controlling GSCView. The virtual test environment included in the SDK should be successfully installed and set up before following these instructions (see topic *Setting up a virtual test environment*).

Step 1: start the GeViScope server

Start the server by double clicking on file “\BIN\GSCServer.exe“. Now a console application should start.

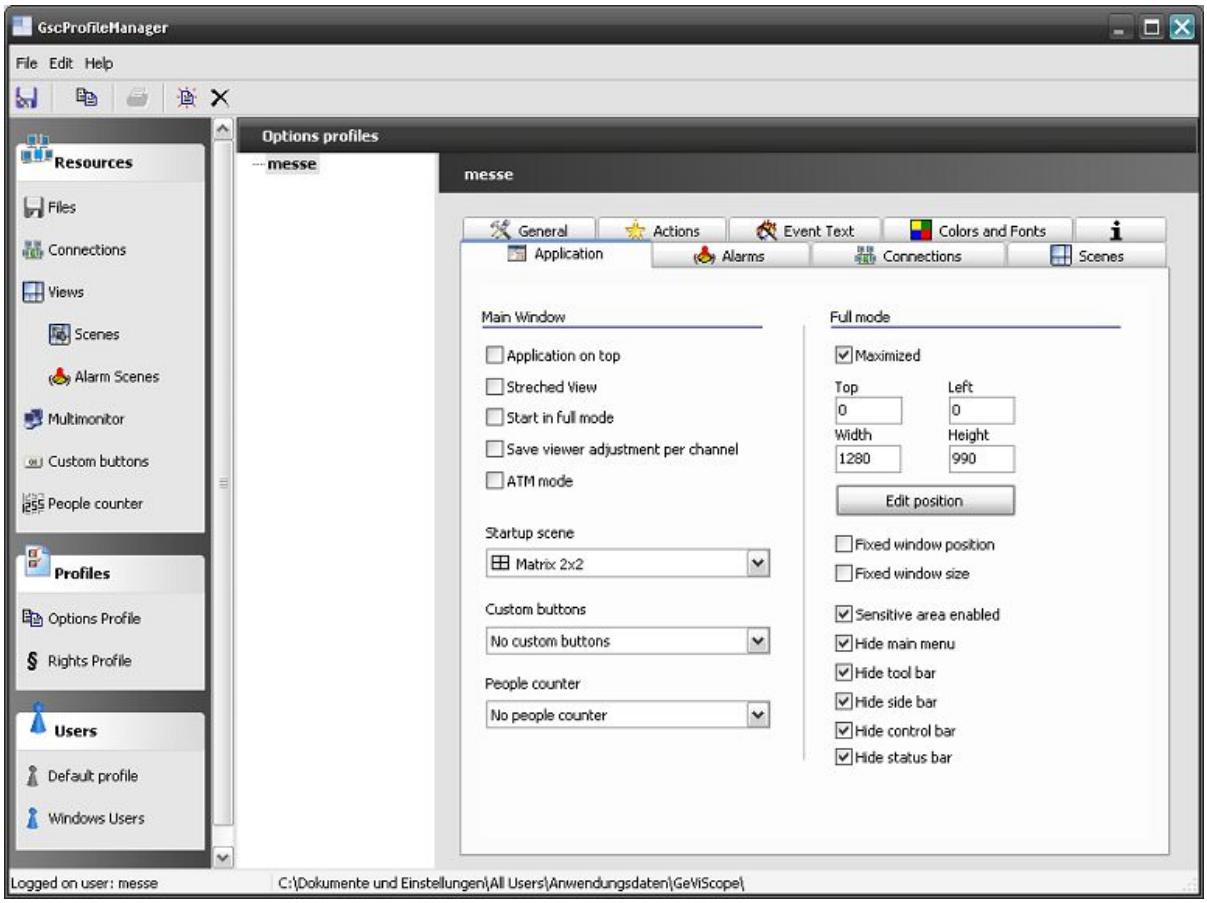


Step 2: start GSCView

Start the GSCView software (file “\BIN\GSCView.exe”).

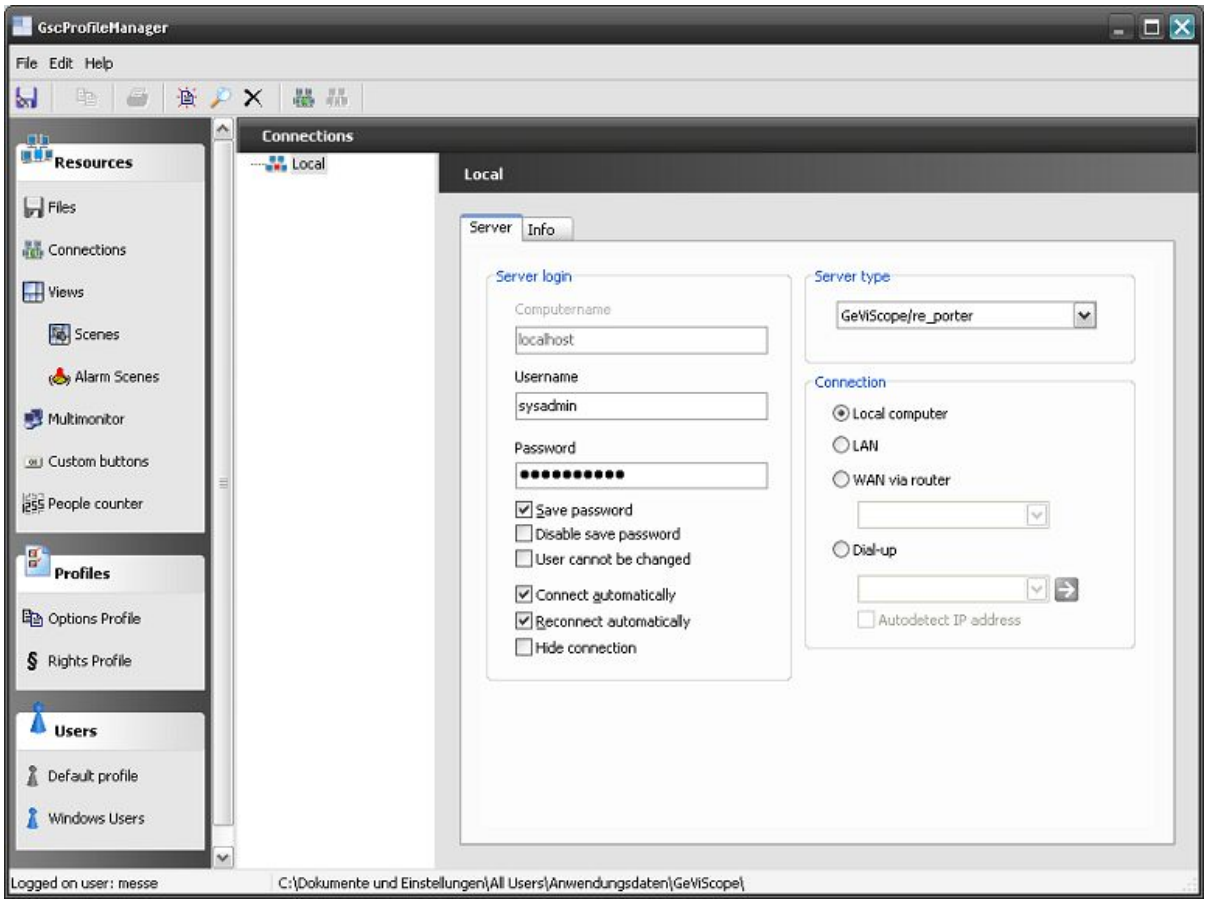
Step 3: start the profile manager

The menu entry “Options – Profile manager…” starts the internal profile manager of GSCView. The profil manager allows configuring all GSCView settings.



Step 4: declare local connection as "connect automatically"

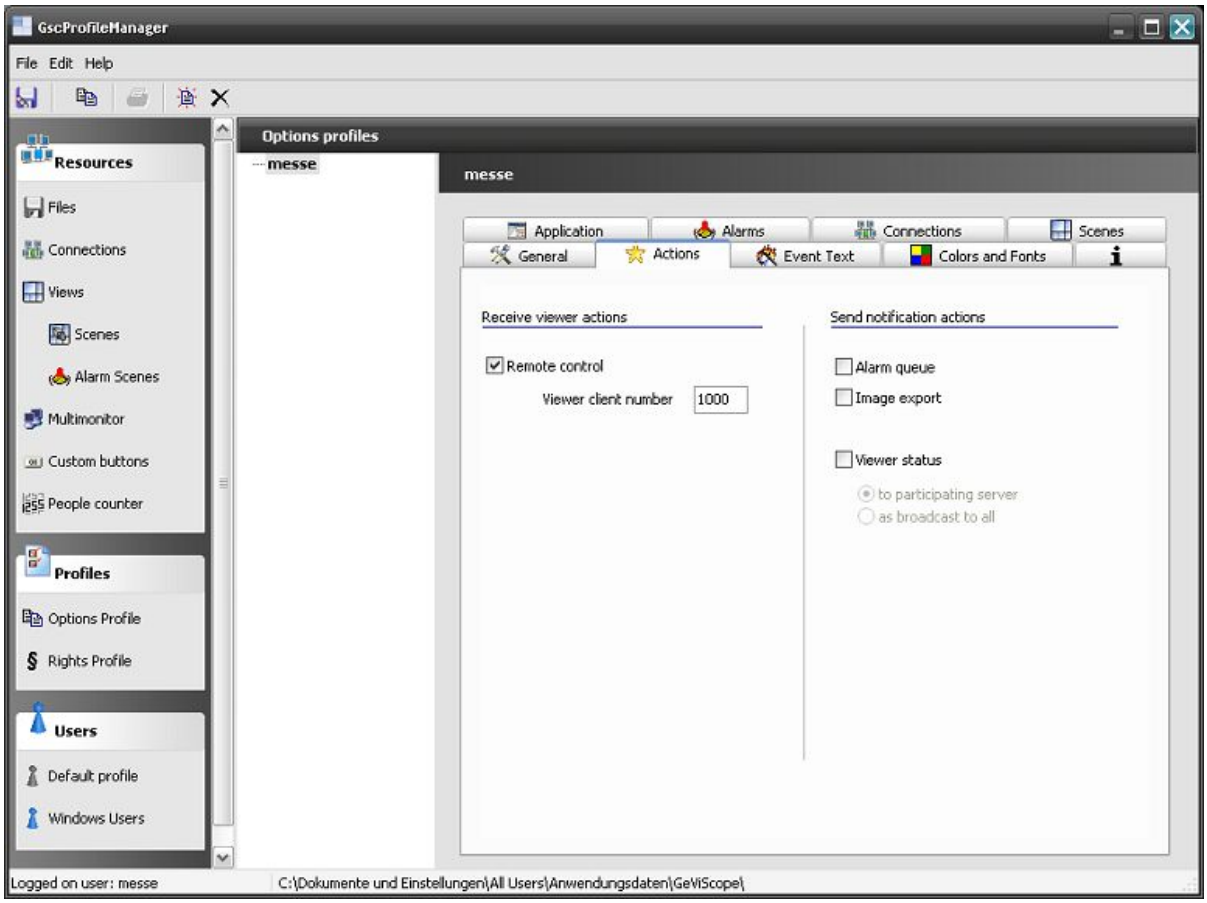
By selecting "Connections" in the section "Resources" the local connection can be declared as a connection that is automatically built up after starting GSCView. Additional the option "Reconnect automatically" should be activated.



If the connection is open in GSCView or GSCSetup, the settings of the connection cannot be changed. Close all local connections at first to be able to change the connection settings.

Step 5: configure GSCView to be able to remote control it by actions

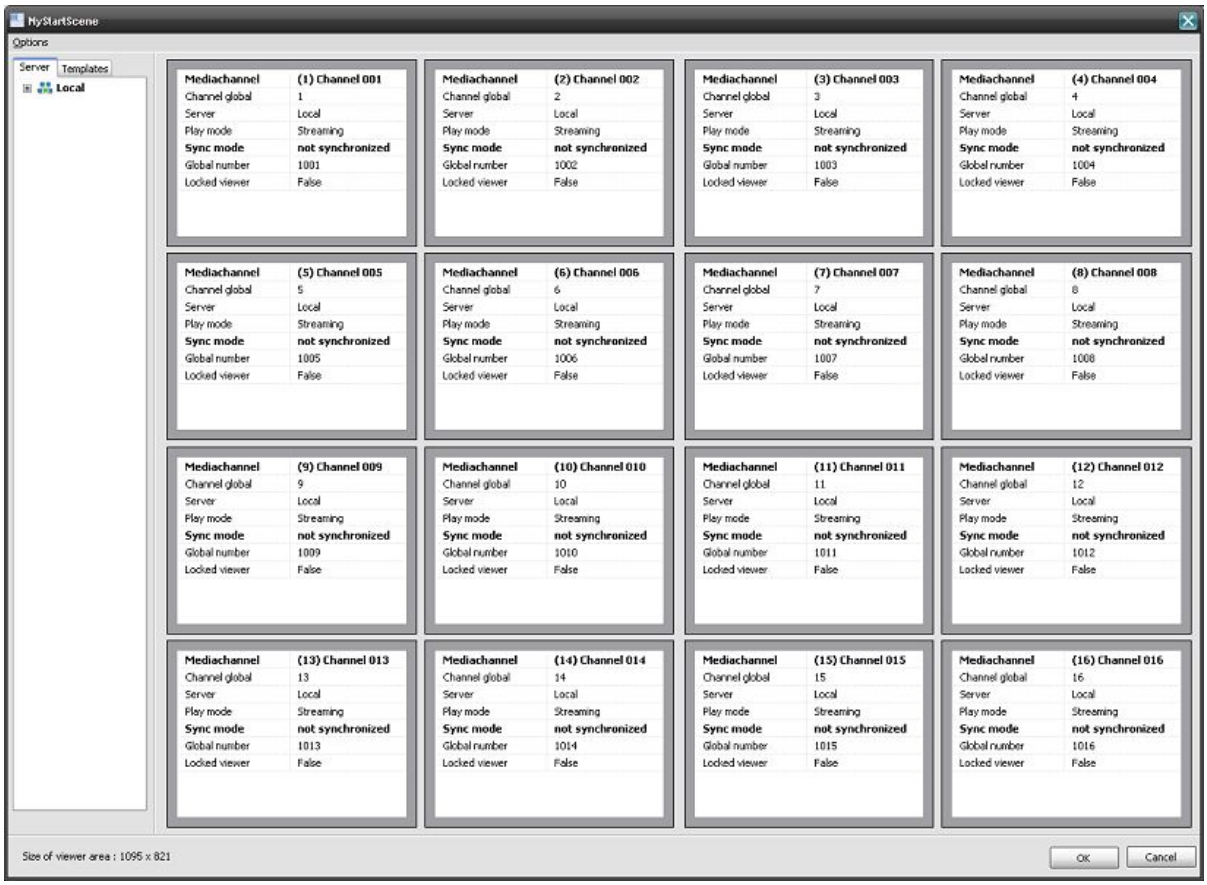
The entry “Options profile” in the section “Profiles” shows a tab control with a lot of different GSCView settings. To be able to remote control GSCView the option “Remote control” on the “Actions” tab has to be set.



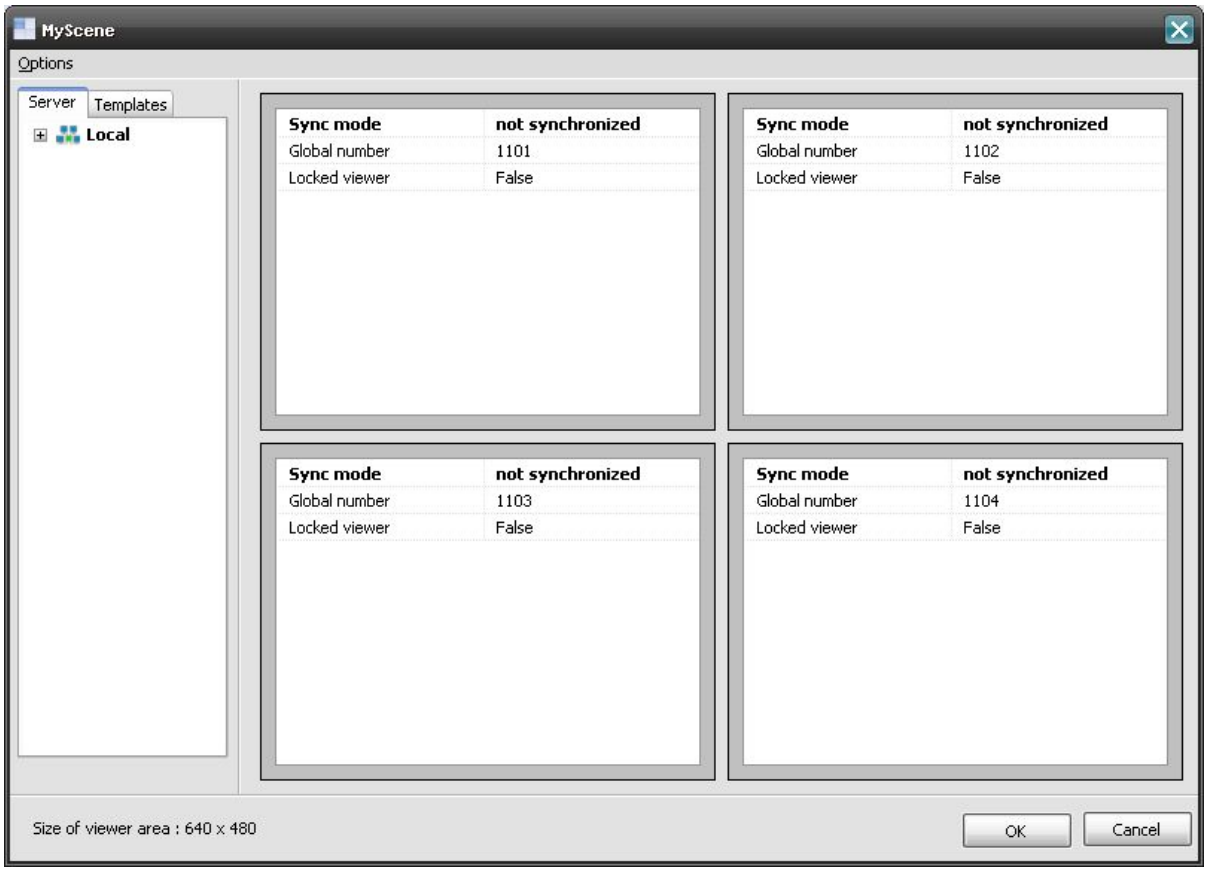
The “Viewer client number” should be set to a arbitrary global number that is unique in the whole system. This global “Viewer client number” identifies this special instance of GSCView in the whole network. The number is used in different actions to remote control GSCView.
By contrast the “global number” of a viewer in a custom scene identifies a special viewer in a user defined scene. Details about user defined scenes will be topic of the next step.

Step 6: user defined scenes

To define user defined scenes in GSCView the entry “Scenes” in section “Resources” should be selected. By right clicking on one of the predefined scenes new user defined scenes can be created. For this step by step example two new scenes with the names “MyStartScene” and “MyScene” have to be added. With the button “Edit scene” the global numbers of the viewers of the scene and the video channels that should be displayed can be set.
The “MyStartScene” should be based on the “Matrix 4x4”. The viewers should have the global numbers 1001 to 1016. Each viewer should display live pictures of a video channel of the local connection. The video channels can be set via drag & drop while editing the scene.

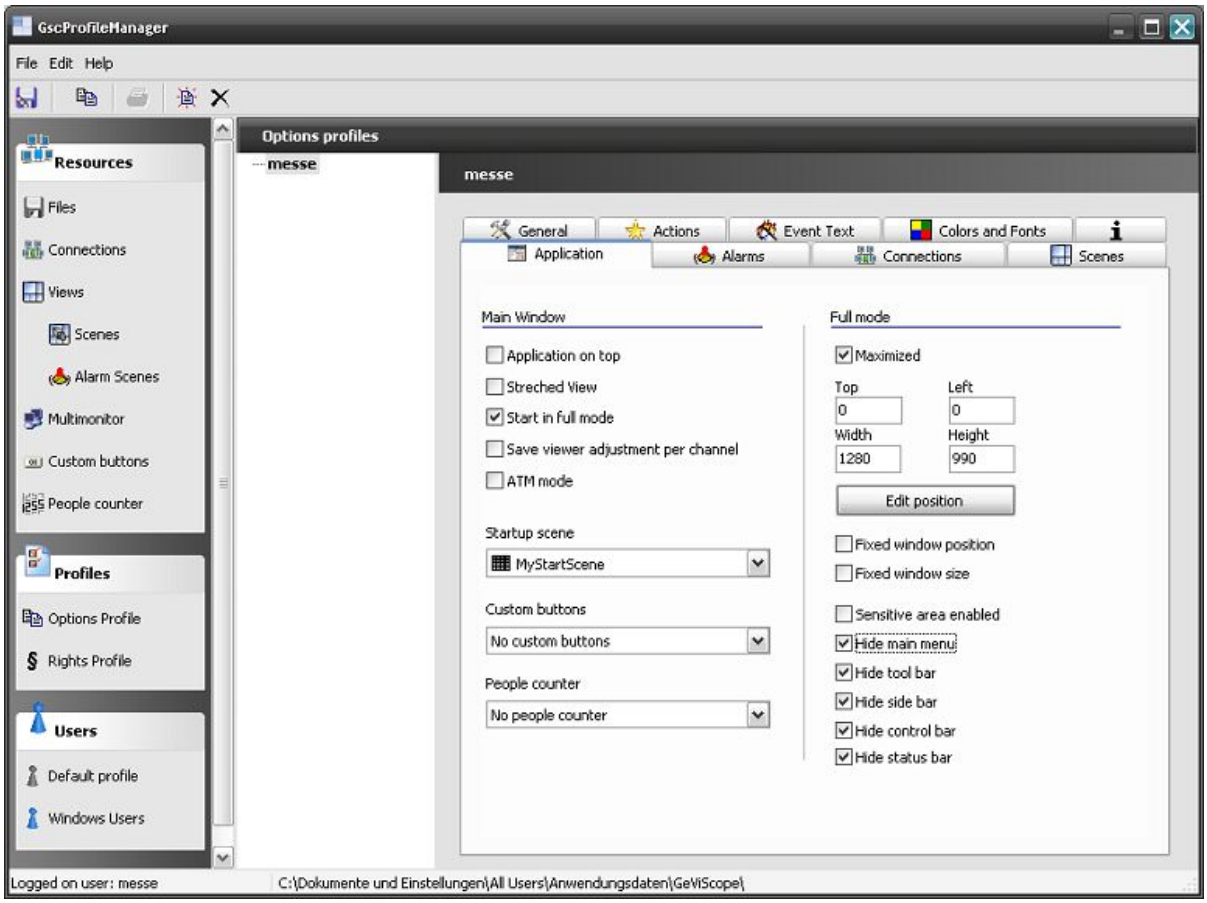


The “MyScene” should be based on the “Matrix 2x2” and the viewers should have the global numbers 1101 to 1104. The viewers should not automatically display any video channel. They will be used by special actions to display video channels.



Step 7: modify the appearance of GSCView

The appearance of GSCView can be controlled by different settings in the entry “Options profile” of the section “Profiles”. For this test scenario, GSCView should appear as a stupid video wall without any user controls directly visible in the GSCView application window. To achieve this, the following options on the “Application” tab have to be set:



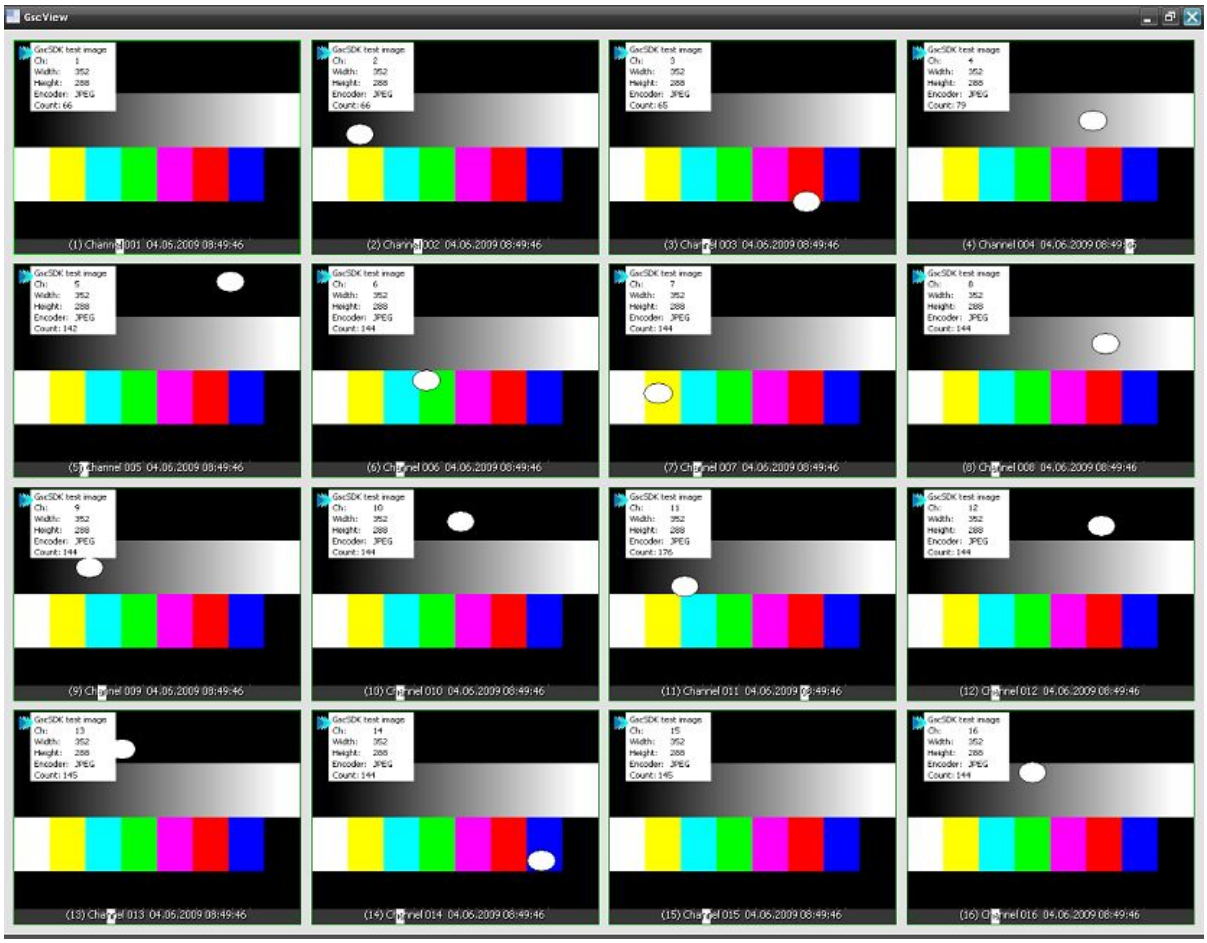
Please keep in mind, that if the option “Sensitive area enabled” is not set and if all “Hide…” options are set, the main menu of GSCView only can be accessed by pressing F10!

Step 8: save all settings

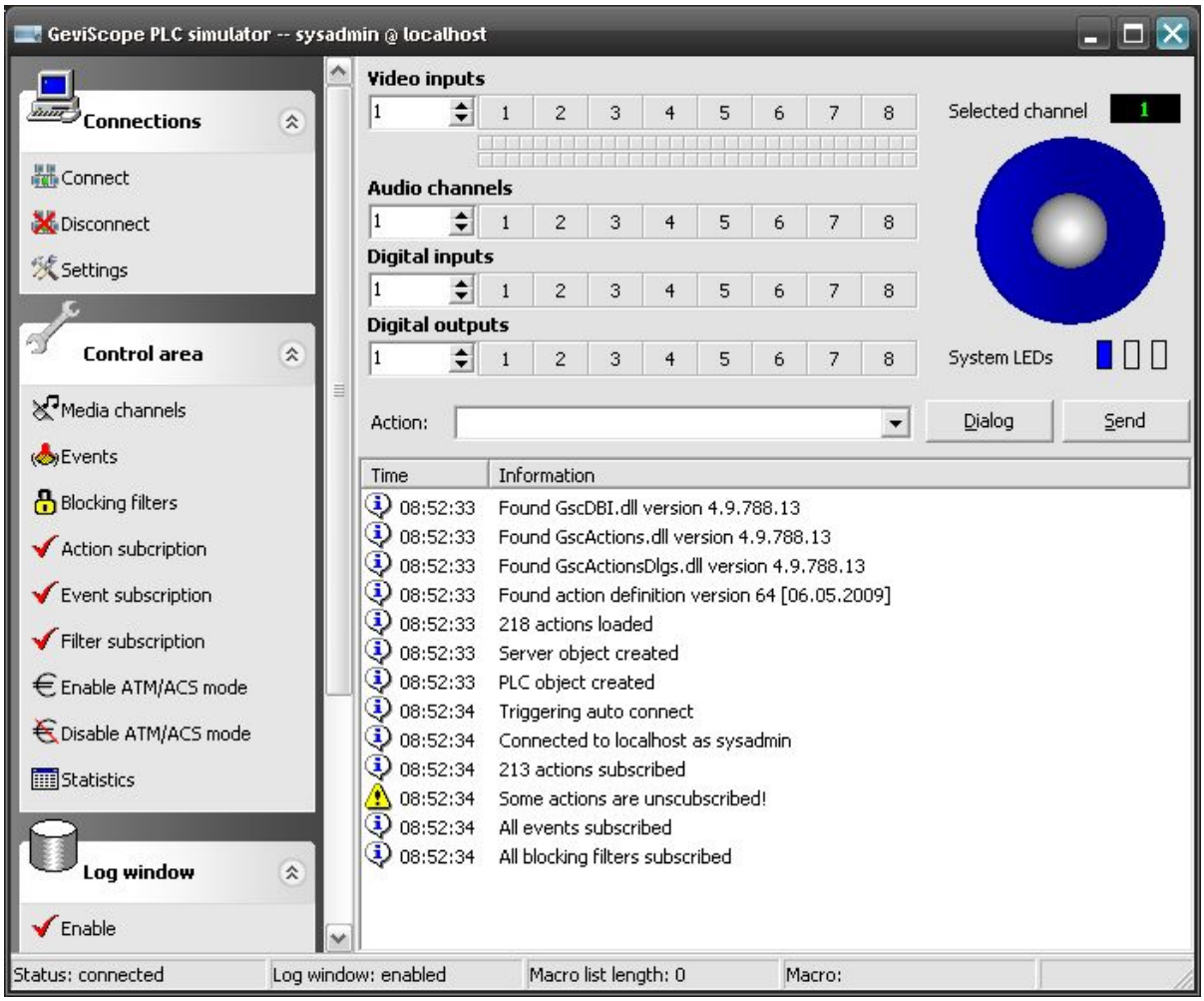
All settings should be saved by selecting the menu entry “File – Save”.

Step 9: test the system with GSCPLCSimulator

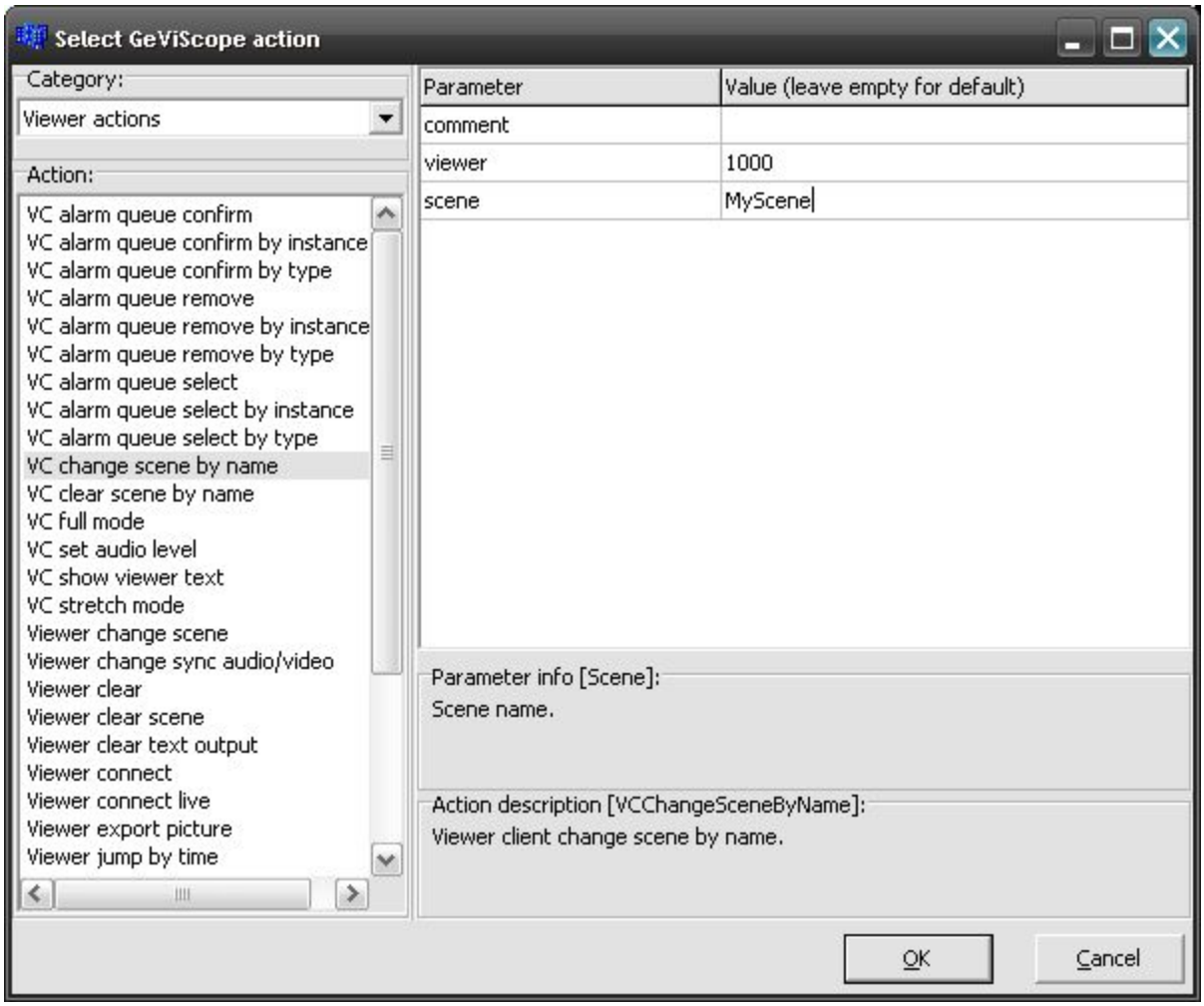
After restarting GSCView it should appear in full mode with 16 viewers displaying live pictures of the video channels of the local connection.



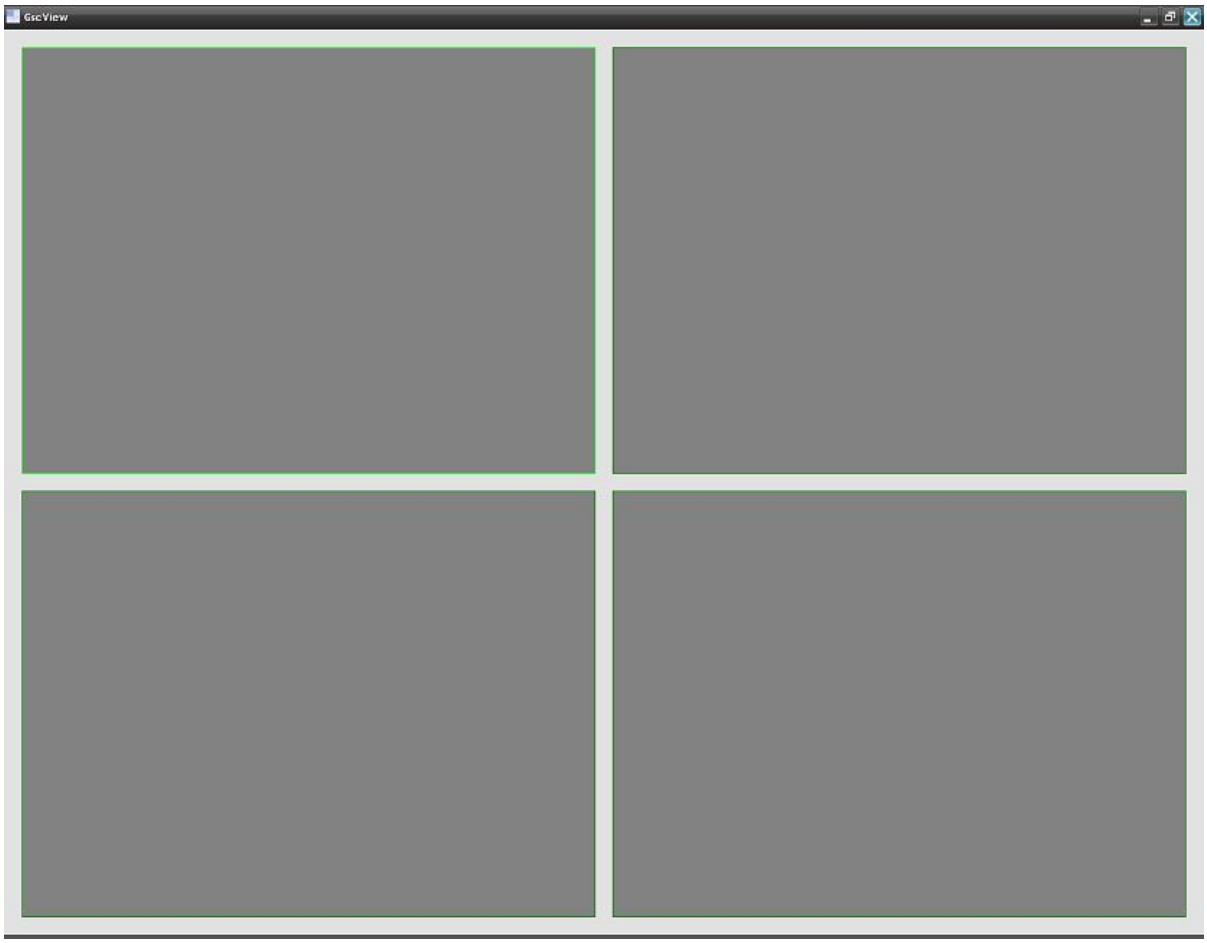
Now start the software “\BIN\ GSCPLCSimulator.exe” to test the system. The GSCPLCSimulator serves as a monitoring tool for all messages (actions) and events that are transported inside the complete system. Furthermore actions can be triggered and events can be started and stopped.
After its start the connection to the local server should be build up automatically and all action traffic is displayed in a list.



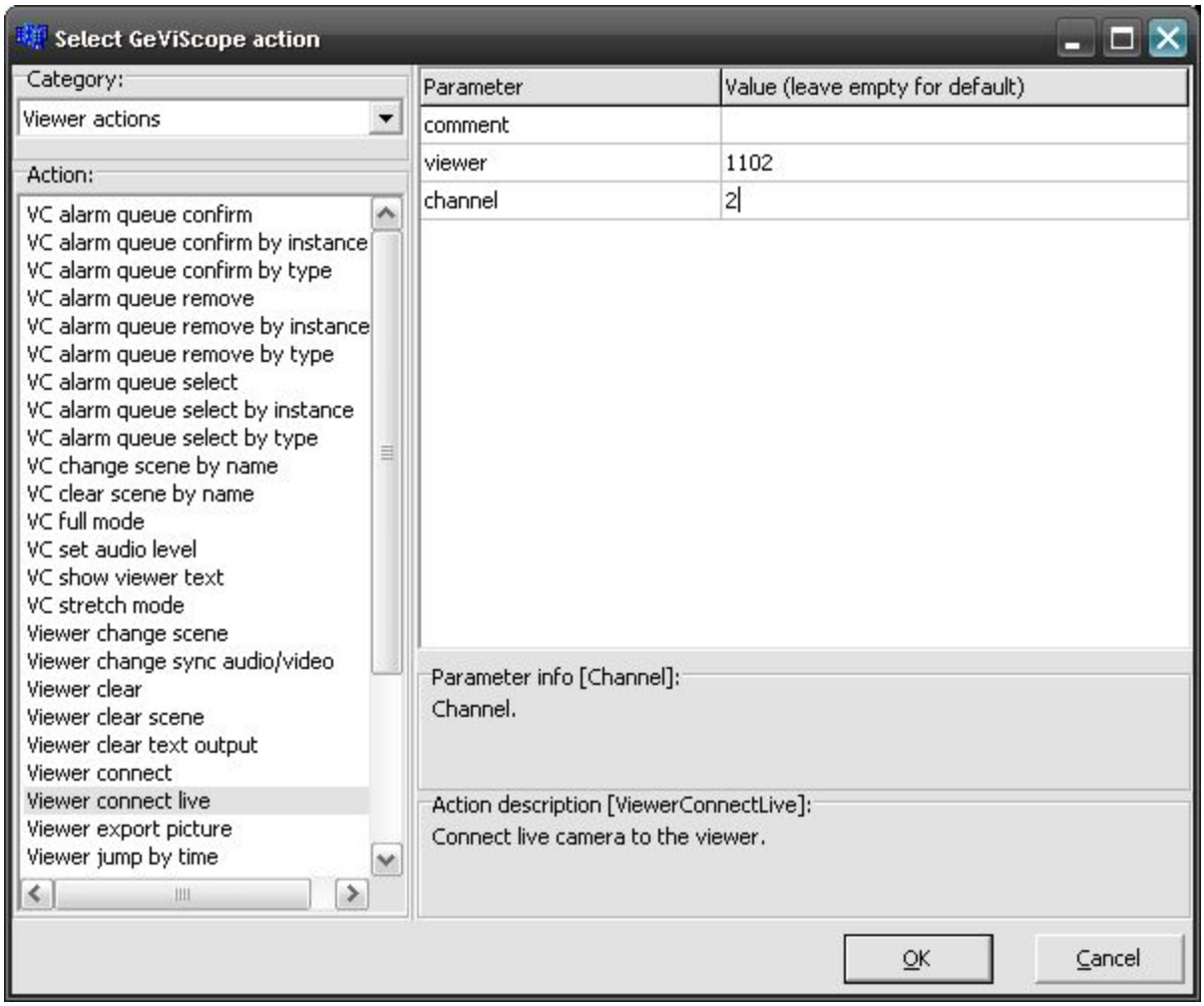
With the button “Dialog” an action can be selected and with the button “Send” this action can be send to the GeViScope server. For testing the system first select the action “VC change scene by name” in the category “Viewer actions” to display “MyScene” on the GSCView with the global “Viewer client number” 1000.



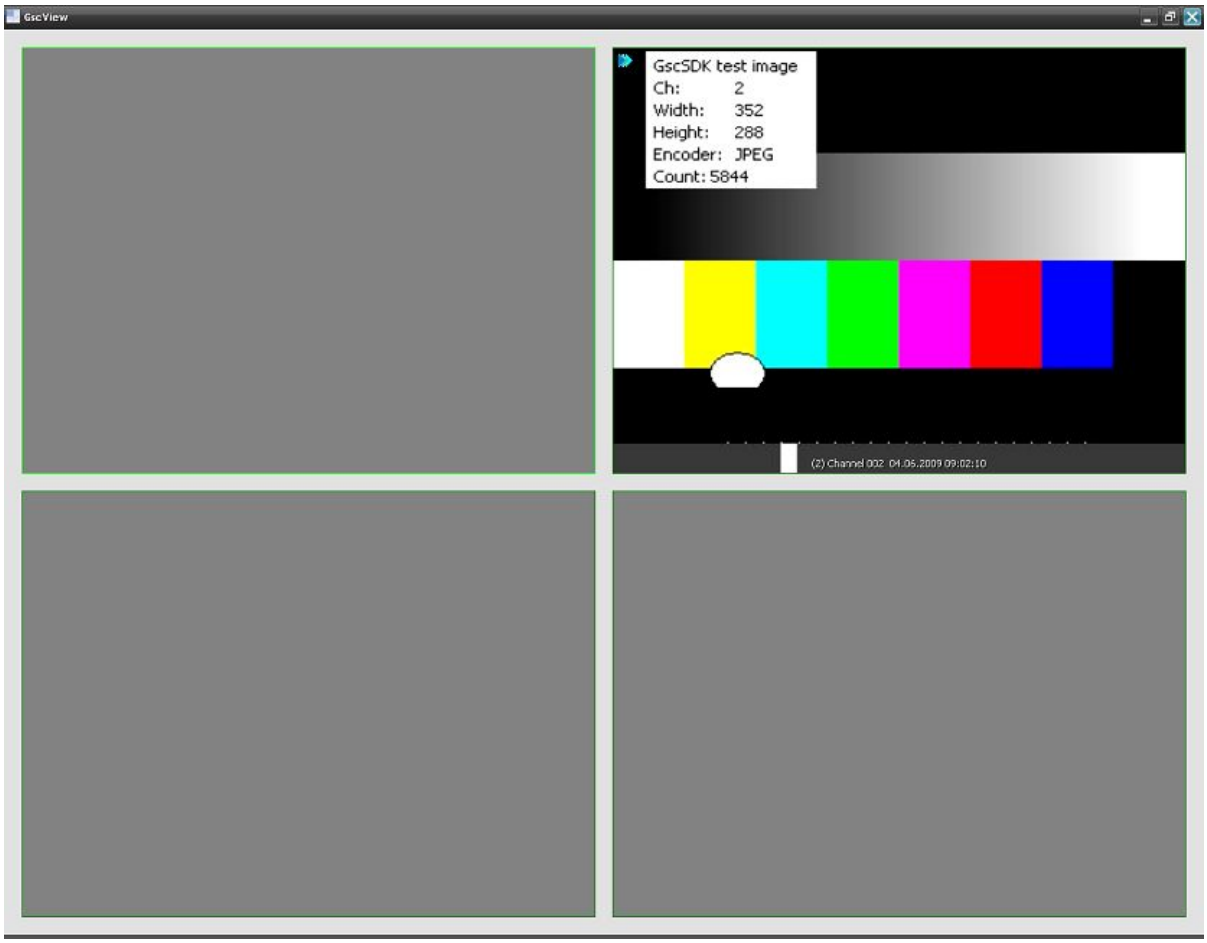
After sending the action, GSCView should display an “empty” “MyScene”.



To display video channels in the viewers of “MyScene” the action “Viewer connect live” can be used. The parameter “viewer” now means the global number of a viewer of “MyScene”, e.g. 1102. The parameter “channel” should be set to the global number of the video channel that should be displayed, e.g. 2.



After sending the action, GSCView displays live video of the video channel 2 on the upper left viewer in GSCView.



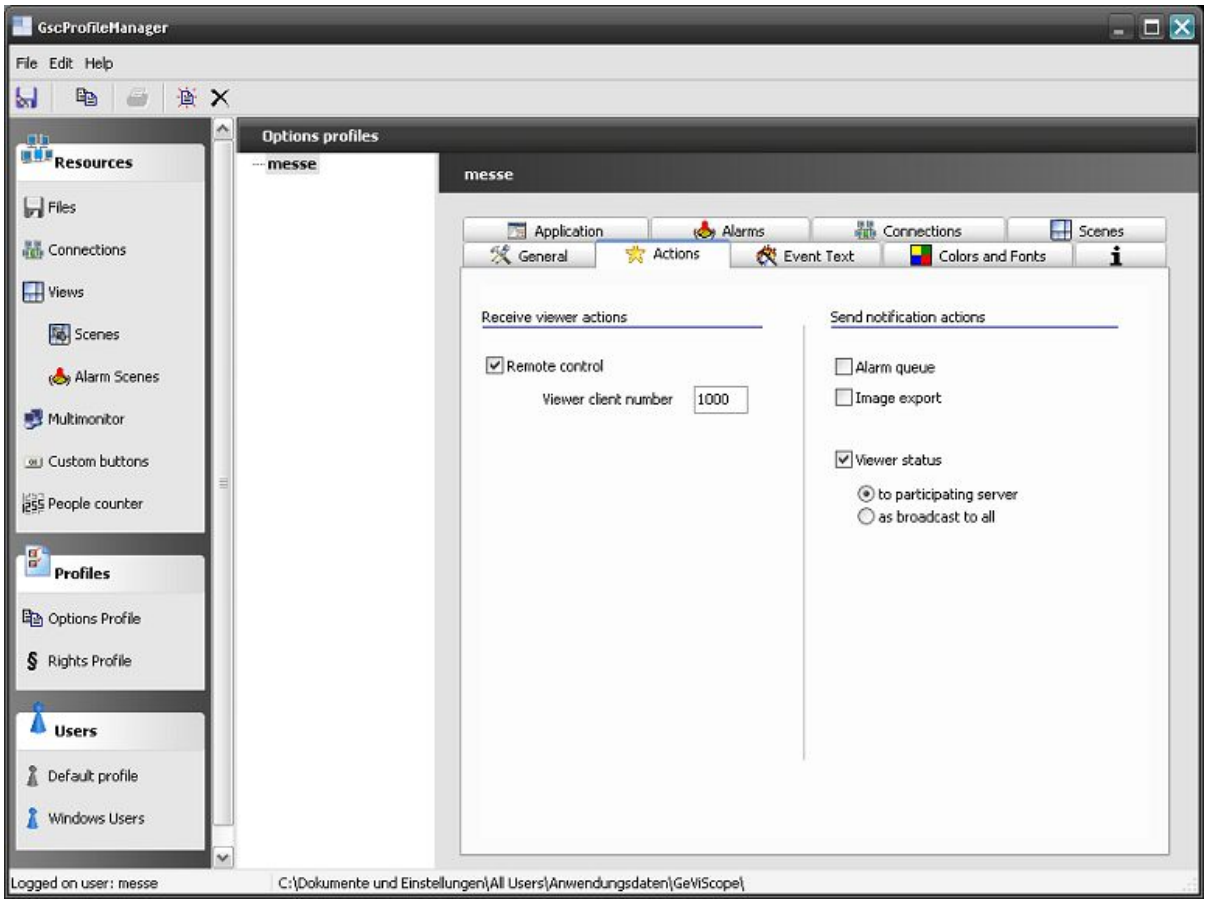
Background information

In GeViScope systems actions are used to communicate between the GeViScope 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 GeViScope server and they don’t directly result in any reaction in the GeViScope server, for example “Custom action”).

All actions are grouped in different categories. The category “Viewer actions” contains all actions that are relevant for remote controlling GSCView.

To get notifications about GSCView activities, one of the options “Send notification actions” in the profile manager of GSCView 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 GSCView 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”). GSCView does not have an input queue for remote control actions.

Supported development platforms

The SDK is designed and tested to be used with the following development environments:

- CodeGear C++ Builder 6 ©
- CodeGear C++ Builder 2009 ©
- CodeGear Delphi 7 ©
- CodeGear Delphi 2005 ©
- CodeGear Delphi 2009 ©
- Microsoft Visual Studio 2005, C++, MFC ©
- Microsoft Visual Studio 2008, C++, MFC ©
- Microsoft Visual Studio 2005, C++/CLI ©
- Microsoft .NET © (wrapper classes are contained in the “Examples” folder)

Guidelines and hints

Introduction

It is recommended to be familiar with the GeViScope system and the possibilities of modern video surveillance systems and video management systems. Before starting programming your custom GeViScope client you should know basics of video formats, video compression, GeViScope events, GeViScope actions and the principles of a client - server network communication.

The following sections support you with some suggestions and hints about using the SDK interfaces.

General hints

If your application needs to listen to events and actions please use the application PLCSimulator.exe that you can find on Your GeViScope device. This software allows you to start actions and events which might be used by your program.

You should work and do some tests with a real GeViScope device or with the virtual test environment belonging to the SDK. Create some events and actions, start them with PLCSimulator.exe.

Starting the setup software GSCSetup.exe with the command line parameter /utilities will offer you the possibility to open DBITest to discover the database structure and to evaluate and test select statements against the database. Additionally this tool offers you the possibility to start the registry editor to evaluate the internal structure of the GeViScope setup.

```
Make sure to delete all objects that are created inside of DLLs. The objects
themselves should always offer a Destroy() or Free() method for that.
```

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 may 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 were 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 DVD player software like PowerDVD or the VCL Media Player software.

Working with handles and instances

Integral part of the SDK are units that give the user a comfortable access to the plain functions of the DLL, e.g. GSCDBI.h/.cpp/.pas. 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 work with instances is finished, the instances have to be deleted by calling their destroy() or free() method. Otherwise there will be memory leaks left.

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

The following example (in pseudo code) should illustrate the above facts:

```
// define a handle to a server object
HGscServer MyServer;

// create a server object instance inside the DLL and
// get a handle to it
MyServer = DBICreateRemoteserver();

...

// work with the object instance with the help of the handle
MyServer->Connect();

...

// define a handle to a PLC object
HGscPLC PLC;

 // create a PLC object instance inside the DLL and
// get a handle to it
PLC = MyServer.CreatePLC();

...

// work with the object instance with the help of the handle
PLC->OpenPushCallback(...);

...

// destroy PLC object
PLC->Destroy();

...

// destroy server object
MyServer->Destroy();
```

Interaction between DBI and MediaPlayer

The DBI interface gives access to GeViScope 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.

The following methods of a server object instance can be called to get access to different kinds of functions (not a complete list):

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

The example (in pseudo code) of the previous chapter should illustrate the above facts.

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 needs a handle to a windows control and a handle to a server object instance. It handles fetching data, decompressing data and displaying video in the linked windows control by itself.

The following methods of a viewer object instance can be called to get access to different kinds of functions (not a complete list):

Method	Function
ConnectDB()	Fetch video data from the database and display it in any play mode required. Filter and search criteria can optionally be defined.
SetPlayMode (pmPlayNextEvent)	Display the next available event pictures

The following example (in pseudo code) shows how to create a viewer and use it afterwards:

```
// define a handle to a viewer object
HGscViewer MyViewer;

// create a viewer object instance inside the DLL and
// get a handle to it
MyViewer = GMPCreateViewer(WindowHandle, ...);

// define a structure with data needed to link
// the viewer to a media channel in the server
TMPConnectData MyViewerConnectData;
 // handle to the server object instance
MyViewerConnectData.Connection = MyServer;
MyViewerConnectData.ServerType = ctGSCServer;
MyViewerConnectData.MediaType = mtServer;
// ID of the media channel that should be displayed
MyViewerConnectData.MediaChID = ...

// link the viewer to a media channel and display live data
MyViewer->ConnectDB(MyViewerConnectData, pmPlayStream, ...);

// destroy viewer object
 MyViewer->Destroy();
```

Beside the viewer object class there is another class in the MediaPlayer interface: The off-screen viewer object class. If you want to decompress media, which should not be

displayed with the help of the viewer object, you can use the offscreen 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.

The DecompBuffer class encapsulates special functions for effective decompressing. So it is recommend to use it. Creating an instance of the buffer can be reached 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.

Here is a short example (in pseudo code) how to work with an offscreen viewer object:

```
// define a handle to a DecompBuffer object
HGscDecompBuffer MyDecompBuffer;

// create a DecompBuffer object instance inside the DLL and
// get a handle to it
MyDecompBuffer = GMPCreateDecompBuffer();

// define a handle to a offscreen viewer object
HGscViewer MyOffscreenViewer;

// create an offscreen viewer object instance inside the DLL and
// get a handle to it
MyOffscreenViewer = GMPCreateOffscreenViewer(MyDecompBuffer);

// set callback of the offscreen viewer object
MyOffscreenViewer.SetNewOffscreenImageCallBack(NewOff-
screenImageCallback);

// define a structure with data needed to link
// the offscreen viewer to a media channel in the server
TMPConnectData MyOffscreenViewerConnectData;
// handle to the server object instance
MyOffscreenViewerConnectData.Connection = MyServer;
MyOffscreenViewerConnectData.ServerType = ctGSCServer;
MyOffscreenViewerConnectData.MediaType = mtServer;
// ID of the media channel that should be decompressed
MyOffscreenViewerConnectData.MediaChID = ...

// link the offscreen viewer to a media channel and decompress live data
MyOffscreenViewer->ConnectDB(MyOffscreenViewerConnectData, pmPlayStream,
...);

...

// destroy offscreen viewer object
MyOffscreenViewer->Destroy();

// destroy DecompBuffer object
```

```
MyDecompBuffer->Destroy();

...

// callback function, that is called after images have been decompressed

...

// get a raw pointer to the picture in the DecompBuffer
// object
MyDecompBuffer->GetBufPointer(BufferPointer, ...);

// copy the picture into a windows bitmap resource
// for example
SetDIBits(..., BitmapHandle, ..., BufferPointer, ..., DIB_RGB_COLORS);

...
```

Enumeration of setup data

GeViScope 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 GSCSetupReadRequest 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().

Here is an example (in pseudo code), that shows how to enumerate the media channels:

```
...

// connect to the server
MyServer->Connect();

...

// define a handle to a setup object
HGscRegistry MySetup;

// create a setup object instance inside the DLL and
// get a handle to it
MySetup = MyServer->CreateRegistry();

// define a array for the setup read request
GscSetupReadRequest SetupReadRequest[1];
SetupReadRequest[0].NodeName = "/";
```

```
// read the setup data from the server
MySetup->ReadNodes(&SetupReadRequest, ...);

// define a GUID array for the GUIDs of the
// existing media channels
GuidDynArray MediaChannels;

// get the GUID array out of the setup data
MySetup->GetMediaChannels(MediaChannels);

// get the data of each single media channel
for each MediaChannelGUID in MediaChannels
      MySetup->GetMediaChannelSettings(MediaChannelGUID,
                                       MediaChannelID,
                                       GlobalNumber,
                                       ...);

...

// destroy setup object
MySetup->Destroy();

// destroy server object
MyServer->Destroy();

...
```

Please note that especially the media channels can be enumerated by using the global function GMPQueryMediaChannelList() of the MediaPlayer interface as well.

PLC, actions and events

The PLC (Prcess 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.

The following methods of a PLC object instance can be called to get access to different kinds of functions (not a complete list):

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

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 the an instance of the class GSCActionDispatcher. The action dispatcher gives you a simple way to react on specific actions. Here is a short example (in pseudo code):

```
// initialization code:

    ...

    // connect to the server
    MyServer->Connect();

    ...

    // define a handle to a PLC object
    HGSCPLC PLC;

    // create a PLC object instance inside the DLL and
    // get a handle to it
    PLC = MyServer.CreatePLC();

     ...

    // link your callback function for a custom action
    // to the action dispatcher, so that the callback function
    // is called automatically if a cutsom action arrives
    ActionDispatcher->OnCustomAction = this->MyCustomActionHandler;

    // register a callback function for notifications,
    // events and actions (this callback function dispatches
    // all received actions with the help of the
    // GSCActionDispatcher)
    PLC->OpenPushCallback(...);

    ...

    // destroy PLC object
    PLC->Destroy();

    ...

    // destroy server object
    MyServer->Destroy();

// callback function for all notifications, events and
// subscribed actions:

    ...

    // dispatch the received action to the linked
    // callback functions
    ActionDispatcher->Dispatch(ActionHandle);

    ...
```

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 GeViScope server. Every created media channel gets an ID that is always unique. So if you remove media channels from the setup and add them again, they will sure receive some new IDs.

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 “Enumeration of setup data” in this document to see how this is done.

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

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. It is fired 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 a phase of reconnect tries. The reconnect tries should start every 30 seconds for example. Additionally your application can listen to UDP broadcasts that are sent by the GeViScope 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.

Using MediaPlayer with GeViScope and MULTISCOPE III servers

Generally the MediaPlayer interface can be used with GeViScope as well as MULTISCOPE III servers. To link the server connection to the viewer object, the connection data structure has to be defined. The type of the structure is “TMPConnectData”. The element “ServerType” identifies the kind of server whose media should be displayed in the viewer.

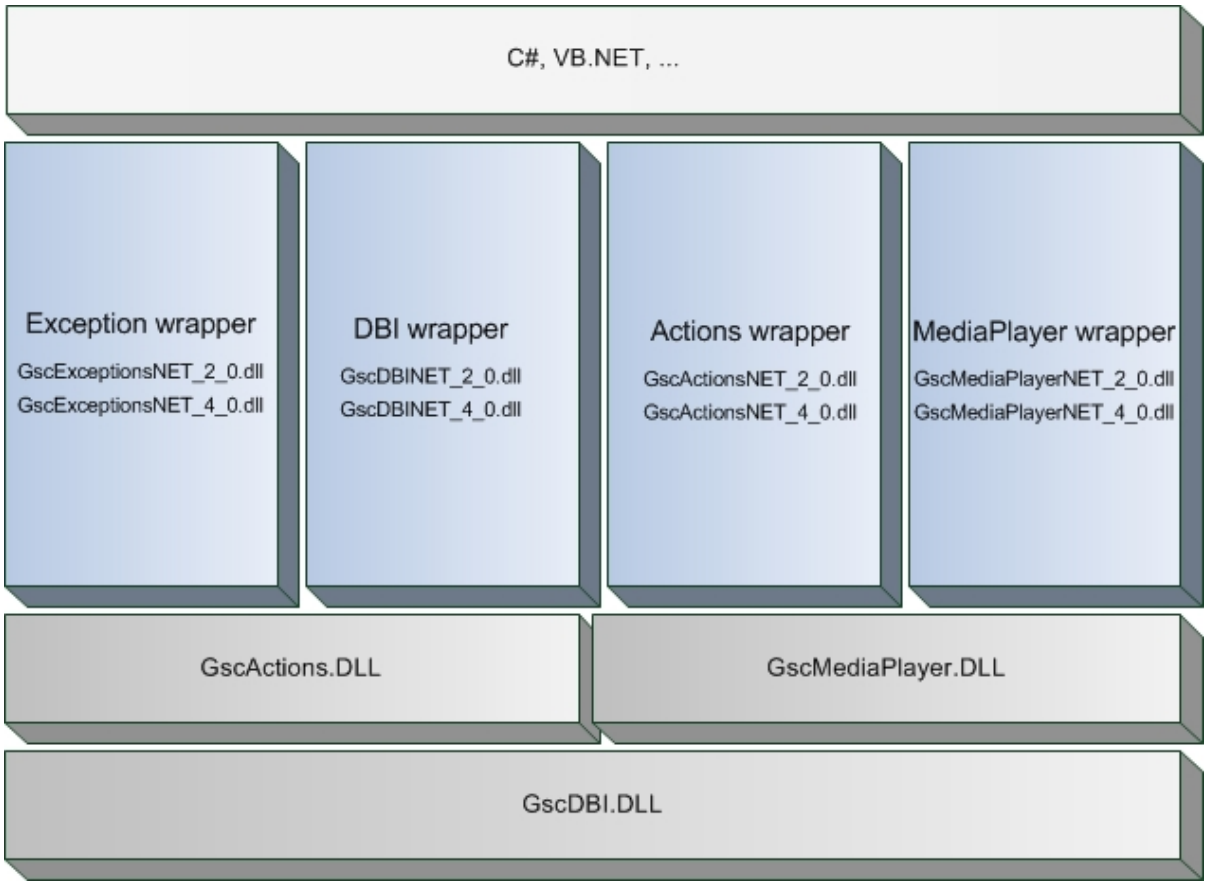
Please have a look on the example (in pseudo code) in the chapter “Interaction between DBI and MediaPlayer” in this document.

For creating different kind of connections, different DLLs have to be used. For GeViScope the DLL “GSCDBI.DLL” and for MULTISCOPE III the DLL “MscDBI.DLL” has to be included in the project or solution of the custom application. They can coexist.

Handling a connection to a MULTISCOPE III server is similar to GeViScope. Details can be found in the MULTISCOPE III SDK documentation.

Using the SDK with .NET

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



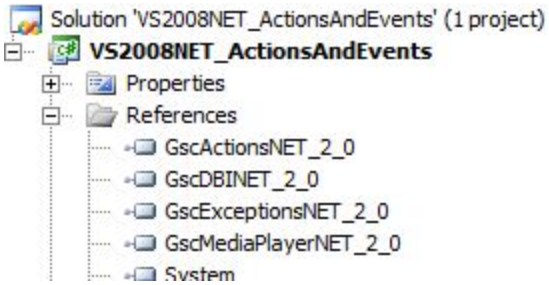
These wrapper assemblies are developed in C++/CLI and published with the SDK. The assemblies can be found in the GeViScope SDK binary folder “GeViScopeSDK\BIN”.

The SDK provides wrapper assemblies for the .NET-Frameworks versions 2.0 and 4.0 which are named as follows:

.NET-Framework 2.0
• GscExceptionsNET_2_0.dll
• GscActionsNET_2_0.dll
• GscMediaPlayerNET_2_0.dll
• GscDBINET_2_0.dll

.NET-Framework 4.0
• GscExceptionsNET_4_0.dll
• GscActionsNET_4_0.dll
• GscMediaPlayerNET_4_0.dll
• GscDBINET_4_0.dll

These wrapper assemblies can be used together with our native SDK DLLs (GscActions.DLL, GscDBI.DLL, GscHelper.DLL, GscMediaPlayer.DLL, MscDBI.DLL) to create custom applications under any .NET language on a windows platform. The assemblies need to be referenced by the .NET project and all the files (assemblies and native DLLs) have to reside in the application folder.



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:

a) Microsoft Visual C++ Redistributable Package has to be installed

The wrapper assemblies are developed in C++/CLI. So for executing them on a none development machine, the Microsoft Visual C++ Redistributable Package is needed. This package exists in a debug or in a release version. On productive machines the release version needs to be installed.

For applications using the .NET-Framework 2.0 the Visual C++ 2008 Redistributable Package is needed. In case that the application is developed using the .NET-Framework 4.0 you need to install the Visual C++ 2010 Redistributable Package.

b) .NET Framework Version 2.0 SP 1 or newer has to be installed

If updating the .NET Framework on a GEUTEBRÜCK device (GeViScope or re_porter) fails, a special Microsoft tool Windows Installer CleanUp Utility (MSICUU2.exe) can improve the situation. After executing this tool, updating the Framework should be possible.

c) Wrapper assemblies AND native SDK DLLs are needed

Beside the custom application also the wrapper assemblies and the native SDK DLLs (listed above) are needed in the same folder as in which the custom application resides.

If the application uses the .NET-Framework 4.0 you need to reference the GeViScope wrapper DLLs with the extension _4_0 otherwise please use the wrapper assemblies with the extension _2_0 (see above).

GeViScope REGISTRY

Using the GscRegistry with .NET

Introduction

By using the GeViScope registry (GSCREGISTRY) it is possible to modify GeViScope/Re_porter settings programmatically. The GscRegistry is a proprietary registry format developed by GEUTEBRÜCK. This registry format is similar to the Microsoft Windows registry.

All needed GeViScope server settings are stored in the GscRegistry database. The creation of own registry databases based on files is also possible.

The GEUTEBRÜCK GEVISCOPE SDK provides several classes and methods to allow a comfortable access to the GscRegistry.

Requirements
The following requirements are needed to create a .NET application that uses the GscRegistry functionality:

• .NET-Framework 2.0 SP1 or newer
  - .NET-Framework 2.0 SP1 Wrapper-Assemblies:
    GscExceptionsNET_2_0.dll
    GscDBINET_2_0.dll
  - .NET-Framework 4.0 Wrapper-Assemblies:
    GscExceptionsNET_4_0.dll
    GscDBINET_4_0.dll
• Native Win32-DLLs, used by the .NET-Wrapper:
  - GscActions.dll
  - GscDBI.dll
  - GscMediaPlayer.dll
  - GscHelper.dll
  - MscDBI.dll
• Microsoft Visual C++ Redistributable Package

Using the registry
In the following, the usage of the GscRegistry with .NET is explained in detail. It discusses the following steps:

- Open the registry
- Read values out of nodes
- Create a node
- Add values to a node
- Save the registry

All necessary classes and methods for using the GscRegistry are available in the GscDBI namespace. To include this namespace the following using-statement is needed:

```
using GEUTEBRUECK.GeViScope.Wrapper.DBI;
```

Open the registry

To read or modify GeViScope/Re_porter settings it is necessary to establish a connection to the preferred GeViScope/Re_porter server before. After this is done you need to create a new object of the class GscRegistry and initialize it by using the CreateRegistry() method which is contained in the GscServer object.

C#-Code: Open the registry

```
if (_GscServer != null)
{
    // create an object instance of the server registry
    GscRegistry GscRegistry = _GscServer.CreateRegistry();
    if (GscRegistry != null)
    {
        // define an array for the setup read request (registry node paths
        to read)
```

```
        GscRegistryReadRequest[] ReadRequests = new GscRegistryReadRequest
        [1];
        ReadRequests[0] = new GscRegistryReadRequest("/", 0);
        // read the nodes (setup data) out of the server registry
        GscRegistry.ReadNodes(ReadRequests);
    }
}
```

The method *ReadNodes()* of the GscRegistry object expects an array of the type *GscRegistryReadRequest* which contains all node paths to be read out of the registry. In the source code snippet above, the array simply contains one element which represents the root node (“/”). By reading the root node the entire registry will be read out.

Read values of nodes

The following source code snippet shows how to read values out of nodes:

C#-Code: Read values out of nodes

```
if (GscRegistry != null)
{
    GscRegNode RegNode = GscRegistry.FindNode("/System/MediaChannels/");

    for (int i = 0; i < RegNode.SubNodeCount; ++i)
    {
        // find the GeViScope registry node of the parent node by means of
        the index
        GscRegNode SubRegNode = RegNode.SubNodeByIndex(i);
        GscRegVariant RegVariant = new GscRegVariant();

        // Get the value "Name" out of the sub registry type and store the
        value and
        // value type in the GscRegVariant class
        SubRegNode.GetValueInfoByName("Name", ref RegVariant);

        if (RegVariant != null && RegVariant.ValueType ==
        GscNodeType.ntWideString)
        Console.WriteLine(RegVariant.Value.WideStringValue);
    }
}
```

To read a specific node out of the registry the *GscRegistry* class provides the method *FindNode()*.

For that the path to the preferred node has to be committed to the method and it you will get back an object of the type of *GscRegNode*. This object contains all sub nodes and values of the found node.

To access a sub node of the parent node the method *SubNodeByIndex()* provided by the class *GscRegNode* can be used or use the *SubNodeByName()* method if the name of the sub node is already known.

The method *GetValueInfoByName()* can be used to access a specific value of a node. This method expects the name of the specific value as well as a reference to an object of type of *GscRegVariant*. The *GscRegVariant* object will be filled with the type of the value (ValueType) as well as the value itself (Value).

Create a node

To create a new node in a parent node the method *CreateSubNode()* which is provided by the class *GscRegNode* needs to be called. The method expects the name of the new node.

C#-Code: Create a node

```
if (_GscRegistry != null)
{
    GscRegNode RegNode = _GscRegistry.FindNode("/System/MediaChannels/0000");

    // create a new sub node in NodePath
    if (RegNode != null)
    RegNode.CreateSubNode("NewNode");
}
```

Add values to a node

There are several methods in the class *GscRegNode* to add values to a node. Depending on the type of the value it is needed to call the right method for writing this type into the registry. For example if you would like to write an Int32 value into the registry you need to use the method *WriteInt32()*.

C#-Code: Add values to node

```
public void AddValue(string NodePath, string ValueName, GscNodeType ValueType,
object Value)
{
    GscRegNode RegNode = _GscRegistry.FindNode(NodePath);

    if (RegNode != null)
    {
        switch (ValueType)
        {
            case GscNodeType.ntWideString:
            {
                RegNode.WriteWideString(ValueName, Value.ToString());
                break;
            }
            case GscNodeType.ntInt32:
            {
                RegNode.WriteInt32(ValueName, Convert.ToInt32(Value));
                break;
            }
        }
    }
}
```

Save the registry

After the *GscRegistry* object was modified (e.g. new nodes/new values), the server also needs to know about the changes made. For this the GscRegistry class provides the method *WriteNodes()*.

C#-Code: Add values to node

```
// define an array for the setup write request
```

```
GscRegistryWriteRequest[] WriteRequests = new GscRegistryWriteRequest[1];
WriteRequests[0] = new GscRegistryWriteRequest("/", 0);
GscRegistry.WriteNodes(WriteRequests, true);
```

The *WriteNodes()* method expects an array containing objects of the type of *GscRegistryWriteRequest*. Each GscRegistryWriteRequest contains a path to a node that has to be saved.

NOTICE

It is recommended to only add one element to this array which contains the root path ("/"). This results in saving the entire registry structure.

Structure of GSCRegistry
The GEVISCOPE SDK offers two possibilities to browse the structure of the *GscRegistry*. By means of the application *GscRegEdit* that is delivered with the SDK, it is possible to browse or modify the registry similar to Microsoft's Windows registry.

In addition to *GscRegEdit* you can also use the registry editor which is integrated in GSCSetup. To activate this feature the key combination *STRG+ALT+U* needs to be actuated. The entry *Registry editor* in the section *Utilities* in the navigation bar on the left will now be shown.

Examples
To get a better idea of how to use the GscRegistry, the GEVISCOPE SDK provides further .NET example applications.

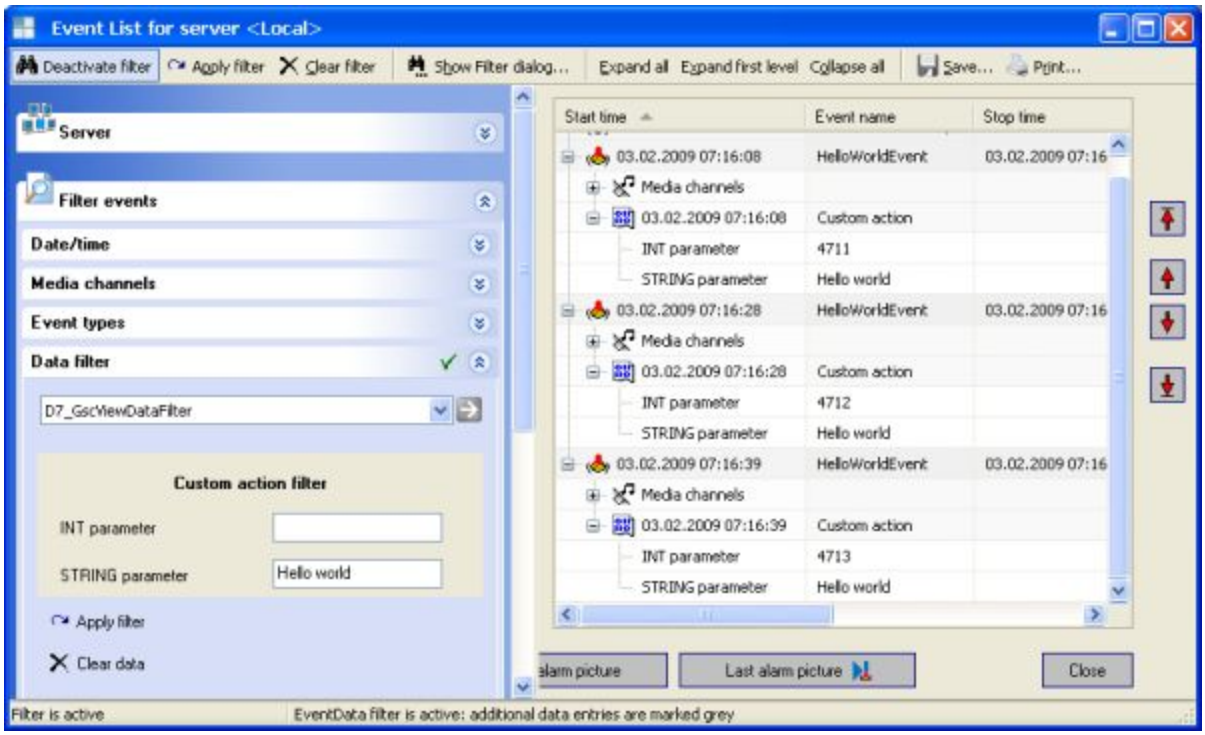
The examples can be found in the folder „Examples" folder in the GeViScopeSDK main folder:

- C:\Program Files (x86)\GeViScopeSDK\Examples\VS2008NET\VS2008NET_GscRegEdit
  Simple registry editor, GUI application (Visual Studio 2008)
- C:\Program Files (x86)\GeViScopeSDK\Examples\VS2008NET\VS2010NET_GscRegEdit
  Simple registry editor, GUI application (Visual Studio 2010)
- C:\Program Files (x86)\GeViScopeSDK\Examples\VS2008NET\VS2008NET_GscRegistryBasics
  Console application (Visual Studio 2008)
- C:\Program Files (x86)\GeViScopeSDK\Examples\VS2010NET\VS2010NET_GscRegistryBasics
  Console application (Visual Studio 2010)

GSCView data filter plugins

Introduction

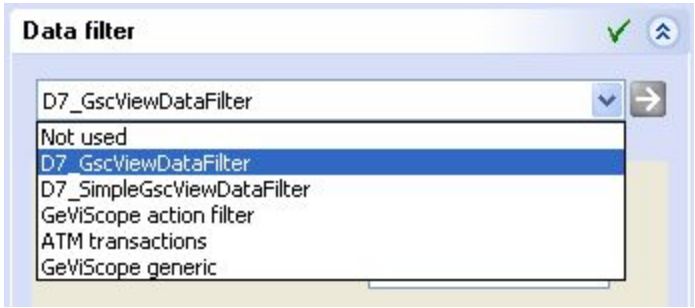
GSCView offers the possibility to integrate customized data filter dialogs. Data filter dialogs are used to search and filter video footage by additional event data. They can be customized to the different business environments in which GeViScope is used.



The following sections support you with some suggestions and hints about creating customized data filter plugins.

General hints

Custom data filters are hosted in flat windows 32Bit dynamic link libraries. Differing from normal DLLs the data filter DLLs have the extension ".GPI". All data filter DLLs existing in the same folder as GSCView are integrated in GSCView automatically.



The customized data filter DLL interface

Each DLL has to export the function GSCPluginRegisterSearchFilter() that is called by GSCView to use the customized dialogs. The exact definition of this function and some additional type definitions can be found in the unit "GSCGPIFilter.pas/.h".

Inside the function GSCPluginRegisterSearchFilter() one or even more data filter dialogs have to be registered by calling the function Callbacks.RegisterFilter().

The following example (in pseudo code) shows how this is done:

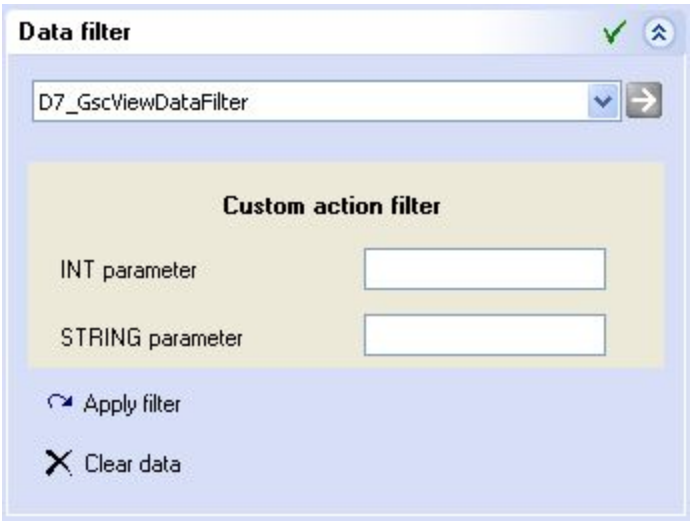
```
if(Callbacks.RegisterFilter == NULL)
```

```
        return FALSE;

    TPluginFilterDefinition def;

    def = SimpleFilter.GetFilterDefinition();
    Callbacks.RegisterFilter(Callbacks.HostHandle, def);
```

The structure TPluginFilterDefinition defines some informational data and all the callback functions needed for a single dialog. GSCView uses the definition to call the different call-back functions during its execution.

Name of callback function	Function
InitFilter()	Can be used to initialize the data filter dialog. To integrate the dialog in GSCView, the function has to return true.
ShowFilter()	Inside this function the dialog should be displayed as a stand-alone (modal) dialog. GSCView calls the function after the user activates the button.
DeinitFilter()	Can be used to deinitialize the data filter dialog. The function has to return true, even if it is not used.
GetFilterGuid()	The function should provide a global unique identifier (GUID) that is used inside GSCView to identify the dialog. The GUID can be defined as a static constant value.

As an alternative to the modal display of the data filter dialog, the dialog can be displayed nested in the GSCView main window or GSCView event list. But at the moment this feature is only supported by custom filter dialogs created with Borland Delphi ©.



To achieve the nested display, the additional callback functions of the structure TPluginFilterDefinition have to be implemented. The Borland Delphi © example “GSCViewDataFilter” demonstrates the details.

Creating the filter criteria

If the custom data filter is applied, GSCView does a query against the tables “events” and “eventdata” of the internal GeViScope database. For this query a filter criteria is needed. The

custom data filter delivers the criteria and gives it back to GSCView in the ShowFilter() call-back function.

To build up meaningful filter criteria some background knowledge of the GeViScope database is needed.
The table “events” contains all the events recorded in the database (only event information, not the samples; the samples are linked to the events).

The table “eventdata” contains additional data belonging to the events. Inside the table the different parameters of actions are saved. If for example an event is started by the CustomAction(4711, “Hello world”), the value 4711 is saved in the row “Int64_A” and the value “Hello world” is saved in the row “String_A”. Because the event is started by a CustomAction, the value 8 is saved in the row “EventDataKind”. Each action has an individual mapping of action parameters to rows in the table “eventdata”.

For different business environments special actions can be created by GEUTEBRÜCK. There already exist some special actions like:

Action name	Business environment
ATMTransaction()	Automated teller machines
ACSAccessGranted()	Access control systems
SafebagOpen()	Cash management systems
POSData()	Point of sale systems

The action internally defines the mapping of action parameters to rows in the table “eventdata”. The code of an action (for a CustomAction the code is 8) is stored in the row “EventDataKind”. The codes of actions are listed in the action reference documentation “GSCActionsReference_EN.pdf”.

To evaluate the mapping of action parameters to database rows, GSCSetup can be used. By pressing STRG+ALT+U in GSCSetup the special utility “DBI test” gets available.

With “DBI test” the structure and content of the GeViScope database can be analyzed. The following SQL queries can be helpful:

SQL query	Function
select * from events	Fetches records from the table “events”
select * from eventdata	Fetches records from the table “eventdata”
select * from samples	Fetches records from the table “samples”

The following table should demonstrate how to build up filter criteria depending on parameters given in the custom data filter dialog (here the CustomAction() is used to start the events):

Action para-meter INT	Action para-meter STRING	Fil-terCriteria.SQLstatement	SQL query
Nothing	Nothing	EventData.EventDataKind = 8	select * from EventData left join Events on EventData.EventID = Events.EventID with EventData.EventDataKind = 8
Nothing	Hello world	EventData.EventString_A = "Hello world" and EventData.EventDataKind = 8	select * from EventData left join Events on EventData.EventID = Events.EventID with EventData.EventString_A = "Hello world" and EventData.EventDataKind = 8
4711	Nothing	EventData.EventInt64_A = 4711 and EventData.EventDataKind = 8	select * from EventData left join Events on EventData.EventID = Events.EventID with EventData.EventInt64_A = 4711 and EventData.EventDataKind = 8
4711	Hello world	EventData.EventInt64_A = 4711 and EventData.EventString_A = "Hello world" and EventData.EventDataKind = 8	select * from EventData left join Events on EventData.EventID = Events.EventID with EventData.EventInt64_A = 4711 and EventData.EventString_A = "Hello world" and EventData.EventDataKind = 8
Nothing	Hello*	EventData.EventString_A = "Hello*" and EventData.EventDataKind = 8	select * from EventData left join Events on EventData.EventID = Events.EventID with EventData.EventDataKind = 8 where EventData.EventString_A LIKE "Hello*"

During testing the custom data filter dialog in the GSCView event list a double click on the status bar of the event list delivers the SQL query that is executed in the GeViScope server.

Examples overview

The examples overview is organized in two different views on all examples including the GeViScopeSDK:

Examples grouped by programming tasks

Examples grouped by development platforms

Examples grouped by programming tasks

Connect to and disconnect from a GeViScope server

- LiveStream (CodeGear C++ Builder 6 and 2009)
- SimpleClient (CodeGear Delphi 7, 2005 and 2009)
- GSCLiveStream (Microsoft Visual Studio 2005, C++, MFC)
- VS2008CPP_SimpleClient (Microsoft Visual Studio 2008, C++, MFC)
- VS2008CPP_ActionsAndEvents (Microsoft Visual Studio 2008, C++, MFC)
- VS2008NET_SimpleClient (Microsoft Visual Studio 2008, C#)
- VS2008NET_ActionsAndEvents (Microsoft Visual Studio 2008, C#)
- VS2008WPF_SimpleClient (Microsoft Visual Studio 2008, C#, WPF)
- VS2010NET_SimpleClient (Microsoft Visual Studio 2010, C#)
- VS2010NET_ActionsAndEvents (Microsoft Visual Studio 2010, C#)
- VS2010WPF_SimpleClient (Microsoft Visual Studio 2010, C#, WPF)

Enumerate existing media channels and event types from a GeViScope server

- LiveStream (CodeGear C++ Builder 6 and 2009)
- SimpleClient (CodeGear Delphi 7, 2005 and 2009)
- GSCLiveStream (Microsoft Visual Studio 2005, C++, MFC)
- VS2008CPP_SimpleClient (Microsoft Visual Studio 2008, C++, MFC)
- VS2008CPP_ActionsAndEvents (Microsoft Visual Studio 2008, C++, MFC)
- VS2008NET_GscRegEdit (Microsoft Visual Studio 2008, C#)
- VS2008NET_GscRegistryBasics (Microsoft Visual Studio 2008, C#)
- VS2008NET_SimpleClient (Microsoft Visual Studio 2008, C#)
- VS2010CPP_ConfigReader (Microsoft Visual Studio 2010, C++)
- VS2010NET_SimpleClient (Microsoft Visual Studio 2010, C#)
- VS2010NET_ActionsAndEvents (Microsoft Visual Studio 2010, C#)
- VS2010NET_GscRegEdit (Microsoft Visual Studio 2010, C#)
- VS2010NET_GscRegistryBasics (Microsoft Visual Studio 2010, C#)
- VS2010WPF_SimpleClient (Microsoft Visual Studio 2010, C#, WPF)

Display live and recorded media with the MediaPlayer interface

- LiveStream (CodeGear C++ Builder 6 and 2009)
- SimpleClient (CodeGear Delphi 7, 2005 and 2009)
- GSCLiveStream (Microsoft Visual Studio 2005, C++, MFC)
- VS2008CPP_SimpleClient (Microsoft Visual Studio 2008, C++, MFC)
- VS2008CPP_ActionsAndEvents (Microsoft Visual Studio 2008, C++, MFC)
- VS2008NET_SimpleClient (Microsoft Visual Studio 2008, C#)
- VS2008WPF_SimpleClient (Microsoft Visual Studio 2008, C#, WPF)
- VS2010NET_SimpleClient (Microsoft Visual Studio 2010, C#)
- VS2010WPF_SimpleClient (Microsoft Visual Studio 2010, C#, WPF)

Display recorded event media with the MediaPlayer interface

- VS2008CPP_ActionsAndEvents (Microsoft Visual Studio 2008, C++, MFC)
- VS2008NET_ActionsAndEvents (Microsoft Visual Studio 2008, C#)
- VS2010NET_ActionsAndEvents (Microsoft Visual Studio 2010, C#)

Handling actions and PLC notifications

- GSCActions (CodeGear C++ Builder 6 and 2009)
- SimpleClient (CodeGear Delphi 7, 2005 and 2009)
- ResourceStateMonitor (Delphi 2009)
- VS2008CPP_ActionsAndEvents (Microsoft Visual Studio 2008, C++, MFC)
- VS2008NET_ActionsAndEvents (Microsoft Visual Studio 2008, C#)
- VS2010CPP_ControlBlockingFilters (Microsoft Visual Studio 2010, C++)
- VS2010NET_ActionsAndEvents (Microsoft Visual Studio 2010, C#)

Handling events

- LiveStream (CodeGear C++ Builder 6 and 2009)
- GSCLiveStream (Microsoft Visual Studio 2005, C++, MFC)
- VS2008CPP_ActionsAndEvents (Microsoft Visual Studio 2008, C++, MFC)
- VS2008NET_ActionsAndEvents (Microsoft Visual Studio 2008, C#)
- VS2010NET_ActionsAndEvents (Microsoft Visual Studio 2010, C#)

Creating backups

- Backup (CodeGear Delphi 7 and 2009)

Synchronized display of more than one media channels

- SynchPlayback (CodeGear C++ Builder 6 and 2009)

Custom draw in viewers of MediaPlayer interface

- SynchPlayback (CodeGear C++ Builder 6 and 2009)
- VS2008CPP_SimpleClient (Microsoft Visual Studio 2008, C++, MFC)
- VS2008NET_SimpleClient (Microsoft Visual Studio 2008, C#)
- VS2010NET_SimpleClient (Microsoft Visual Studio 2010, C#)

Export picture data

- MediaPlayerExport (CodeGear Delphi 7 and 2009)
- MPEGExport (CodeGear Delphi 7 and 2009)
- VS2008NET_MediaPlayerExport (Microsoft Visual Studio 2008, C#)
- VS2010NET_MediaPlayerExport (Microsoft Visual Studio 2010, C#)

Control PTZ cams

- Telecontrol (CodeGear Delphi 7 and 2009)

Fetch a user blocking list from the server

- UserBlockingList (CodeGear C++ Builder 6 and 2009)

Decompress live and recorded media with the offscreen viewer

- OffscreenViewer (CodeGear Delphi 7 and 2009)
- VS2008CPP_OffscreenViewer (Microsoft Visual Studio 2008, C++, MFC)

- VS2008CPP_OffscreenViewer_Console (Microsoft Visual Studio 2008, C++)
- VS2008NET_OffscreenViewer (Microsoft Visual Studio 2008, C#)
- VS2010NET_OffscreenViewer (Microsoft Visual Studio 2010, C#)

Decompress raw live media by using the DBI

- VS2008CPP_RawLiveStreamDecompress (Microsoft Visual Studio 2008, C++, MFC)
- VS2008CPP_RawLiveStreamDecompress_Console (Microsoft Visual Studio 2008, C++)

Create a general service application

- WindowsService (CodeGear C++ Builder 6 and 2009)
- VS2008CPP_ServiceFrameworkDemo (Microsoft Visual Studio 2008, C++)
- VS2008NET_ServiceFrameworkDemo (Microsoft Visual Studio 2008, C#)
- VS2010NET_ServiceFrameworkDemo (Microsoft Visual Studio 2010, C#)

Full-duplex audio communication between GeViScope components

The AudioBackChannel GeViScope Server Plugin (Visual Studio 2010) is an example for a GeViScope Server plugin. It realizes a full-duplex audio communication between different GeViScope components. The full scope of operation can be found in the document *Audio Back Channel (ABC) Plugin documentation*.

Simulate media channels in GeViScope servers

The MCS (Media Channel Simulator) GeViScope Server Plugin (CodeGear C++ Builder 6) is another example for a GeViScope Server plugin. It shows how to channel media data inside the GeViScope system without using special video hardware. In addition the handling of actions inside a server plugin is demonstrated. The full scope of operation can be found in the document *MCS Documentation*.

Simulate a screen saver as a GeviScope Server Plugin

The DelphiScreenSaverPlugin GeViScope Server Plugin (CodeGear Delphi 7) is another example to demonstrate channeling media into a GeViScope Server with the help of a Server Plugin.

Provide a customized data filter dialog in GSCView

GSCView offers the possibility to integrate customized data filter dialogs. Data filter dialogs are used to search and filter video footage by additional event data. They can be customized to the different business environments in which GeViScope is used. Detailed information can be found in the document *GSCView data filter plugins*.
The following examples demonstrate how to create customized data filter dialogs:

- SimpleGSCViewDataFilter (CodeGear Delphi 7 and 2009)
- GSCViewDataFilter (CodeGear Delphi 7 and 2009)
- VS2008CPP_SimpleGSCViewDataFilter (Microsoft Visual Studio 2008, C++, MFC)

Presenting GEUTEBRÜCK Backup Files (GBF)

- VS2008CPP_SimpleGBFViewer (Microsoft Visual Studio 2008, C++, MFC)
- SimpleGBFViewer (CodeGear Delphi 2009)
- VS2008NET_SimpleGBFViewer (Microsoft Visual Studio 2008, C#)

Monitor the state of media channels (cameras)

- ResourceStateMonitor (CodeGear Delphi 2009)

Examples grouped by development platforms

CodeGear C++ Builder 6 and 2009 ©

- LiveStream
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels and event types from a GeViScope server
  - Display live and recorded media with the MediaPlayer interface
  - Handling events
- GSCActions
  - Connect to and disconnect from a GeViScope server
  - Handling actions
- SynchPlayback
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels and event types from a GeViScope server
  - Display live and recorded media with the MediaPlayer interface
  - Handling events
  - Synchronized display of more than one media channels
- UserBlockingList
  - Connect to and disconnect from a GeViScope server
  - Fetch a user blocking list from the server
- WindowsService
  - WindowsService (CodeGear C++ Builder 6 and 2009)
- The MCS(Media Channel Simulator) GeViScope Server Plugin is another example for a GeViScope Server plugin. It shows how to channel media data inside the GeViScope system without using special video hardware. In addition the handling of actions inside a server plugin is demonstrated. The full scope of operation can be found in the document *MCS Documentation*.

CodeGear Delphi 7, 2005 und 2009 ©

- SimpleClient
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels and event types from a GeViScope server
  - Display live and recorded media with the MediaPlayer interface
- Backup
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels from a GeViScope server
  - Creating backups
- MediaPlayerExport
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels from a GeViScope server
  - Export picture data
- MPEGExport
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels from a GeViScope server

Export picture data
- Telecontrol
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels and event types from a GeViScope server
  Display live and recorded media with the MediaPlayer interface
  Handling actions
  Control PTZ cams
- OffscreenViewer
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  Decompress live and recorded media
  Custom draw
- The SimpleGSCViewDataFilter and GSCViewDataFilter example are examples for customized data filter dialogs of GSCView. Detailed information can be found in the document *GSCView data filter plugins*.
- SimpleGBFViewer (only Delphi 2009)
  Open and close a GEUTEBRÜCK Backup Files (GBF)
  enumerate existing media channels in the GBF file
  Display media with the MediaPlayer interface
- ResourceStateMonitor (only Delphi 2009)
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  monitor the state of media channels (cameras)
  Handling actions

Microsoft Visual Studio 2005, C++, MFC ©

- GSCLiveStream
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  Display live and recorded media with the MediaPlayer interface

Microsoft Visual Studio 2005, C++, CLI ©

- The VSIPCamPlugin GeViScope Server Plugin is an example to show how simple it is to channel some pictures from an IP cam into a GeViScope server

Microsoft Visual Studio 2008, C++, MFC ©

- VS2008CPP_SimpleClient
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  Display live and recorded media with the MediaPlayer interface
  Custom draw
- VS2008CPP_OffscreenViewer
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  Decompress live and recorded media
  Custom draw
- VS2008CPP_ActionsAndEvents
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels and event types from a GeViScope server
  Display live and recorded media with the MediaPlayer interface
  Handling actions

Handling events
Display recorded event media with the MediaPlayer interface

- VS2008CPP_SimpleGBFViewer
  Open and close a GEUTEBRÜCK Backup Files (GBF)
  enumerate existing media channels in the GBF file
  Display media with the MediaPlayer interface
- The VS2008CPP_SimpleGSCViewDataFilter example is an example for a customized data filter dialog of GSCView. Detailed information can be found in the document *GSCView data filter plugins*.
- VS2008CPP_RawLiveStreamDecompress_Console
  Receiving live streams by using the DBI
  Decompressing frames by means of the decompressor object of the GscMediaPlayer-DLL
- VS2008CPP_OffscreenViewer_Console
  Using the OffscreenViewer to receive a live stream in a console application
  OffscreenViewer provides a decompressed image in a callback
  Only the picture ID (PicID) of the image will be displayed in the console
- VS2008CPP_RawLiveStreamDecompress_Console
  Receiving live streams by using the DBI
  Decompressing frames by means of the decompressor object of the GscMediaPlayer-DLL
- VS2008CPP_OffscreenViewer_Console
  Using the OffscreenViewer to receive a live stream in a console application
  OffscreenViewer provides a decompressed image in a callback
  Only the picture ID (PicID) of the image will be displayed in the console

Microsoft ActiveX ©

- GscViewer (ActiveX Control)
  Encapsulating of GeViScope functionality into an ActiveX control
- ActiveX_DOTNETClient
  Invocation of the GscViewer ActiveX control from C#
- ActiveX_HTML_Page
  Invocation of the GscViewer ActiveX control from inside a web page (html)
- ActiveX_VB6Client (deprecated)
  Invocation of the GscViewer ActiveX control from inside a VB6 application
- ActiveX_VB6MultiClient (deprecated)
  Invocation of several GscViewer ActiveX control from inside a VB6 application

Microsoft Visual Studio 2008, C# ©

- VS2008NET_SimpleClient
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  Display live and recorded media with the MediaPlayer interface
  Custom draw
- VS2008NET_ActionsAndEvents
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels and event types from a GeViScope server
  Display live and recorded media with the MediaPlayer interface
  Handling actions
  Handling events
  Display recorded event media with the MediaPlayer interface

- VS2008NET_OffscreenViewer
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels from a GeViScope server
  - Decompress live and recorded media
  - Custom draw
- VS2008NET_RawDBDecompress
  - Fetching database records
  - Decompressing the fetched records as fast as possible
- VS2008NET_MediaPlayerExport
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels from a GeViScope server
  - Export picture data
- VS2008NET_SimpleGBFViewer
  - Open and close a GEUTEBRÜCK Backup Files (GBF)
  - enumerate existing media channels in the GBF file
  - Display media with the MediaPlayer interface
- The VS2010NET_ServiceFrameworkDemo example
  - is an example for a general service application. Services based on the GEUTEBRÜCK
  - Service Framework behave like all GEUTEBRÜCK product
- VS2008NET_GscRegEdit
  - Simple GeViScope registry editor
  - Connect to a GeViScope server
  - Modify GeViScope settings using the GeViScope registry
  - Export settings to GeViScope registry file format
- VS2008NET_GscRegistryBasics
  - Simple demonstration in using the GeViScope registry
  - Reading out media channels
  - Add a value to the GeViScope registry
  - Saving the GeViScope registry

Microsoft Visual Studio 2008, C#, WPF ©

- VS2008WPF_SimpleClient
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels from a GeViScope server
  - Display live and recorded media with the MediaPlayer interface

Microsoft Visual Studio 2010, C++

- VS2010CPP_ConfigReader
- VS2010CPP_ControlBlockingFilters

Microsoft Visual Studio 2010, C# ©

- VS2010NET_SimpleClient
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels from a GeViScope server
  - Display live and recorded media with the MediaPlayer interface
  - Custom draw
- VS2010NET_ActionsAndEvents
  - Connect to and disconnect from a GeViScope server
  - enumerate existing media channels and event types from a GeViScope server
  - Display live and recorded media with the MediaPlayer interface

Handling actions
Handling events
Display recorded event media with the MediaPlayer interface

- VS2010NET_OffscreenViewer
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  Decompress live and recorded media
  Custom draw
- VS2010NET_RawDBDecompress
  Fetching database records
  Decompressing the fetched records as fast as possible
- VS2010NET_MediaPlayerExport
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  Export picture data
- VS2010NET_SimpleGBFViewer
  Open and close a GEUTEBRÜCK Backup Files (GBF)
  enumerate existing media channels in the GBF file
  Display media with the MediaPlayer interface
- The VS2010NET_ServiceFrameworkDemo
  example is an example for a general service application. Services based on the GEUTEBRÜCK Service Framework behave like all GEUTEBRÜCK product services.
- VS2010NET_GscRegEdit
  Simple GeViScope registry editor
  Connect to a GeViScope server
  Modify GeViScope settings using the GeViScope registry
  Export settings to GeViScope registry file format
- VS2010NET_GscRegistryBasics
  Simple demonstration in using the GeViScope registry
  Reading out media channels
  Add a value to the GeViScope registry
  Saving the GeViScope registry

Microsoft Visual Studio 2010, C#, WPF ©

- VS2010WPF_SimpleClient
  Connect to and disconnect from a GeViScope server
  enumerate existing media channels from a GeViScope server
  Display live and recorded media with the MediaPlayer interface

Action documentation

The following chapter contains a short overview about the existing GEUTEBRÜCK actions and there parameter descriptions.

ATM / ACS

ACS access denied

Action name:*ACSAccessDenied(ACSName, ACSNo, Account, BancCode, CardNo, TimeStamp, Reason)*Action category: logical ACS access denied.

Parameter		Function
ACS	*ACSName*	ACS name.
ACS no	*ACSNo*	ACS no.
account	*Account*	Account no.
bank code	*BancCode*	Bank code.
card no	*CardNo*	Card no.
time stamp	*TimeStamp*	Time stamp.
reason	*Reason*	Reason.

ACS access granted

Action name:*ACSAccessGranted(ACSName, ACSNo, Account, BancCode, CardNo, TimeStamp)*Action category: logical
ACS access granted.

Parameter		Function
ACS	*ACSName*	ACS name.
ACS no	*ACSNo*	ACS no.
account	*Account*	Account no.
bank code	*BancCode*	Bank code.
card no	*CardNo*	Card no.
time stamp	*TimeStamp*	Time stamp.

ACS raw answer

Action name:*ACSRawAnswer(ACSName, TimeStamp, ACSData)*Action category: logical
ACS raw answer.

Parameter		Function
ACS	*ACSName*	ACS name.
time stamp	*TimeStamp*	Time stamp.
answer	*ACSData*	ACS answer.

ACS raw data

Action name:*ACSRawData(ACSName, TimeStamp, ACSData)*Action category: logical

ACS raw data.

Parameter		Function
ACS	*ACSName*	ACS name.
time stamp	*TimeStamp*	Time stamp.
data	*ACSData*	ACS data.

ATM raw answer

Action name:*ATMRawAnswer(ATMName, TimeStamp, ATMData)*Action category: logical
ATM raw answer.

Parameter		Function
ATM	*ATMName*	ATM name.
time stamp	*TimeStamp*	Time stamp.
answer	*ATMData*	ATM answer.

ATM raw data

Action name:*ATMRawData(ATMName, TimeStamp, ATMData)*Action category: logical
ATM raw data.

Parameter		Function
ATM	*ATMName*	ATM name.
time stamp	*TimeStamp*	Time stamp.
data	*ATMData*	ATM data.

ATM transaction

Action name:*ATMTransaction(ATMName, NewTransaction, Photostep, ATMNo, Account, BancCode, CardNo, TAN1, TAN2, TimeStamp1, TimeStamp2, Amount, Currency)*Action category: logical ATM transaction.

Parameter		Function
ATM	*ATMName*	ATM name.
new transaction	*NewTransaction*	New transaction.
photostep	*Photostep*	Photostep.
ATM no	*ATMNo*	ATM no.
account	*Account*	Account no.
bank code	*BancCode*	Bank code.
card no	*CardNo*	Card no.
tan 1	*TAN1*	TAN 1.
tan 2	*TAN2*	TAN 2.
time stamp 1	*TimeStamp1*	Time stamp 1.
time stamp 2	*TimeStamp2*	Time stamp 2.
amount	*Amount*	Amount.
currency	*Currency*	Currency.

Audio control

All actions to control the audio streams, also all notifications about the state change of the audio streams.

ABC connect

Action name:*ABCConnect(Address)*Action category: logical Connect audio back channel.

Parameter		Function
address	*Address*	Address of the remote server.

ABC disconnect

Action name:*ABCDisconnect()*Action category: logical Disconnect audio back channel.

ABC play file

Action name:*ABCPlayFile(FileID, FileName, AutoRepeat)*Action category: logical Play file on audio back channel.

Parameter		Function
file id	*FileID*	File ID.
file name	*FileName*	Name of the file.
repeat	*AutoRepeat*	Repeat file automatically

Sensor audio alarm

Action name:*SensorAudioAlarm(Channel)*Action category: logical Audio alarm detected.

Parameter		Function
channel	*Channel*	Channel.

Backup actions

All actions for backup.

Abort all auto backups

Action name:*AbortAllAutoBackups()*Action category: logical Abort all auto backups.

Abort auto backup

Action name:*AbortAutoBackup(Schedule)*Action category: logical Abort auto backup.

Parameter		Function
schedule	*Schedule*	Schedule.

Auto backup capacity warning

Action name:*AutoBackupCapacityMonitoringCapacityWarning(Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByGbf, PercentFree, PercentAllocated, Per-*

*centAllocatedByGbf)*Action category: logical Auto backup capacity monitoring: capacity warning.

Parameter		Function
warning	*Warning*	Warning.
destination	*Destination*	Destination.
total capacity	*TotalCapacity*	Total capacity.
free capacity	*FreeCapacity*	Free capacity.
allocated by GBF	*AllocatedByGbf*	Allocated by GBF.
percent free	*PercentFree*	Percent free.
percent allocated	*PercentAllocated*	Percent allocated.
percent allocated by GBF	*PercentAllocatedByGbf*	Percent allocated by GBF.

Auto backup capacity file auto deleted

Action name:*AutoBackupCapacityMonitoringFileAutoDeleted(Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByGbf, PercentFree, PercentAllocated, PercentAllocatedByGbf, FileSize, FileName)*Action category: logical Auto backup capacity monitoring: file auto deleted.

Parameter		Function
warning	*Warning*	Warning.
destination	*Destination*	Destination.
total capacity	*TotalCapacity*	Total capacity.
free capacity	*FreeCapacity*	Free capacity.
allocated by GBF	*AllocatedByGbf*	Allocated by GBF.
percent free	*PercentFree*	Percent free.
percent allocated	*PercentAllocated*	Percent allocated.
percent allocated by GBF	*PercentAllocatedByGbf*	Percent allocated by GBF.
file size	*FileSize*	File size.
file name	*FileName*	File name.

Auto backup capacity out of disk space

Action name:*AutoBackupCapacityMonitoringOutOfDiskSpace(Warning, Destination, TotalCapacity, FreeCapacity, AllocatedByGbf, PercentFree, PercentAllocated, PercentAllocatedByGbf)*Action category: logical Auto backup capacity monitoring: out of disk space.

Parameter		Function
warning	*Warning*	Warning.
destination	*Destination*	Destination.
total capacity	*TotalCapacity*	Total capacity.
free capacity	*FreeCapacity*	Free capacity.
allocated by GBF	*AllocatedByGbf*	Allocated by GBF.
percent free	*PercentFree*	Percent free.
percent allocated	*PercentAllocated*	Percent allocated.
percent allocated by GBF	*PercentAllocatedByGbf*	Percent allocated by GBF.

Auto backup file done

Action name:*AutoBackupFileDone(Schedule, StartTime, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, Source, Destination, FileSizeLimit, BandWidthLimit, FileIndex, FileName, FileSize)*Action category: logical Auto backup progress notification: file done.

Parameter		Function
schedule	*Schedule*	Schedule.
start time	*StartTime*	Start time, empty during event backup.
effective start time	*EffectiveStartTime*	Effective schedule start time.
operation count	*OperationCount*	Operation count.
timer start	*TimerStart*	Timer start.
operation index	*OperationIndex*	Operation index.
operation start time	*OperationStartTime*	Operation start time.
source	*Source*	Source.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.
file index	*FileIndex*	File index.
file name	*FileName*	File name.
file size	*FileSize*	File size.

Auto backup file progress

Action name:*AutoBackupFileProgress(Schedule, StartTime, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, Source, Destination, FileSizeLimit, BandWidthLimit, FileIndex, FileName, FileSize)*Action category: logical Auto backup progress notification: file progress.

Parameter		Function
schedule	*Schedule*	Schedule.
start time	*StartTime*	Start time, empty during event backup.
effective start time	*EffectiveStartTime*	Effective schedule start time.
operation count	*OperationCount*	Operation count.
timer start	*TimerStart*	Timer start.
operation index	*OperationIndex*	Operation index.
operation start time	*OperationStartTime*	Operation start time.
source	*Source*	Source.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.
file index	*FileIndex*	File index.
file name	*FileName*	File name.
file size	*FileSize*	File size.

Auto backup file started

Action name:*AutoBackupFileStarted(Schedule, StartTime, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, Source, Destination,*

*FileSizeLimit, BandWidthLimit, FileIndex, FileName)*Action category: logical Auto backup progress notification: file started.

Parameter		Function
schedule	*Schedule*	Schedule.
start time	*StartTime*	Start time, empty during event backup.
effective start time	*EffectiveStartTime*	Effective schedule start time.
operation count	*OperationCount*	Operation count.
timer start	*TimerStart*	Timer start.
operation index	*OperationIndex*	Operation index.
operation start time	*OperationStartTime*	Operation start time.
source	*Source*	Source.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.
file index	*FileIndex*	File index.
file name	*FileName*	File name.

Auto backup operation done

Action name:*AutoBackupOperationDone(Schedule, StartTime, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, OperationStopTime, Source, Destination, FileSizeLimit, BandWidthLimit)*Action category: logical Auto backup progress notification: operation done.

Parameter		**Function**
schedule	*Schedule*	Schedule.
start time	*StartTime*	Start time, empty during event backup.
effective start time	*EffectiveStartTime*	Effective schedule start time.
operation count	*OperationCount*	Operation count.
timer start	*TimerStart*	Timer start.
operation index	*OperationIndex*	Operation index.
operation start time	*OperationStartTime*	Operation start time.
operation stop time	*OperationStopTime*	Operation stop time.
source	*Source*	Source.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.

Auto backup operation started

Action name:*AutoBackupOperationStarted(Schedule, StartTime, EffectiveStartTime, OperationCount, TimerStart, OperationIndex, OperationStartTime, Source, Destination, FileSizeLimit, BandWidthLimit)*Action category: logical Auto backup progress notification: operation started.

Parameter		Function
schedule	*Schedule*	Schedule.
start time	*StartTime*	Start time, empty during event backup.
effective start time	*EffectiveStartTime*	Effective schedule start time.
operation count	*OperationCount*	Operation count.

Parameter		Function
timer start	*TimerStart*	Timer start.
operation index	*OperationIndex*	Operation index.
operation start time	*OperationStartTime*	Operation start time.
source	*Source*	Source.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.

Auto backup schedule done

Action name:*AutoBackupScheduleDone(Schedule, StartTime, EffectiveStartTime, StopTime, OperationCount, TimerStart)*Action category: logical Auto backup progress notification: schedule done.

Parameter		**Function**
schedule	*Schedule*	Schedule.
start time	*StartTime*	Start time, empty during event backup.
effective start time	*EffectiveStartTime*	Effective schedule start time.
stop time	*StopTime*	Schedule stop time.
operation count	*OperationCount*	Operation count.
timer start	*TimerStart*	Timer start.

Auto backup schedule started

Action name:*AutoBackupScheduleStarted(Schedule, StartTime, EffectiveStartTime, OperationCount, TimerStart)*Action category: logical Auto backup progress notification: schedule started.

Parameter		Function
schedule	*Schedule*	Schedule.
start time	*StartTime*	Start time, empty during event backup.
effective start time	*EffectiveStartTime*	Effective schedule start time.
operation count	*OperationCount*	Operation count.
timer start	*TimerStart*	Timer start.

Backup event

Action name:*BackupEvent(EventID, TypeID, Destination, StartHintID, StopHintID, Subfolder)*Action category: logical Backup event.

Parameter		Function
instance ID	*EventID*	Instance ID of the event.
event type	*TypeID*	Type of the event.
destination	*Destination*	Destination.
start hint ID	*StartHintID*	Optional start hint ID.
stop hint ID	*StopHintID*	Optional stop hint ID.
sub folder	*Subfolder*	Sub folder to backup event.

Event backup done

Action name:*EventBackupDone(JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime, StopTime)*Action category: logical Event backup progress notification: backup done.

Parameter		Function
job ID	*JobID*	Backup job ID.
event type	*EventTypeID*	Type of the event.
instance ID	*EventID*	Instance ID of the event.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.
start time	*StartTime*	Backup start time.
stop time	*StopTime*	Backup stop time.

Event backup file done

Action name:*EventBackupFileDone(JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime, FileIndex, FileName, FileSize)*Action category: logical Event backup progress notification: file done.

Parameter		Function
job ID	*JobID*	Backup job ID.
event type	*EventTypeID*	Type of the event.
instance ID	*EventID*	Instance ID of the event.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.
start time	*StartTime*	Effective backup start time.
file index	*FileIndex*	File index.
file name	*FileName*	File name.
file size	*FileSize*	File size.

Event backup file progress

Action name:*EventBackupFileProgress(JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime, FileIndex, FileName, FileSize)*Action category: logical Event backup progress notification: file progress.

Parameter		Function
job ID	*JobID*	Backup job ID.
event type	*EventTypeID*	Type of the event.
instance ID	*EventID*	Instance ID of the event.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.
start time	*StartTime*	Effective backup start time.
file index	*FileIndex*	File index.
file name	*FileName*	File name.
file size	*FileSize*	File size.

Event backup file started

Action name:*EventBackupFileStarted(JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime, FileIndex, FileName)*Action category: logical Event backup progress notification: file started.

Parameter		Function
job ID	*JobID*	Backup job ID.
event type	*EventTypeID*	Type of the event.
instance ID	*EventID*	Instance ID of the event.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.
start time	*StartTime*	Effective backup start time.
file index	*FileIndex*	File index.
file name	*FileName*	File name.

Event backup started

Action name:*EventBackupStarted(JobID, EventTypeID, EventID, Destination, FileSizeLimit, BandWidthLimit, StartTime)*Action category: logical Event backup progress notification: backup started.

Parameter		Function
job ID	*JobID*	Backup job ID.
event type	*EventTypeID*	Type of the event.
instance ID	*EventID*	Instance ID of the event.
destination	*Destination*	Destination.
file size limit	*FileSizeLimit*	File size limit.
band width limit	*BandWidthLimit*	Band width limit.
start time	*StartTime*	Backup start time.

Start auto backup

Action name:*StartAutoBackup(Schedule)*Action category: logical Start auto backup.

Parameter		Function
schedule	*Schedule*	Schedule.

Camera control

Actions to set and control PTZ/normal cameras.

> Note: Which camera types are supported always depends on model and manufacturer!

Auto focus off

Action name: AutoFocusOff(PTZ Head)
Action category: command
This action disables the auto-focus function of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Auto focus on

Action name:AutoFocusOn(PTZ Head)
Action category: command
This action enables the auto-focus function of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera backlight compensation mode

Action name: CameraBacklightCompensationMode(PTZ Head, mode)
Category: command
This action changes the backlight compensation of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number
mode	*Mode*	off=backlight compensation is turned off on=backlight compensation is turned on

Camera clear preset text

Action name: CameraClearPresetText(PTZ Head, position)
Category: command
This action clears the text that was previously defined and assigned to a particular camera position by the action “CameraSetPresetText” and displayed when the camera moves to this position.

Parameter		Function
PTZ head	*Camera*	Global camera number
position	*Position*	Number of the camera position for which the previously defined text (by the action “CameraSetPresetText”) has to be cleared.

Camera day/night mode

Action name: CameraDayNightMode(PTZ Head, mode)
Category: command
This action changes the day/night mode of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number
mode	*Mode*	day=day mode is activated night=night mode is activated auto=the camera changes automatically between day and night mode

Camera light off

Action name: CameraLightOff(PTZ Head)
Category: command
This action turns the camera light off.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera light on

Action name: CameraLightOn(PTZ Head)
Category: command
This action turns the camera light on.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera manual iris off

Action name: CameraManualIrisOff(PTZ Head)
Category: command
This action disables the option to adjust the camera iris manually.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera manual iris on

Action name: CameraManualIrisOn(PTZ Head)
Category: command
This action enables the option to adjust the camera iris manually.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera off

Action name: CameraOff(PTZ Head)
Category: command
This action turns off the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera on

Action name: CameraOn(PTZ Head)
Category: command
This action turns on the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera pump off

Action name: CameraPumpOff(PTZ Head)
Category: command
This action disables the pump of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera pump on

Action name: CameraPumpOn(PTZ Head)
Category: command
This action enables the pump of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera RAW output

Action name: CameraRAWOutput(PTZ Head, output)
Category: command
This action sends a raw string (parameter output) to the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number
output	*Output*	raw string The following escape sequences are supported: \\a, b, f, n, r, t, v => \ a, b, f, n, r, t, v \\\=> \\ \\` => \` \" => \" \Xhh or \xhh => ASCII-character

Camera select char mode

For internal use only

Camera set preset text

Action name: CameraSetPresetText(PTZ Head, position)
Category: command
With this action, one defines the text that is associated with a particular camera position and displayed when the camera moves to this position.

Parameter		Function
PTZ head	*Camera*	Global camera number
position	*Position*	Number of the camera for which the text is defined.

Camera spec func U off

Action name: CameraSpecFuncUOff(PTZ Head)
Category: command
Special functions are mapped to this action.
(MBeg functions X, Y, U and V).

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera spec func U on

Action name: CameraSpecFuncUOn(PTZ Head)
Category: command

Special functions are mapped to this action.
(MBeg functions X, Y, U and V).

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera spec func V off

Action name: CameraSpecFuncVOff(PTZ Head)
Category: command
Special functions are mapped to this action.
(MBeg functions X, Y, U and V).

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera spec func V on

Action name: CameraSpecFuncVOn(PTZ Head)
Category: command
Special functions are mapped to this action.
(MBeg functions X, Y, U and V).

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera spec func X off

Action name: CameraSpecFuncXOff(PTZ Head)
Category: command
Special functions are mapped to this action.
(MBeg functions X, Y, U and V).

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera spec func X on

Action name: CameraSpecFuncXOn(PTZ Head)
Category: command
Special functions are mapped to this action.
(MBeg functions X, Y, U and V).

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera spec func Y off

Action name: CameraSpecFuncYOff(PTZ Head)
Category: command
Special functions are mapped to this action.
(MBeg functions X, Y, U and V).

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera spec func Y on

Action name: CameraSpecFuncYOn(PTZ Head)
Category: command
Special functions are mapped to this action.
(MBeg functions X, Y, U and V).

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera stop all

Action name: CameraStopAll(PTZ Head)
Category: command
This action stops all movements of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera text off

Action name: CameraTextOff(PTZ Head)
Category: command
This action turns off the text display of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera text on

Action name: CameraTextOn(PTZ Head)
Category: command
This action turns on the text display of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera tour start

Action name: CameraTourStart(PTZ Head, tour ID, tour name)
Category: command
This action starts a pre-defined tour.

Parameter		Function
PTZ head	*Camera*	Global camera number
tour id	*TourID*	Tour id.
tour name	*TourName*	Tour name.

Camera tour stop

Action name: CameraTourStop(PTZ Head)
Category: command
This action stops a running tour.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera version off

Action name: CameraVersionOff(PTZ Head)
Category: command
With this action the firmware version of the camera will be hidden.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera version on

Action name: CameraVersionOn(PTZ Head)
Category: command
With this action the firmware version of the camera will be shown as OSD.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera wash-wipe off

Action name: CameraWashOff(PTZ Head)
Category: command
This action disables the functions “wash” and “wipe”.

Parameter		Function
PTZ head	*Camera*	Global camera number

Camera wash-wipe on

CameraWashWhipeOn
Action name: CameraWashOn(PTZ Head)
Category: command
This action enables the functions “wash” and “wipe”.

Parameter		Function
PTZ head	*Camera*	Global camera number

Move to default position

Action name:DefaultPosCallUp(Camera)
Action category: command
The PTZ camera moves back to the home position (usually position 1).
Therefor the home position has to be set and saved in advance by the action "SaveDefaultPosition".

Parameter		Function
PTZ head	Camera	Global camera number

Clear default position

Action name: ClearDefaultPosition(PTZ Head)
Category: command
This action deletes the currently defined default position.

Parameter		Function
PTZ head	*Camera*	Global camera number

Clear preset position

Action name: CameraPresetPosition(PTZ Head, position)
Category: command
This action deletes a position previously saved by the action “SavePresetPosition”.

Parameter		Function
PTZ head	*Camera*	Global camera number
position		Number of camera position to be deleted.

Save default position

Action name: SaveDefaultPosition(PTZ Head)
Category: command
This action saves the current position of the camera as default position.

Parameter		Function
PTZ head	*Camera*	Global camera number

Fast speed off

Action name: FastSpeedOff(PTZ Head)
Category: command
This action switches from high-speed of the camera to normal speed of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Fast speed on

Action name: FastSpeedOn(PTZ Head)
Category: command
This action switches from normal speed of the camera to high-speed of the camera.

Parameter		Function
PTZ head	*Camera*	Global camera number

Focus far

Action name:FocusFar(Camera, Speed)
Action category: command
The camera focus adjusts on far.

Parameter		Function
PTZ head	*Camera*	Global camera number
speed	*Speed*	Depending on the protocol of camera man-ufacturer velocities between 1 and 255 are being adjusted to the velocity range of the camera.

Focus near

Action name:FocusNear(Camera, Speed)
Action category: command
The camera focus adjusts on near.

Parameter		Function
PTZ head	*Camera*	Global camera number
speed	*Speed*	Depending on the protocol of camera manufacturer velocities between 1 and 255 are being adjusted to the velocity range of the camera.

Focus stop

Action name:FocusStop(Camera)
Action category: command
The camera stops the focusing process.

Parameter		Function
PTZ head	*Camera*	Global camera number

Iris close

Action name:IrisClose(Camera)
Action category: command
The camera closes the aperture.

Parameter		Function
PTZ head	*Camera*	The camera closes the aperture

Iris open

Action name:IrisOpen(Camera)
Action category: command
The camera opens the aperture.

Parameter		Function
PTZ head	*Camera*	The camera opens the aperture

Iris stop

Action name:IrisStop(Camera)
Action category: command
The camera stops closing/opening aperture.

Parameter		Function
PTZ head	*Camera*	The camera stops closing/opening aperture

Move to absolute position

For internal use only

Move to by speed

For internal use only

Move to relative position

For internal use only

Pan auto

Action name:PanAuto(Camera, Modus)
Action category: command
Cameras without automatic end stop turn on and on until this function is stopped through the action "PanStop". Cameras with automatic end stop do stop automatically after a 360 turn. It depends on the camera type if this function is even available and in case how it is going to be accomplished.

Parameter		Function
PTZ head	*Camera*	Global camera number
modus	*Modus*	Depends on camera type (model and manufacturer)

Pan left

Action name:PanLeft(Camera, Speed)
Action category: command
The camera pans to the left.

Parameter		Function
PTZ head	*Camera*	Global camera number
modus	*Speed*	Depending on the protocol of camera manufacturer velocities between 1 and 255 are being adjusted to the velocity range of the camera.

Pan right

Action name:PanRight(Camera, Speed)
Action category: command
The camera pans to the right.

Parameter		Function
PTZ head	*Camera*	Global camera number
modus	*Speed*	Pan speed.

Pan stop

Action name:PanStop(Camera)
Action category: command
The camera stops pan movement.

Parameter		Function
PTZ head	*Camera*	Global camera number

Move to preset position

Action name:PrePosCallUp(Camera, Position)
Action category: command
The camera moves to a preset position determined in advance through the action "SavePresetPosition".

Parameter		Function
PTZ head	*Camera*	Global camera number
position	*Position*	Number of selected preset position. The amount of positions to save depends on

Parameter		Function
		the camera type (model and manufacturer).

Clear preset position

Action name:PrePosClear(Camera, Position)
Action category: command
Clear camera preset position.

Parameter		Function
PTZ head	*Camera*	Global camera number
position	*Position*	Preset position.

Save preset position

Action name:PrePosSave(Camera, Position)
Action category: command
Saves current position of the PTZ camera as a preset position.

Parameter		Function
PTZ head	*Camera*	Global camera number
position	*Position*	Number of preset position on which the current position of the camera should be saved. The amount of positions to save depends on the camera type (model and manufacturer).

Set camera text

Action name: SaveCameraText(PTZ Head, text)
Category: command
This action saves the camera description in accordance with the parameter “text”.

Parameter		Function
PTZ head	*Camera*	Global camera number
text	*Text*	Text to be displayed on the camera as OSD.

Tilt down

Action name:TiltDown(Camera, Speed)
Action category: command
The camera tilts down.

Parameter		Function
PTZ head	*Camera*	Global camera number
speed	*Speed*	Depending on the protocol of camera manufacturer velocities between 1 and 255 are being adjusted to the velocity range of the camera.

Tilt stop

Action name:TiltStop(Camera)
Action category: command
The camera stops the tilt movement.

Parameter		Function
PTZ head	*Camera*	Global camera number

Tilt up

Action name:TiltUp(Camera, Speed)
Action category: command
The camera tilts up.

Parameter		Function
PTZ head	*Camera*	Global camera number
speed	*Speed*	Depending on the protocol of camera manufacturer velocities between 1 and 255 are being adjusted to the velocity range of the camera.

Zoom in

Action name:ZoomIn(Camera, Speed)
Action category: command
The Camera zooms in (tele range).

Parameter		Function
PTZ head	*Camera*	Global camera number
speed	*Speed*	Depending on the protocol of camera manufacturer velocities between 1 and 255 are being adjusted to the velocity range of the camera.

Zoom out

Action name:ZoomOut(Camera, Speed)
Action category: command
The camera zooms out (wide-angle range).

Parameter		Function
PTZ head	*Camera*	Global camera number
speed	*Speed*	Depending on the protocol of camera manufacturer velocities between 1 and 255 are being adjusted to the velocity range of the camera.

Zoom stop

Action name:ZoomStop(Camera)
Action category: command
The camera stops zooming process.

Parameter		Function
PTZ head	*Camera*	Global camera number

Cash management actions

Cash Management Actions offer the exchange of accompanying meta data between Cash Management Systems and GeViScope/Re_porter. With these actions money handling processes can be documented consistently via video. The use of these actions for starting and restarting of event recordings leads to the display of the accompanying video data in live

streams of GscView and the storage of those in the video database. The video sequences recorded by Cash Management Actions can later be recovered easily in GscView by using the accompanying meta data and a special data filter dialog.

Safebag close

Action name:SafebagClose(WorkingPlace, StartTime, StopTime, SafebagNo, SafebagInfo, StepID, Debit, Total, Difference, HasDifference, Notes, Coins, Cheques)
Action category: logical
Safebag close.
The integrated Cash Management System sends the action as soon as the user has finished the counting of one safe bag and has confirmed that to the Cash Management System .
Via the parameter "working place" the affected working place will be identified. The further parameter will be provided with accompanying video data by Cash Management System.
The parameter "StepID" can be provided with a code figure by the Cash Management System for the currently running process step.

Parameter		Function
working place	*WorkingPlace*	Working place no.
start time	*StartTime*	Time stamp, when the handling of the safe bag began.
stop time	*StopTime*	Time stamp, when the handling of the safe bag stopped.
safebag no.	*SafebagNo*	Alphanumerical identification of safe bag; search criteria in GscView
safebag info	*SafebagInfo*	Additional alphanumerical identification of safe bag
step id	*StepID*	Code figure for the currently running process step (given by Cash Management System individually)
debit	*Debit*	Debit amount of safebag
total	*Total*	Effective total amount of safe bag according to counting (will be accumulated by Cash Management Systems on counting)
difference	*Difference*	Difference between total amount and result respectively progress of counting
has difference	*HasDifference*	Yes = current total amount has a difference to debit amount No = current total amount is identical with debit amount
notes	*Notes*	Denomination of counted banknotes. The Display in GscView is in table form. The table has max. 2 columns. The individual lines can be separated via insertion of control '\r' (0x0D). The separation of both columns within one line can be carried out via insertion of control '\t' (0x09).
coins	*Coins*	Denomination of counted coins. The Display in GscView is in table form. The table has max. 2 columns. The individual lines can be separated via insertion of control '\r' (0x0D). The separation of both columns within one line can be carried out via insertion of control '\t' (0x09).
cheques	*Cheques*	Denomination of counted cheques. The Display in GscView is in table form. The table has max. 2 columns. The individual lines can be separated via insertion of control '\r' (0x0D). The separation of both columns within one line can be carried out via insertion of control '\t' (0x09).

Safebag data

Action name:SafebagData(WorkingPlace, StartTime, SafebagNo, SafebagInfo, StepID, Debit, Total, Difference, HasDifference, Notes, Coins, Cheques)
Action category: command
Safebag data.
The integrated Cash Management System sends the action as soon as the user has finished counting one variety of notes or coins and has confirmed that to the system.
Via the parameter "working place" the affected working place will be identified. The further parameter will be provided with accompanying meta data by the Cash Management System.
The parameter "StepID" can be provided with a code figure by the Cash Management System for the currently running process step.

Parameter		Function
working place	*WorkingPlace*	Working place no.
start time	*StartTime*	Time stamp, when the handling of the safe bag began.
safebag no.	*SafebagNo*	Alphanumerical identification of safe bag; search criteria in GscView
safebag info	*SafebagInfo*	Additional alphanumerical identification of safe bag
step id	*StepID*	Code figure for the currently running process step (given by cash management system individually
debit	*Debit*	Debit amount of safe bag
total	*Total*	Effective total amount of safe bag according to counting (will be accumulated by the Cash management System during counting)
difference	*Difference*	Difference between total amount and result respectively progress of counting
has difference	*HasDifference*	Yes = current total amount has a difference to debit amount No = current total amount is identical with debit amount
notes	*Notes*	Denomination of counted banknotes. The Display in GscView is in table form. The table has max. 2 columns. The individual lines can be separated via insertion of control '\r' (0x0D). The separation of both columns within one line can be carried out via insertion of control '\t' (0x09).
coins	*Coins*	Denomination of counted coins. The Display in GscView is in table form. The table has max. 2 columns. The individual lines can be separated via insertion of control '\r' (0x0D). The separation of both columns within one line can be carried out via insertion of control '\t' (0x09).
cheques	*Cheques*	Denomination of counted cheques. The Display in GscView is in table form. The table has max. 2 columns. The individual lines can be separated via insertion of control '\r' (0x0D). The separation of both columns within one line can be carried out via insertion of control '\t' (0x09).

Safebag open

Action name:SafebagOpen(WorkingPlace, StartTime, SafebagNo, SafebagInfo, StepID)
Action category: notification
Safebag open.
The integrated Cash Management System sends the action as soon as the user has opened the safe bag and confirmed that with an entry in the Cash Management System.

The affected Working Place will be identified via the parameter "Working place". Further parameters will be filled with accompanying meta data on the part of the Cash Management System.
The Parameter "StepID" can be addressed by the Cash Management System with a code figure for the currently running process step.

Parameter		Function
working place	*WorkingPlace*	Working place no.
start time	*StartTime*	Time stamp, when the handling of the safe bag began.
safebag no.	*SafebagNo*	Alphanumerical identification of safe bag; search criteria in GscView
safebag info	*SafebagInfo*	Additional alphanumerical identification of safe bag
step id	*StepID*	Code figure for the currently running process step (given by cash management system individually

Safebag passing of risk data

Action name:SafebagPassingOfRiskData(WorkingPlace, StartTime, SafebagNo, SafebagInfo, StepID, UserID1, UserID2, TourNumber, TargetWorkingPlace, PassingOfRiskType)
Action category: command
The integrated Cash Management System sends the action continuously for each safe bag while the amount of safe bags between two employees will be transferred and this will be confirmed to the Cash Management System. This part of the money handling process is a "passing of risk". Via the parameter "working place" the affected transfer place and respectively the working place will be identified. The further parameters will be filled with accompanying video data by the Cash Management System. The parameter "StepID" can be provided with a code figure by the Cash Management System for the currently running process step.

Parameter		Function
working place	*WorkingPlace*	Working place no.
start time	*StartTime*	Time stamp, when the handling of the safe bag began.
safebag no.	*SafebagNo*	Alphanumerical identification of safe bag; search criteria in GscView
safebag info	*SafebagInfo*	Additional alphanumerical identification of safe bag
step id	*StepID*	Code figure for the currently running process step (given by cash management system individually
user 1	*UserID1*	Number of employee, transferring the safe bag to another employee.
user 2	*UserID2*	Number of employee, who receives the safe bag from another employee.
tour no	*TourNumber*	Tour-Number (optional)
target working place	*TargetWorkingPlace*	Alphanumerical identification of a place respectively a working place where safe bags will be transferred to (optional)
passing of risk type	*PassingOfRiskType*	Detailed information to "passing of risk" (optional)

Safebag passing of risk start

Action name:SafebagPassingOfRiskStart(WorkingPlace, StartTime, SafebagNo, SafebagInfo, StepID, UserID1, UserID2, TourNumber, TargetWorkingPlace, PassingOfRiskType)
Action category: command
The integrated Cash Management System sends the action as soon as a number of safe bags will be transferred between two employees and this is confirmed to the Cash Management System. This part of the money handling process is a "passing of risk". Via the parameter "working place" the affected transfer place and respectively the working place will be identified. The further parameters will be filled with accompanying meta data by the Cash Management System. The parameter "StepID" can be provided with a code figure by the Cash Management System for the currently running process step.

Parameter		Function
working place	*WorkingPlace*	Working place no.
start time	*StartTime*	Time stamp, when the handling of the safe bag began.
safebag no.	*SafebagNo*	Alphanumerical identification of safe bag; search criteria in GscView
safebag info	*SafebagInfo*	Additional alphanumerical identification of safe bag
step id	*StepID*	Code figure for the currently running process step (given by cash management system individually
user 1	*UserID1*	Number of employee, transferring the safe bag to another employee.
user 2	*UserID2*	Number of employee, who receives the safe bag from another employee.
tour no	*TourNumber*	Tour-Number (optional)
target working place	*TargetWorkingPlace*	Alphanumerical identification of a place respectively a working place where safe bags will be transferred to (optional)
passing of risk type	*PassingOfRiskType*	Detailed information to "passing of risk" (optional)

Safebag passing of risk stop

Action name:SafebagPassingOfRiskStop(WorkingPlace, StartTime, StopTime, SafebagNo, SafebagInfo, StepID, UserID1, UserID2, TourNumber, TargetWorkingPlace, PassingOfRiskType)
Action category: command
The integrated Cash Management System sends the action closing after the last safe bag, while the number of safe bags will be transferred between two employees and this is confirmed to the Cash Management System. This part of the money handling process is a "passing of risk". Via the parameter "Working place" the affected transfer place respective working place will be identified. The further parameters will be filled with accompanying video data by the Cash Management System. The parameter "StepID" can be supplied by Cash Management System with a code figure for a currently running process step

Parameter		Function
working place	*WorkingPlace*	Working place no.
start time	*StartTime*	Time stamp, when the handling of the safe bag began.
safebag no.	*SafebagNo*	Alphanumerical identification of safe bag; search criteria in

Parameter		Function
		GscView
safebag info	*SafebagInfo*	Additional alphanumerical identification of safe bag
step id	*StepID*	Code figure for the currently running process step (given by cash management system individually
user 1	*UserID1*	Number of employee, transferring the safe bag to another employee.
user 2	*UserID2*	Number of employee, who receives the safe bag from another employee.
tour no	*TourNumber*	Tour-Number (optional)
target working place	*TargetWorkingPlace*	Alphanumerical identification of a place respectively a working place where safe bags will be transferred to (optional)
passing of risk type	*PassingOfRiskType*	Detailed information to "passing of risk" (optional)

Device information

All actions for low-level notification of the device or media channels changes.

Device found

Action name:DeviceFound(Type, Name, Serial)
Action category: logical
This action will be fired when the USB or NET device is connected to the system. It is also fired at start-up for all detected devices.

Parameter		Function
device type	Type	Type of the device.
device name	Name	Device name if assigned in setup, empty otherwise.
serial ID	Serial	Serial ID of the device.

New firmware received

Action name:DeviceNewFirmware(Type, Name, Serial, Firmware)
Action category: logical
This action will be fired when the USB or NET device has got the new firmware.

Parameter		Function
device type	Type	Type of the device.
device name	Name	Device name if assigned in setup, empty otherwise.
serial ID	Serial	Serial ID of the device.
firmware serial	Firmware	Serial ID of the firmware.

Device plugin error

Action name:DevicePluginError(Channel, Type, SubType, Name, Serial, ErrorClass, ErrorCode, Description)
Action category: logical
This action notifies device plugin error.

Parameter		Function
channel	Channel	Channel.
device type	Type	Type of the device.
device sub type	SubType	Sub type of the device.

Parameter		Function
device name	Name	Device name.
serial ID	Serial	Serial ID of the device.
error class	ErrorClass	Error class of the error occured.
error code	ErrorCode	Plugin type specific error code.
description	Description	Error description.

Device plugin state

Action name:DevicePluginState(Channel, Type, SubType, Name, Serial, State, InternalState, Description)
Action category: logical
This action notifies device plugin state.

Parameter		Function
channel	Channel	Channel.
device type	Type	Type of the device.
device sub type	SubType	Sub type of the device.
device name	Name	Device name.
serial ID	Serial	Serial ID of the device.
plugin state	State	New plugin device state.
internal state	InternalState	Plugin device specific state.
description	Description	State description.

Device reattached

Action name:DeviceReattached(Type, Name, Serial)
Action category: logical
This action will be fired when the USB or NET device is reattached to the system.

Parameter		Function
device type	Type	Type of the device.
device name	Name	Device name if assigned in setup, empty otherwise.
serial ID	Serial	Serial ID of the device.

Device removed

Action name:DeviceRemoved(Type, Name, Serial)
Action category: logical
This action will be fired when the USB or NET device is disconnected from the system. It is also fired at the start-up for all parameterized but not present devices.

Parameter		Function
device type	Type	Type of the device.
device name	Name	Device name if assigned in setup, empty otherwise.
serial ID	Serial	Serial ID of the device.

Digital contacts

All actions for handling digital inputs and outputs.

Digital input

Action name:DigitalInput(Contact, State)

Action category: logical
This action will be fired when the state of the digital input has changed.

Parameter		Function
contact	Contact	Contact.
state	State	New state.

IOI43 reset mainboard

Action name:IOI43ResetMainboard()
Action category: logical

Reset mainboard using IOI43a/ab USB Alarm-I/O.

IOI43 temperature notification

Action name:IOI43Temperature(ID, Temperature)
Action category: logical

Temperature notification from IOI43a/ab USB Alarm-I/O.

Parameter		Function
ID	ID	ID of the IOI43 module (like IOI43-00).
temperature	Temperature	Temperature.

IOI43 watchdog activate

Action name:IOI43WDActivate()
Action category: logical

Activate watchdog on IOI43a/ab USB Alarm-I/O.

IOI43 watchdog deactivate

Action name:IOI43WDDeactivate()
Action category: logical

Deactivate watchdog on IOI43a/ab USB Alarm-I/O.

IOI43 watchdog trigger

Action name:IOI43WDTrigger()
Action category: logical

Trigger watchdog on IOI43a/ab USB Alarm-I/O.

Key pressed

Action name:KeyPressed(Key)
Action category: logical
This action is notified if one of the GEVISCOPE system keys is pressed.

Parameter		Function
Key	Key	System key.

Key released

Action name:KeyReleased(Key)
Action category: logical
This action is notified if one of the GEVISCOPE system keys is released.

Parameter		Function
Key	Key	System key.

Set digital output

Action name:SetDigitalOutput(Contact, State)
Action category: logical
This action is used to modify the state of the digital output and to notify this change.

Parameter		Function
contact	Contact	Contact.
state	State	New state.

Set system LED

Action name:SetLED(LED, State)
Action category: logical
This action is used to turn the system LEDs on or off.

Parameter		Function
LED	LED	System LED.
state	State	New state.

Set system LED to blink

Action name:SetLEDBlink(LED, LedTimeOnMs, LedTimeOffMs)
Action category: logical
This action is used to blink the system LEDs.

Parameter		Function
LED	LED	System LED.
Led time ON	LedTimeOnMs	Time in milliseconds the LED will be switched on.
Led time OFF	LedTimeOffMs	Time in milliseconds the LED will be switched off.

Lenel

Lenel OnGuard actions.

Lenel access event

Action name:*LenelAccessEvent(ID, Panel, Device, SecondaryDevice, CardNumber, AccessResult, Type, SubType, Description, SerialNumber, TimeStamp, AreaEnteredID, AreaExitedID, AssetID, CardholderEntered, Duress, ElevatorFloor, FacilityCode, IsReadableCard, IssueCode, CommServerHostName, EventText)*
Action category: logical
Lenel OnGuard access event.

Parameter		Function
ID	*ID*	The ID that uniquely identifies the type of this event.
panel	*Panel*	The name of the panel where this event originated.
device	*Device*	The name of the device where this event originated.
secondary device	*SecondaryDevice*	The ID of the secondary device where this event originated.
card number	*CardNumber*	The badge ID for the card that was read, if available.
access result	*AccessResult*	The level of access that was granted that resulted from reading the card.

Parameter		Function
type	*Type*	Event type i.e., duress, system, etc.
subtype	*SubType*	Event sub-type i.e., granted, door forced open, etc.
description	*Description*	A human readable, brief description of this event.
serial number	*SerialNumber*	A number that uniquely identifies the instance of the event for a particular panel.
time stamp	*TimeStamp*	Time stamp.
area entered	*AreaEnteredID*	The ID of the area that was entered, if any.
area exited	*AreaExitedID*	The ID of the area that was exited, if any.
asset ID	*AssetID*	The ID of the asset related to this event, if any.
cardholder entered	*CardholderEntered*	Whether entry was made by the cardholder.
duress	*Duress*	Indicates whether this card access indicates an under duress/emergency state.
elevator floor	*ElevatorFloor*	The elevator floor on which the access event was generated, if any.
facility code	*FacilityCode*	The facility code for the card that was read, if available.
readable card	*IsReadableCard*	Whether the card could be read.
issue code	*IssueCode*	The issue code for the card that was read, if available.
server host	*CommServerHostName*	Host name of the Communication server through which the event arrived.
event text	*EventText*	Text associated with event

Lenel fire event

Action name:*LenelFireEvent(ID, Panel, Device, SecondaryDevice, TroubleCode, Type, SubType, Description, SerialNumber, TimeStamp, CommServerHostName, EventText)*
Action category: logical
Lenel OnGuard fire event.

Parameter		Function
ID	*ID*	The ID that uniquely identifies the type of this event.
panel	*Panel*	The name of the panel where this event originated.
device	*Device*	The name of the device where this event originated.
secondary device	*SecondaryDevice*	The ID of the secondary device where this event originated.
trouble code	*TroubleCode*	A trouble code associated with the fire event.
type	*Type*	Event type i.e., duress, system, etc.
subtype	*SubType*	Event sub-type i.e., granted, door forced open, etc.
description	*Description*	A human readable, brief description of this event.
serial number	*SerialNumber*	A number that uniquely identifies the instance of the event for a particular panel.
time stamp	*TimeStamp*	Time stamp.
server host	*CommServerHostName*	Host name of the Communication server through which the event arrived.
event text	*EventText*	Text associated with event

Lenel intercom event

Action name:*LenelIntercomEvent(ID, Panel, Device, SecondaryDevice, IntercomData, LineNumber, Type, SubType, Description, SerialNumber, TimeStamp, CommServerHostName, EventText)*
Action category: logical
Lenel OnGuard intercom event.

Parameter		Function
ID	*ID*	The ID that uniquely identifies the type of this event.
panel	*Panel*	The name of the panel where this event originated.
device	*Device*	The name of the device where this event originated.
secondary device	*SecondaryDevice*	The ID of the secondary device where this event originated.
intercom data	*IntercomData*	Additional data for the intercom event that occurred.
line number	*LineNumber*	The line number involved in the intercom event.
type	*Type*	Event type i.e., duress, system, etc.
subtype	*SubType*	Event sub-type i.e., granted, door forced open, etc.
description	*Description*	A human readable, brief description of this event.
serial number	*SerialNumber*	A number that uniquely identifies the instance of the event for a particular panel.
time stamp	*TimeStamp*	Time stamp.
server host	*CommServerHostName*	Host name of the Communication server through which the event arrived.
event text	*EventText*	Text associated with event

Lenel raw data

Action name:*LenelRawData(TimeStamp, LenelData)*
Action category: logical
Lenel OnGuard raw data.

Parameter		Function
time stamp	*TimeStamp*	Time stamp.
data	*LenelData*	Lenel OnGuard data.

Lenel refresh names

Action name:*LenelRefreshNames()*
Action category: logical
Lenel OnGuard refresh names.

Lenel security event

Action name:*LenelSecurityEvent(ID, Panel, Device, SecondaryDevice, Type, SubType, Description, SerialNumber, TimeStamp, CommServerHostName, EventText)*
Action category: logical
Lenel OnGuard security event.

Parameter		Function
ID	*ID*	The ID that uniquely identifies the type of this event.
panel	*Panel*	The name of the panel where this event originated.
device	*Device*	The name of the device where this event originated.
secondary device	*SecondaryDevice*	The ID of the secondary device where this event originated.
type	*Type*	Event type i.e., duress, system, etc.
subtype	*SubType*	Event sub-type i.e., granted, door forced open, etc.
description	*Description*	A human readable, brief description of this event.
serial number	*SerialNumber*	A number that uniquely identifies the instance of the event for a particular panel.
time stamp	*TimeStamp*	Time stamp.
server host	*CommServerHostName*	Host name of the Communication server through which the event arrived.

Parameter		Function
event text	*EventText*	Text associated with event

Lenel video event

Action name:*LenelVideoEvent(ID, Panel, Device, SecondaryDevice, Channel, Type, SubType, Description, SerialNumber, TimeStamp, StartTime, EndTime, CommServerHostName, EventText)*
Action category: logical
Lenel OnGuard video event.

Parameter		Function
ID	*ID*	The ID that uniquely identifies the type of this event.
panel	*Panel*	The name of the panel where this event originated.
device	*Device*	The name of the device where this event originated.
secondary device	*SecondaryDevice*	The ID of the secondary device where this event originated.
channel	*Channel*	The physical channel the camera is connected to that is creating this event.
type	*Type*	Event type i.e., duress, system, etc.
subtype	*SubType*	Event sub-type i.e., granted, door forced open, etc.
description	*Description*	A human readable, brief description of this event.
serial number	*SerialNumber*	A number that uniquely identifies the instance of the event for a particular panel.
time stamp	*TimeStamp*	Time stamp.
start stamp	*StartTime*	The time the video event started
end time	*EndTime*	The time the video event ended.
server host	*CommServerHostName*	Host name of the Communication server through which the event arrived.
event text	*EventText*	Text associated with event

Logistic

Logistic actions are used in the logistic environment where meta data, e.g. barcodes, is used to start recording events. Later, a research on the barcodes is done to show the scanning operation in the recorded images. To speed up the search, a CRC32 checksum is used as a hash and serves as a foreign key of the event startd. The foreign key is indexed in the event table an can therefore be found much faster than a lookup on the string itself. Additional parameters are used to notify positioning information since the assignment of scanning and recording camera is often done according to the position of the scanner.

Log barcode data

Action name:*LogBarcodeData(Barcode, Hash, Scanner, AreaID, AreaName, Channel, TimeStamp)*
Action category: notification
Logistic barcode data .

Parameter		Function
barcode	*Barcode*	Barcode.
hash value	*Hash*	Hash value of barcode (Optional)
scanner name	*Scanner*	Scanner name or IP Address (Optional)
area number	*AreaID*	Global number of area for event mapping (Optional)

Parameter		Function
area name	*AreaName*	Area name (Optional)
channel	*Channel*	Global number of a media channel for mapping (Optional)
time stamp	*TimeStamp*	Time stamp (Optional)

Log barcode data LPS

Action name:*LogBarcodeDataLPS(Barcode, Hash, Scanner, AreaID, AreaName, Channel, TimeStamp, X, Y, Z, LpsTagID, LpsStatus, LpsCellID, LpsAreaID, UserParam)*
Action category: notification
Logistic barcode data including positioning and area information.

Parameter		Function
barcode	*Barcode*	Barcode.
hash value	*Hash*	Hash value of the barcode (Optional)
scanner name	*Scanner*	Scanner name or IP Address (Optional)
area number	*AreaID*	Global number of area for event mapping (Optional)
area name	*AreaName*	Area name. (Optional)
channel	*Channel*	Global number of a media channel for mapping (Optional)
time stamp	*TimeStamp*	Time stamp (Optional)
X coordinate	*X*	X coordinate of the position query (Optional)
Y coordinate	*Y*	Y coordinate of the position query (Optional)
Z coordinate	*Z*	Z coordinate of the position query (Optional)
LPS tag ID	*LpsTagID*	Tag ID of the positioning system (Optional)
LPS status	*LpsStatus*	LPS status of the position query(Optional)
LPS cell ID	*LpsCellID*	Cell ID of the positioning system (Optional)
LPS area ID	*LpsAreaID*	Area ID of the positioning system (Optional)
User param	*UserParam*	User param for internal use (Optional)

Log NPR recognition

Action name:*LogNPRRecognition(PlateNo, Hash, Country, Channel, TimeStamp, Restriction, Category)*
Action category: logical
Log NPR recognition.

Parameter		Function
plate no.	*PlateNo*	Recognized plate no.
hash value	*Hash*	Hash value of the recognized plate no. (Optional)
country	*Country*	Country (Optional)
channel	*Channel*	Channel (Optional)
time stamp	*TimeStamp*	Time stamp (Optional)
restriction	*Restriction*	Restriction of recognized number (Optional)
category	*Category*	Category of recognized number (Optional)

LPS Actions

LPS (Local Positioning System) actions are used to query and receive position data. The positioning system is integrated by the GscLPS plugin and is used to locate tagged objects, e.g. mobile scanners in the logistic environment. The tags have IDs that can be used to query the position which is then notified as cartesian or geografic coordinates. Some tags are able to initiate a position request by an external trigger or by a scan event on a mobile scanner.

LPS position data

Action name:*LPSPositionData(TagID, ScannerID, X, Y, Z, Latitude, Longitude, AreaID, CellID, Status, TimeStamp, Data, AreaName)*
Action category: logical
LPS position data.

Parameter		Function
tag ID	*TagID*	Tag ID.
scanner ID	*ScannerID*	Scanner ID or IP Address.
X coordinate	*X*	X coordinate of cartesian coordinates.
Y coordinate	*Y*	Y coordinate of cartesian coordinates.
Z coordinate	*Z*	Z coordinate of cartesian coordinates.
Latitude	*Latitude*	Latitude of geographic coordinates.
Longitude	*Longitude*	Longitude of geographic coordinates.
area ID	*AreaID*	Area ID.
cell ID	*CellID*	Cell ID.
status	*Status*	Status.
time stamp	*TimeStamp*	Time stamp.
data	*Data*	Data received by the positioning system, eg. barcode.
area name	*AreaName*	Area Name.

LPS query position

Action name:*LPSQueryPosition(TagID, ScannerID, Data)*
Action category: command
Send position query for a Tag to LPS server.

Parameter		Function
tag ID	*TagID*	Tag ID.
scanner ID	*ScannerID*	Scanner ID or IP Address.
data	*Data*	Data.

POS

Points of sales (POS) Actions enable the exchange of accompanying meta data between POS Management Systems and GeViScope/re_porter. With these actions payment processes can be documented consistently by video. The use of these actions for start and restart of event recordings leads to the output of accompanying meta data in live video in GSCView as well as in the storage of those in the video data base. The video sequences recorded via POS Actions can easily be retrieved in GscView using the accompanying meta data und special data filter dialogs (optional) Besides the actions POSStatus and POSData for the general integration into POS Management Systems there are also POS actions which belong to special GeViScope drivers. The actions FillingPumpStatus, TerminalArticleData and TerminalPaymentData are used by the driver "HUTH". The driver "HUTH" is a GeViScope Media Plugin, which was developed by GEUTEBRÜCK, to integrate filling station management systems of the manufacturer HUTH Elektronik Systeme GmbH into GeViScope/re_porter. The driver is compatible to HUTH Video Interface T400/T450/Maxi/mini V1.2. The actions InterfaceRawData and InterfaceRawAnswer are also used by the driver "HUTH". But they only serve for debugging and fault analysis purpose. They can also be used in general for any link that the concerned action supports - respectively uses these actions. The action BarcodeData serves as a general integration of barcode scanners.

Barcode data

Action name:BarcodeData(ReaderName, TimeStamp, Barcode)
Action category: notification
The POS Management System (or any other system like barcode scanner or similar) sends the action as soon as a barcode was read. Via the parameter "ReaderName" the affected barcode scanner will be identified. The further parameter will be filled with video meta data by the POS Management System.

Parameter		Function
scanner	*ReaderName*	Alphanumerical identification of the barcode scanner
time stamp	*TimeStamp*	Time stamp.
code	*Barcode*	Alphanumerical field for recording the scanned barcode.

Filling pump status

Action name:FillingPumpStatus(TerminalName, TimeStamp, PumpNo, Status, Amount, Price, Details) Action category: notification
The "HUTH" driver sends the action for each status change of one filling pump. Via the parameter "TerminalName" the concerned device will be identified. The "HUTH" driver is principally able to build up several connections to different Huth devices. The driver sends the alphanumerical value defined in his setup as "Interface name". The further parameter will be filled with video meta data by the driver.

Parameter		Function
Terminal	*TerminalName*	Identifies the affected device. The "HUTH" driver is principally able to build up several connections to different Huth devices. The driver sends the alphanumerical value defined in his setup as "Interface name".
time stamp	*TimeStamp*	Time Stamp, when the status change was detected by the Huth-System
pump no	*PumpNo*	Number of the filling pump
status	*Status*	New status of the filling pump Filling started = Huth-device status "taken off before filling" Filling stopped = Huth-device status "put back on end of filling" Pump released = Huth-device status "disconnect after filling" Amount message = sum - respectively amount notice of the filling pump
amount	*Amount*	Amount of the booking (optional)
price	*Price*	Sum of the booking (optional)
details	*Details*	Free text (optional)

Interface raw answer

Action name:InterfaceRawAnswer(InterfaceName, TimeStamp, Data)
Action category: notification
This action is used by the "HUTH" driver. ". It serves only as a debug service and can also be used in general for any integration that supports or uses this action. The "HUTH" driver sends the action for each telegram it has sent to the end device. The affected end device will be identified by the parameter "TerminalName". The "HUTH" driver can always build up numerous connections to different Huth devices. The driver then sends the alphanumerical value defined in its setup as "Interface name" The further parameter will be filled with video meta data by the driver.

Parameter		Function
interface	*InterfaceName*	Identifies the affected end device. The "HUTH" driver is principally able to build up several connections to different Huth devices. The driver sends the alphanumerical value defined in his setup as "Interface name".
time stamp	*TimeStamp*	TimeStamp when the telegram was received from the Huth system.
answer	*Data*	The sent telegram in raw format.

Interface raw data

Action name:InterfaceRawData(InterfaceName, TimeStamp, Data)
Action category: notification
This action is used by the driver "HUTH". It serves only as a debug service and can also be used in general for any integration that supports or uses this action. The "HUTH" driver sends the action for each telegram it has received from the end device. The affected end device will be identified by the parameter "TerminalName". The "HUTH" driver can always build up numerous connections to different Huth devices. The driver then sends the alphanumerical value defined in its setup as "Interface name" The further parameter will be filled with video meta data by the driver.

Parameter		Function
interface	*InterfaceName*	Identifies the affected end device. The "HUTH" driver is principally able to build up several connections to different Huth devices. The driver sends the alphanumerical value defined in his setup as "Interface name".
time stamp	*TimeStamp*	TimeStamp when the telegram was received from the Huth system.
data	*Data*	The received telegram in raw format.

POS data

Action name:POSData(POSName, TimeStamp, Article, Price, Units, PricePerUnit, Line1, Line2, Line3, Line4, Line5)
Action category: logical
The POS Management System sends the action for each transaction carried out at a cash point
Via the parameter "POS"the affected cash point will be identified. The further parameter will be filled with video meta data by the POS Management System

Parameter		Function
POS	*POSName*	Alphanumerical identification of the cash point
time stamp	*TimeStamp*	Time Stamp, when the action was send from the POS management system
article	*Article*	Identification of the booked article (optional)
price	*Price*	Amount (single price multiplied with number of articles) of transaction (optional)
units	*Units*	Amount of articles of the transaction (optional)
price per unit	*PricePerUnit*	Single article price of the transaction (optional)
line 1	*Line1*	Alphanumerical fields /sections for storing of additional information concerning the transaction or for storing information which have been printed out on the sales slip (optional)
line 2	*Line2*	Alphanumerical fields /sections for storing of additional information concerning the transaction or for storing inform-

Parameter		Function
		ation which have been printed out on the sales slip (optional)
line 3	*Line3*	Alphanumerical fields /sections for storing of additional information concerning the transaction or for storing information which have been printed out on the sales slip (optional)
line 4	*Line4*	Alphanumerical fields /sections for storing of additional information concerning the transaction or for storing information which have been printed out on the sales slip (optional)
line 5	*Line5*	Alphanumerical fields /sections for storing of additional information concerning the transaction or for storing information which have been printed out on the sales slip (optional)

POS status

Action name:POSStatus(POSName, TimeStamp, Status, Details)
Action category: logical
The POS management system sends the action as soon as the cash point is opened or closed or as soon as a cancellation will be made at a cash point.
Via the parameter "POS" the concerned cash point will be identified. The further parameter will be filled with video meta data from the POS management system.
The parameter "Status" can be addressed by the POS management system with a code figure for the currently notified status.

Parameter		Function
POS	*POSName*	Alphanumerical identification of cash point
time stamp	*TimeStamp*	Time Stamp, when the action was sent from the POS management system
status	*Status*	Identification figure for the currently notified status
details	*Details*	Additional alphanumerical information from POS management system (optional)

Terminal article data

Action name:TerminalArticleData(TerminalName, TimeStamp, CashierStation, PumpNo, AlarmStatus, Amount, Price, Details)
Action category: notification
The "Huth" driver sends the actions for each product-group-booking. Via the parameter "TerminalName" the affected device will be identified. The "HUTH" driver is principally able to build up several connections to different Huth devices . The driver sends the alphanumerical value defined in his setup as "Interface name". The further parameter will be filled with video meta data via the driver.

Parameter		Function
Terminal	*TerminalName*	Identifies the affected device. The "HUTH" driver is principally able to build up several connections to different Huth devices. The driver sends the alphanumerical value defined in his setup as "Interface name".
time stamp	*TimeStamp*	Time Stamp, when the status change was detected by the Huth-System
cashier station	*CashierStation*	Number of the cash point where the booking is carried out
pump no	*PumpNo*	Number of the filling point
alarm	*AlarmStatus*	Status of Alarm-Flags Yes = Alarm-Flag was set by the Huth system No = Alarm-Flag not set

Parameter		Function
amount	*Amount*	Amount of the booking (optional)
price	*Price*	Sum of the booking (optional)
details	*Details*	Free text (optional)

Terminal payment data

Action name:TerminalPaymentData(TerminalName, TimeStamp, CashierStation, PumpNo, AlarmStatus, Amount, Price, Details)
Action category: notification
The "HUTH" driver sends the action for each termination of a booking with the used method of payment. Via the parameter "TerminalName" the affected device will be identified. The "HUTH" driver is principally able to build up several connections to different Huth devices. The driver sends the alphanumerical value defined in his setup as "Interface name". The further parameter will be filled with video meta data via the driver.

Parameter		Function
Terminal	*TerminalName*	Identifies the affected device. The "HUTH" driver is principally able to build up several connections to different Huth devices. The driver sends the alphanumerical value defined in his setup as "Interface name".
time stamp	*TimeStamp*	Time Stamp, when the status change was detected by the Huth-System
cashier station	*CashierStation*	Number of the cash point where the booking is carried out with the used payment method
pump no	*PumpNo*	Number of the filling point (optional)
alarm	*AlarmStatus*	Status of Alarm-Flags Yes = Alarm-Flag was set by the Huth system No = Alarm-Flag not set
amount	*Amount*	Amount of the booking (optional)
price	*Price*	Sum of the booking (optional)
details	*Details*	Free text (optional)

Remote export

The actions of the category "Remote Export" subserve to start and control exports over the network. The actions are only at disposal if GSCRemEx service runs on every device and a connection to a central GeViSoft server persists. The GSCServer and GSCRemEx service have to run together on a local machine otherwise exports are not possible. The GSCRemEx service has to be setup in advance by GSCRemExEditor. The exports can be executed by a PILOT center device or other software systems (SDK based, GEUTEBRUECK devices). The PILOT is a system management console of GEUTEBRUECK which simplifies the handling of complex security systems. The PILOT among others can be used to control GSCView. Especially in view of the fact of exports the user can define start and end points by the help of the PILOT through GSCRemEx ("SetExportMarker" action). GSCView remembers the points in time and inserts them to the action "StartRemoteExport". The action "StartRemoteExport" is initiated by GSCView after the PILOT has send the action "InitializeRemoteExport" by indirection via the GeViSoft server and GeViScope server to GSCView. GSCView sends the action "StartRemoteExport" to

the GSCRemEx service and triggers the appropriate export. Exports that have been started through GSCRemEx service can be started or aborted from other devices or software systems over the network. Exports that have been started locally in GSCView cannot be controlled from other devices or software systems. In the curse of an export process no new export can be started. This export has to be restarted after the running export process has been completed! The actions "SetExportMarker" and "InitializeRemoteExport" have been developed especially for the PILOT.

Cancel export

Action name:*CancelExport(ExportID, AbortFlag)*
Action category: command
Through this action the running export process with the specified export ID is being aborted if GSCView remote-controls the GSCRemEx service. If the GSCRemEx service is remote-controlled by an external application the external application has to send the action to abort the running export process.

Parameter		Function
export GUID	*ExportID*	ID of the export process that has to be aborted. The export GUID is being assigned on the action "StartRemoteExport". e.g.: 01E68451-2406-484d-A9BC-5140762931E0
abort flag	*AbortFlag*	reason for abort 0: user abort; abort of export through user 1: low disc space; too little storage capacity 2: no user rights; access based on restricted user rights not possible 3: error; internal error

Export finished

Action name:*ExportFinished(ExportID, Success)*
Action category: notification
The GSCRemEx service notifies through this action that the running process was completed.
Possible status messages are: user abort, low disc space, no user rights, error.

Parameter		Function
export GUID	*ExportID*	ID of completed export process. The export GUID is being assigned on the action "StartRemoteExport". e.g.: 01E68451-2406-484d-A9BC-5140762931E0
success	*Success*	reason for abort 0: user abort; abort of export through user 1: low disc space; too little storage capacity 2: no user rights; access based on restricted user rights not possible 3: error; internal error

Export progress

Action name:*ExportProgress(ExportID, Progress)*
Action category: notification
The GSCRemEx service notifies the current status of the running export process in %.

Parameter		Function
export GUID	*ExportID*	ID of running export. The export GUID is being assigned on the action "StartRemoteExport". e.g.: 01E68451-2406-484d-A9BC-5140762931E0
progress	*Progress*	shows current status of the export process in %

Initialize remote export

Action name:*InitializeRemoteExport(Viewer, Device)*
Action category: command
This action is being used especially in the context of control units or systems like for example the PILOT.
The PILOT center device notifies GSCView that a new export has to be initiated. Thereupon GSCView creates the action "StartRemoteExport" with the appropriate parameters.

Parameter		Function
viewer	*Viewer*	global viewer number
device GUID	*Device*	ID of the PILOT center device (transmitted by the PILOT itself) e.g.: 01E68451-2406-484d-A9BC-5140762931E0

Set export marker

Action name:*SetExportMarker(Viewer, Marker)*
Action category: command
This action is being used especially in the context of control units or systems like for example the PILOT.
It indicates GSCView that an export start and end point has to be set on the current position of viewer X.
The so-called markers are being transferred automatically into the "StartRemoteExport" action once the "InitializeRemoteExport" action has been sent from the PILOT. The action "StartRemoteExport" transfers the start and end points to the GSCRemEx service which conducts the appropriate export.

Parameter		Function
viewer	*Viewer*	global viewer number
marker	*Marker*	tags and stores the start and end point of the data that has to be exported (selection begin=0, selection end=1)

Start remote export

Action name:*StartRemoteExport(ExportID, Device, BackupFormat, Channel, SelectionBegin, SelectionEnd, JobID)*
Action category: command
This action tells the GSCRemEx service to start a new export.
The action "StartRemoteExport" was created because the PILOT or another external software system did send the action "InitializeRemoteExport" to GSCView before.

Parameter		Function
export GUID	*ExportID*	ID of running exports. The export GUID has to be determined separately in advance because the action itself does not create a GUID.
device GUID	*Device*	ID of PILOT center device. If no PILOT is being used the blank GUID can be used instead. e.g.: 01E68451-2406-484d-A9BC-5140762931E0
format	*BackupFormat*	defines the format of the exported file 0=default (in this case it equals 1=GBF) 1=GBF(GEUTEBRUECK backup file) 2=MPEG2
channel	*Channel*	global channel number/camera number
start time	*SelectionBegin*	holds the position of the marker for the start point ("selection begin")
end time	*SelectionEnd*	holds the position of the marker for the end point ("selection end")

Parameter		Function
job ID	*JobID*	Contains the login data (server name, user name, encoded password) Optional second user password. The login data is separated by |. e.g.: <server name>|<user>|<PW>|<user2>|<PW2> localhost|admin|test| If there is no second user (second user password) nothing has to be entered at this point. Passwords in this parameter are encoded. Therefor the function DBIEncodeString() of GscDBI-DLL (from GeViScope/re_porter SDK) is being used.

Start scene store

Action name:*StartSceneStore(SceneStoreID, CutList, PreHistoryLength, RecordingLength)*
Action category: command
For internal use only!

Parameter		Function
scene store GUID	*SceneStoreID*	Scene store GUID.
cut-list	*CutList*	Cut-list.
pre-history length	*PreHistoryLength*	Pre-history length.
recording length	*RecordingLength*	Recording length.

SKIDATA

SKIDATA messages.

SKIDATA control

Action name:*SkidataControl(InterfaceName, Data)*
Action category: logical
SKIDATA control information.

Parameter		Function
interface	*InterfaceName*	Interface name.
state	*Data*	Interface state.

SKIDATA device event

Action name:SkidataDeviceEvent(InterfaceName, DeviceID, EventCode)
Action category: logical

SKIDATA device event.

Parameter		Function
interface	*InterfaceName*	Interface name.
device	*DeviceID*	Device ID.
event code	*EventCode*	Event code.

SKIDATA entry

Action name:*SkidataEntry(InterfaceName, MessageCode, TranscactionID, CarParkNo, DeviceID)*

Action category: logical
SKIDATA entry.

Parameter		Function
interface	*InterfaceName*	Interface name.
message	*MessageCode*	Message code.
transaction	*TranscactionID*	Transcaction ID.
car park	*CarParkNo*	Car park no.
device	*DeviceID*	Device ID.

SKIDATA exit

Action name:*SkidataExit(InterfaceName, MessageCode, TranscactionID, CarParkNo, DeviceID)*
Action category: logical
SKIDATA exit.

Parameter		Function
interface	*InterfaceName*	Interface name.
message	*MessageCode*	Message code.
transaction	*TranscactionID*	Transcaction ID.
car park	*CarParkNo*	Car park no.
device	*DeviceID*	Device ID.

SKIDATA transaction

Action name:*SkidataTransaction(InterfaceName, MessageCode, TranscactionID, CarParkNo, DeviceID)*
Action category: logical
SKIDATA transaction.

Parameter		Function
interface	*InterfaceName*	Interface name.
message	*MessageCode*	Message code.
transaction	*TranscactionID*	Transcaction ID.
car park	*CarParkNo*	Car park no.
device	*DeviceID*	Device ID.

System actions

All actions describing system behaviour.

Custom action

Action name:CustomAction(Int, String)
Action category: logical
This action has no side effects and can be used for customer purposes.

Parameter		Function
INT parameter	Int	Numeric parameter.
STRING parameter	String	Literal parameter.

Database recording info per ring

Action name:DatabaseRecordingInfoRing(DatabaseRing, NoVideoRecording, NoAudioRecording, NoRecordingAtAll, VideoSamplesPerSecond, VideoMBPerSecond, AudioSamplesPerSecond, AudioMBPerSecond, WriteWaitTimesPercent, RingCapacity, OldestItem, RecordingDepth, EstimatedRequiredCapacity)

Action category: logical
Database recording info per ring.

Parameter		Function
database ring	DatabaseRing	Database ring.
no video recording	NoVideoRecording	Video is recording or not.
no audio recording	NoAudioRecording	Audio is recording or not.
no recording	NoRecordingAtAll	Video and/or audio is recording or not.
video samples/s	VideoSamplesPerSecond	Video samples per second.
video samples MB/s	VideoMBPerSecond	Video MB per second.
audio samples/s	AudioSamplesPerSecond	Audio samples per second.
audio samples MB/s	AudioMBPerSecond	Audio MB per second.
write wait %	WriteWaitTimesPercent	Write wait times in percent.
ring capacity	RingCapacity	Ring capacity.
oldest item	OldestItem	Time stamp of the oldest item.
recording depth	RecordingDepth	Recording depth in hours.
estimated required capacity	EstimatedRequiredCapacity	Estimated required capacity.

Database recording info total

Action name:DatabaseRecordingInfoTotal(NoVideoRecording, NoAudioRecording, NoRecordingAtAll, VideoSamplesPerSecond, VideoMBPerSecond, AudioSamplesPerSecond, AudioMBPerSecond, WriteWaitTimesPercent, TotalCapacity, FreeCapacity, AllocatedCapacity, OldestItem, RecordingDepth, EstimatedRequiredCapacity, RequiredCapacityFactor, RequiredCapacityAvailable)
Action category: logical
Database recording info total.

Parameter		Function
no video recording	NoVideoRecording	Video is recording or not.
no audio recording	NoAudioRecording	Audio is recording or not.
no recording	NoRecordingAtAll	Video and/or audio is recording or not.
video samples/s	VideoSamplesPerSecond	Video samples per second.
video samples MB/s	VideoMBPerSecond	Video MB per second.
audio samples/s	AudioSamplesPerSecond	Audio samples per second.
audio samples MB/s	AudioMBPerSecond	Audio MB per second.
write wait %	WriteWaitTimesPercent	Write wait times in percent.
total capacity	TotalCapacity	Total capacity.
free capacity	FreeCapacity	Free capacity.
allocated capacity	AllocatedCapacity	Allocated capacity.
oldest item	OldestItem	Time stamp of the oldest item.
recording depth	RecordingDepth	Recording depth in hours.
estimated required capacity	EstimatedRequiredCapacity	Estimated required capacity.
required capacity factor	RequiredCapacityFactor	Required capacity factor.
required capacity available	RequiredCapacityAvailable	Required capacity available.

Database started

Action name:DatabaseStarted(Status, TotalSize)
Action category: logical

This action will be fired at the database start-up.

Parameter		Function
status	Status	Database status message.
total size	TotalSize	Database total size.

Event recording changed

Action name:EventRecordingChanged(EventID, TypeID)
Action category: logical
Event recording settings are changed.

Parameter		Function
instance ID	EventID	Instance ID of the event.
event type	TypeID	Type of the event.

Event started

Action name:EventStarted(EventID, TypeID, ForeignKey)
Action category: logical
Event has started.

Parameter		Function
instance ID	EventID	Instance ID of the event.
event type	TypeID	Type of the event.
foreign key	ForeignKey	Optional foreign key used to start the alarm.

Event stopped

Action name:EventStopped(EventID, TypeID)
Action category: logical
Event has stopped.

Parameter		Function
instance ID	EventID	Instance ID of the event.
event type	TypeID	Type of the event.

FRC notification

Action name:FRCNotification(Notification, Param, Description, XMLInfo)
Action category: logical
FRC notification.

Parameter		Function
notification	Notification	Notification reason.
param	Param	Additional parameter.
description	Description	Optional notification text.
additional info	XMLInfo	Optional additional info (usually as XML string).

GEMOS alarm

Action name:GEMOSalarm(GEMOSkey, GEMOSint, GEMOSstr)
Action category: logical
GEMOS alarm notification.

Parameter		Function
GEMOS key	GEMOSkey	GEMOS alarm key.
GEMOS int	GEMOSint	GEMOS alarm integer parameter.
GEMOS str	GEMOSstr	GEMOS alarm string parameter.

Kill all events

Action name:KillAllEvents()
Action category: logical
Kill all active events.

Kill event

Action name:KillEvent(TypeID)
Action category: logical
Kill event.

Parameter		Function
event type	TypeID	Type of the event.

Kill event by instance

Action name:KillEventByID(EventID)
Action category: logical
Kill event by instance ID.

Parameter		Function
instance ID	EventID	Instance ID of the event.

Live check

Action name:LiveCheck(Counter, Date)
Action category: logical
This action will be fired every 10 seconds and intended for use as live check.

Parameter		Function
counter	Counter	This is the number of already fired live check actions.
time stamp	Date	Current server time.

Set clock

Action name:SetClock(Date)
Action category: logical

Set clock.

Parameter		Function
current time	Date	Current time.

Setup changed

Action name:SetupChanged(User, Host, Date, ResourceKind, ResourceID, ChangeKind, Details, ClientHost, ClientType, ClientAccount)
Action category: logical
Setup changed.

Parameter		Function
user name	User	Name of the user modified the setup.
remote host	Host	Host from where the connection was done.
current time	Date	Current time.
resource kind	ResourceKind	Modified resource kind.
resource ID	ResourceID	Modified resource ID.
change kind	ChangeKind	Change kind.
details	Details	Details of the modification.
client host	ClientHost	Host from where the connection is done.

Parameter		Function
client type	ClientType	Client type.
client account	ClientAccount	User account from where the connection is done.

Setup upload progress

Action name:SetupUploadProgress(User1, User2, Host, Progress, Date)
Action category: logical
Setup upload progress.

Parameter		Function
first user	User1	Name of the user modified the setup.
second user	User2	Name of the second user by four eyes authentication.
remote host	Host	Host from where the connection was done.
progress %	Progress	Progress in percent.
current time	Date	Current stage time.

Set watchdog

Action name:SetWatchdog(Timeout)
Action category: logical
Set watchdog.

Parameter		Function
timeout	Timeout	Timeout in seconds, before the watchdog must be retriggered and before the hardware watchdog will set the hardware contact.

SMRP viewer cleared

Action name:SMRPViewerCleared()
Action category: logical
SMRP viewer cleared.

SMRP viewer connected

Action name:SMRPViewerConnected(Server, Channel)
Action category: logical
SMRP viewer connected to the camera.

Parameter		Function
server	Server	Server name.
channel	Channel	Channel.

SMTP mail

Action name:SMTPMailSend(Subject, To, Cc, Body, Channel)
Action category: logical
This action will send a user defined email if GscMail is connected

Parameter		Function
subject	Subject	Mail subject.
to	To	Mail recepients.
cc	Cc	Carbon copy recepients.
body	Body	Mail body.
channel	Channel	Channel.

Start event

Action name:StartEvent(TypeID, ForeignKey)
Action category: logical

Start event.

Parameter		Function
event type	TypeID	Type of the event.
foreign key	ForeignKey	Optional foreign key used to store for the alarm.

Stop all events

Action name:StopAllEvents()
Action category: logical
Stop all active events.

Stop event

Action name:StopEvent(TypeID)
Action category: logical
Stop event.

Parameter		Function
event type	TypeID	Type of the event.

Stop event by instance

Action name:StopEventByID(EventID)
Action category: logical
Stop event by instance ID.

Parameter		Function
instance ID	EventID	Instance ID of the event.

System error

Action name:SystemError(Source, Message, WindowsError, Description, XMLInfo)
Action category: logical
Notify system error.

Parameter		Function
source subsystem	Source	Source of the message.
message code	Message	Kind of the message.
Windows error code	WindowsError	Optional Windows error code.
description	Description	Optional description of the message.
additional info	XMLInfo	Optional additional info (usually as XML string).

System info

Action name:SystemInfo(Source, Message, Description, XMLInfo)
Action category: logical
Notify system information.

Parameter		Function
source subsystem	Source	Source of the message.
message code	Message	Kind of the message.
description	Description	Optional description of the message.
additional info	XMLInfo	Optional additional info (usually as XML string).

System settings changed

Action name:SystemSettingsChanged(SetupChanged, User1, User2, Host, TimeRangeChanged, TimeRange, LicenceChanged, Date)
Action category: logical
Setup of the system and/or the current time range changed.

Parameter		Function
setup changed	SetupChanged	System setup has changed.
first user	User1	Name of the user modified the setup.
second user	User2	Name of the second user by four eyes authentication.
remote host	Host	Host from where the connection was done.
time range changed	TimeRangeChanged	Time range has changed.
current time range	TimeRange	Currently active time range.
licence changed	LicenceChanged	Licence has changed.
change time	Date	Time of the system settings changed.

System started

Action name:SystemStarted(Date)
Action category: logical
This action will be fired only once at the system start-up.

Parameter		Function
start time	Date	Time of the system start-up.

System terminating

Action name:SystemTerminating(Date, WindowsShutdown)
Action category: logical
This action will be fired when the system is going shutdown.

Parameter		Function
stop time	Date	Time of the system shutdown.
Windows shut-down	WindowsShutdown	Indicates whether the system shutdown is done due to the windows shutdown.

System warning

Action name:SystemWarning(Source, Message, WindowsError, Description, XMLInfo)
Action category: logical
Notify system warning.

Parameter		Function
source subsystem	Source	Source of the message.
message code	Message	Kind of the message.
Windows error code	WindowsError	Optional Windows error code.
description	Description	Optional description of the message.
additional info	XMLInfo	Optional additional info (usually as XML string).

Transfer binary buffer

Action name:TransferBinaryBuffer(InternalHandle, Parameter)
Action category: logical
Transfer binary buffer.

Parameter		Function
internal handle	InternalHandle	Internal handle.
parameter	Parameter	Parameter.

Transfer binary channel buffer

Action name:TransferBinaryChannelBuffer(Channel, InternalHandle, Parameter)
Action category: logical
Transfer binary channel buffer.

Parameter		Function
channel	Channel	Channel.
internal handle	InternalHandle	Internal handle.
parameter	Parameter	Parameter.

User login

Action name:UserLogin(User1, User2, Host, ClientHost, ClientType, ClientAccount)
Action category: logical
This action will be fired when the user has connected to the system.

Parameter		Function
first user	User1	Name of the user connected to the system.
second user	User2	Name of the second user by four eyes authentication.
remote host	Host	Host from where the connection is done.
client host	ClientHost	Host from where the connection is done.
client type	ClientType	Client type.
client account	ClientAccount	User account from where the connection is done.

User login failed

Action name:UserLoginFailed(User1, User2, Host, RejectReason, ClientHost, ClientType, ClientAccount)
Action category: logical
This action will be fired when the user has tried to connect to the system but was rejected.

Parameter		Function
first user	User1	Name of the user tried to connect to the system.
second user	User2	Name of the second user by four eyes authentication.
remote host	Host	Host from where the connection is done.
reject reason	RejectReason	Reason of the rejection.
client host	ClientHost	Host from where the connection is done.
client type	ClientType	Client type.
client account	ClientAccount	User account from where the connection is done.

User logout

Action name:UserLogout(User1, User2, Host, ClientHost, ClientType, ClientAccount)
Action category: logical
This action will be fired when the user has disconnected from the system.

Parameter		Function
first user	User1	Name of the user disconnected from the system.
second user	User2	Name of the second user by four eyes authentication.
remote host	Host	Host from where the connection was done.
client host	ClientHost	Host from where the connection is done.
client type	ClientType	Client type.
client account	ClientAccount	User account from where the connection is done.

Video control actions

All actions to control the video streams, also all notifications about the state change of the video streams.

Activate external process

Action name:ActivateExternalProcess(Channel, TimeStamp, ExternalSystem)
Action category: logical
Activate external process.

Parameter		Function
channel	Channel	Channel.
time stamp	TimeStamp	Time stamp.
external system	ExternalSystem	External system to activate.

Change AD parameter set

Action name:ChangeADParameterSet(Channel, ParameterSet)
Action category: logical
This action changes the current AD parameter set of the video channel.

Parameter		Function
channel	Channel	Channel.
AD parameter set	ParameterSet	The name of the new AD parameter set.

Change camera profile

Action name:ChangeCameraProfile(HardwareModule, CameraProfile)
Action category: logical
This action changes the current camera profile of the hardware module.

Parameter		Function
hardware	HardwareModule	Hardware module.
profile	CameraProfile	The name of the camera profile.

Change CPA parameter set

Action name:ChangeCPAParameterSet(Channel, ParameterSet)
Action category: logical
This action changes the current CPA parameter set of the video channel.

Parameter		Function
channel	Channel	Channel.
CPA parameter set	ParameterSet	The name of the new CPA parameter set.

Change OBTRACK parameter set

Action name:ChangeObtrackParameterSet(Channel, ParameterSet)
Action category: logical
This action changes the current OBTRACK parameter set of the video channel.

Parameter		Function
channel	Channel	Channel.
OBTRACK parameter set	ParameterSet	The name of the new OBTRACK parameter set.

Change VMD parameter set

Action name:ChangeVMDParameterSet(Channel, ParameterSet)
Action category: logical
This action changes the current VMD parameter set of the video channel.

Parameter		Function
channel	Channel	Channel.
VMD parameter set	ParameterSet	The name of the new VMD parameter set.

Channel error

Action name:ChannelError(Channel, SensorType, Source, Message, WindowsError, Description, XMLInfo)
Action category: logical
Notify channel error.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
source subsystem	Source	Source of the message.
message code	Message	Kind of the message.
Windows error code	WindowsError	Optional Windows error code.
description	Description	Optional description of the message.
additional info	XMLInfo	Optional additional info (usually as XML string).

Channel info

Action name:ChannelInfo(Channel, SensorType, Source, Message, Description, XMLInfo)
Action category: logical
Notify channel information.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
source subsystem	Source	Source of the message.
message code	Message	Kind of the message.
description	Description	Optional description of the message.
additional info	XMLInfo	Optional additional info (usually as XML string).

Channel live check

Action name:ChannelLiveCheck(Channel, SensorType, TimeStamp)
Action category: logical
This action notifies that the channbel is alive.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
time stamp	TimeStamp	Time stamp.

Channel warning

Action name:ChannelWarning(Channel, SensorType, Source, Message, WindowsError, Description, XMLInfo)
Action category: logical
Notify channel warning.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
source subsystem	Source	Source of the message.
message code	Message	Kind of the message.
Windows error code	WindowsError	Optional Windows error code.
description	Description	Optional description of the message.
additional info	XMLInfo	Optional additional info (usually as XML string).

CPA measurement

Action name:CPAMeasurement(Channel, Correlation)
Action category: logical
CPA measurement.

Parameter		Function
channel	Channel	Channel.
correlation	Correlation	Correlation factor.

IAS settings changed

Action name:IASSettingsChanged(Channel, SensorType)
Action category: logical
IAS settings changed.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.

IP camera raw command

Action name:IPCameraRawCommand(URL, User, Password, POST)
Action category: logical

This action sends a special command to the IP camera.

Parameter		Function
url	URL	Complete command URL (like http://192.168.0.165:80/-set?daynight=night).
user	User	User name to authenticate by the camera (optional).
password	Password	Password to authenticate by the camera (optional).
post	POST	POST parameters (optional, separate lines with \\r\\n).

Make CPA reference image

Action name:MakeCPAReferenceImage(Channel)
Action category: logical
Make CPA reference image.

Parameter		Function
channel	Channel	Channel.

Media channel setup

Action name:MediaChannelSetupInfo(Channel, TimeStamp, Parameter)
Action category: logical
Media channel setup info.

Parameter		Function
channel	Channel	Channel.
time stamp	TimeStamp	Time stamp.
parameter	Parameter	Parameter.

NPR raw data

Action name:NPRRawData(PlateNo, Country, Channel, TimeStamp, ZoneRect, Weight, ZoneState, ZonePlace, Speed, Direction, ZoneIndex, CurBest, PlateWidth, PlateHeight, PlateAngle, SymHeight, Type)
Action category: logical
NPR raw data.

Parameter		Function
plate no.	PlateNo	Recognized plate no.
country	Country	Country.
channel	Channel	Channel.
time stamp	TimeStamp	Time stamp.
zone rect	ZoneRect	Zone rectangle.
weight	Weight	Weight of recognition.
zone state	ZoneState	Zone state.
zone status	ZonePlace	Zone status.
speed	Speed	Speed in km/h
direction	Direction	Direction of the motion.
zone index	ZoneIndex	Zone index.
best	CurBest	Current recognition is best.
plate width	PlateWidth	Plate width.
plate height	PlateHeight	Plate height.
plate angle	PlateAngle	Plate angle.
Symbol height	SymHeight	Symbol height.
type	Type	Number type.

NPR recognition

Action name:NPRRecognition(PlateNo, Country, Channel, TimeStamp, ZoneRect, Restriction, Category, Speed, Direction, ZoneIndex, Type, Weight)
Action category: logical
NPR recognition.

Parameter		Function
plate no.	PlateNo	Recognized plate no.
country	Country	Country.
channel	Channel	Channel.
time stamp	TimeStamp	Time stamp.
zone rect	ZoneRect	Zone rectangle.
restriction	Restriction	Restriction of recognized number.
category	Category	Category of recognized number.
speed	Speed	Speed in km/h
direction	Direction	Direction of the motion.
zone index	ZoneIndex	Zone index.
type	Type	Number type.
weight	Weight	Weight of recognition.

OBTRACK channel counter

Action name:ObtrackChannelCounter(Channel, CounterType, CounterValue, ObjectDirection, TimeStamp, ResetTimeStamp)
Action category: logical
OBTRACK channel counter.

Parameter		Function
channel	Channel	Channel.
counter type	CounterType	Counter type.
counter value	CounterValue	Counter value.
object direction	ObjectDirection	Object direction.

Parameter		Function
time stamp	TimeStamp	Time stamp.
reset time stamp	ResetTimeStamp	Reset time stamp.

OBTRACK channel counter threshold

Action name:ObtrackChannelCounterThreshold(Channel, CounterType, CounterValue, ExceedingDirection, TimeStamp)
Action category: logical
OBTRACK channel counter threshold.

Parameter		Function
channel	Channel	Channel.
counter type	CounterType	Counter type.
counter value	CounterValue	Counter value.
exceeding direction	ExceedingDirection	Exceeding direction.
time stamp	TimeStamp	Time stamp.

OBTRACK channel set counter

Action name:ObtrackChannelSetCounter(Channel, CounterType, CounterValue, TimeStamp)
Action category: logical
OBTRACK channel set counter.

Parameter		Function
channel	Channel	Channel.
counter type	CounterType	Counter type.
counter value	CounterValue	Counter value.
time stamp	TimeStamp	Time stamp.

OBTRACK frame raw data

Action name:ObtrackFrameRawData(TimeStamp, Channel, Brightness, Contrast)
Action category: logical
OBTRACK frame raw data.

Parameter		Function
time stamp	TimeStamp	Time stamp.
channel	Channel	Channel.
brightness	Brightness	Brightness.
contrast	Contrast	Contrast.

OBTRACK group counter

Action name:ObtrackGroupCounter(GroupId, CounterType, CounterValue, ObjectDirection, TimeStamp, ResetTimeStamp, GroupName)
Action category: logical
OBTRACK group counter.

Parameter		Function
group id	GroupId	Group ID.
counter type	CounterType	Counter type.
counter value	CounterValue	Counter value.
object direction	ObjectDirection	Object direction.
time stamp	TimeStamp	Time stamp.
reset time stamp	ResetTimeStamp	Reset time stamp.
group name	GroupName	Group name.

OBTRACK group counter threshold

Action name:ObtrackGroupCounterThreshold(GroupId, CounterType, CounterValue, ExceedingDirection, TimeStamp, GroupName)
Action category: logical
OBTRACK group counter threshold.

Parameter		Function
group id	GroupId	Group ID.
counter type	CounterType	Counter type.
counter value	CounterValue	Counter Value.
exceeding direction	ExceedingDirection	Exceeding direction.
time stamp	TimeStamp	Time stamp.
group name	GroupName	Group name.

OBTRACK group set counter

Action name:ObtrackGroupSetCounter(GroupId, CounterType, CounterValue, TimeStamp, GroupName)
Action category: logical
OBTRACK group set counter.

Parameter		Function
group id	GroupId	Group ID.
counter type	CounterType	Counter type.
counter value	CounterValue	Counter value.
time stamp	TimeStamp	Time stamp.
group name	GroupName	Group name.

OBTRACK object raw data

Action name:ObtrackObjectRawData(TimeStamp, Channel, Area, ObjectID, ObjectStatus, ObjectClass, Confidence, Position, Speed, Duration, Direction, Size, ObjectWidth, ObjectHeight, ProcessSize, GscNetName)
Action category: logical
OBTRACK object raw data.

Parameter		Function
time stamp	TimeStamp	Time stamp.
channel	Channel	Channel.
area no	Area	Area no.
object ID	ObjectID	Object ID.
object status	ObjectStatus	Object status.
object class	ObjectClass	Object class.
confidence	Confidence	Confidence.
position	Position	Position.
speed	Speed	Speed.
duration	Duration	Duration.
direction	Direction	Direction.
object size	Size	Object size.
object width	ObjectWidth	Object width in meters.
object height	ObjectHeight	Object height in meters.
process size	ProcessSize	Process size.
GSC net name	GscNetName	GeviScope network name.

OBTRACK tunnel alarm

Action name:ObtrackTunnelAlarm(Channel, TimeStamp, AlarmReason, ObjectID, AlarmAreaID, ObjectArea)
Action category: logical
OBTRACK tunnel alarm notification.

Parameter		Function
channel	Channel	Channel.
time stamp	TimeStamp	Time stamp.
alarm reason	AlarmReason	Alarm reason.
object ID	ObjectID	Object ID.
alarm area ID	AlarmAreaID	Alarm area ID.
object area	ObjectArea	Object area.

Sensor alarm finished

Action name:SensorAlarmFinished(Channel, SensorType)
Action category: logical
This action will be fired when the alarm is finished.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.

Sensor inhibit alarm finished

Action name:SensorInhibitAlarmFinished(Channel, SensorType)
Action category: logical
This action will be fired when the inhibit alarm finished.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.

Sensor inhibit video alarm

Action name:SensorInhibitVideoAlarm(Channel, SensorType, ADArea, ADCell, VMDGroup, VMDZone, VMDCycle, AlarmArea, ObjectClass)
Action category: logical
This action will be fired when the motion in inhibit area detected.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
AD alarm area	ADArea	AD alarm area.
AD cell	ADCell	AD cell nr.
VMD alarm group	VMDGroup	VMD alarm group.
VMD zone	VMDZone	VMD zone nr.
VMD cycle	VMDCycle	VMD measure cycle.
alarm area	AlarmArea	Alarm area.
object class	ObjectClass	OBTRACK object class.

Sensor video alarm

Action name:SensorVideoAlarm(Channel, SensorType, ADArea, ADCell, VMDGroup, VMDZone, VMDCycle, AlarmArea, ObjectClass)
Action category: logical

This action will be fired when video alarm is detected.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
AD alarm area	ADArea	AD alarm area.
AD cell	ADCell	AD cell nr.
VMD alarm group	VMDGroup	VMD alarm group.
VMD zone	VMDZone	VMD zone nr.
VMD cycle	VMDCycle	VMD measure cycle.
alarm area	AlarmArea	Alarm area.
object class	ObjectClass	OBTRACK object class.

Set system time

Action name:SetSystemTime(TimeStamp)
Action category: logical
Set system time.

Parameter		Function
time stamp	TimeStamp	Time stamp.

Set test picture mode

Action name:SetTestPictureMode(Channel, Mode)
Action category: logical
Enable or disable test picture generator.

Parameter		Function
channel	Channel	Channel.
enable	Mode	Enable or disable test picture generator.

Video contrast detected

Action name:VideoContrastDetected(Channel)
Action category: logical
This action will be fired when the contrast is detected in the video signal.

Parameter		Function
channel	Channel	Channel.

Video contrast failed

Action name:VideoContrastFailed(Channel)
Action category: logical
This action will be fired when the contrast is lost in the video signal.

Parameter		Function
Parameter		Function
channel	Channel	Channel.

Video set image brightness

Action name:VideoSetImageBrightness(Channel, SensorType, Brightness)
Action category: logical
Video set image brightness.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
brightness	Brightness	Brightness.

Video set image contrast

Action name:VideoSetImageContrast(Channel, SensorType, Contrast)
Action category: logical
Video set image contrast.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
contrast	Contrast	Contrast.

Video set image saturation

Action name:VideoSetImageSaturation(Channel, SensorType, Saturation)
Action category: logical
Video set image saturation.

Parameter		Function
channel	Channel	Channel.
sensor type	SensorType	Sensor type.
saturation	Saturation	Saturation.

Video source has changed

Action name:VideoSourceChanged(Channel, SignalNorm, SignalType, InterlaceType)
Action category: logical
This action indicates the changes on the video input source.

Parameter		Function
channel	Channel	Channel.
signal norm	SignalNorm	New signal norm.
signal type	SignalType	New signal type.
interlace type	InterlaceType	New interlace type.

Video sync detected

Action name:VideoSyncDetected(Channel)
Action category: logical
This action will be fired when the sync is detected in the video signal.

Parameter		Function
channel	Channel	Channel.

Video sync failed

Action name:VideoSyncFailed(Channel)
Action category: logical
This action will be fired when the sync is lost in the video signal.

Parameter		Function
channel	Channel	Channel.

Viewer actions

Viewer actions allow remote controlling GSCView. To enable remote controlling GSCView the "Remote control" setting in GscProfileManager and a global unique viewer client number has to be configured.

VC alarm queue confirm

Action name:VCAlarmQueueConfirm(Viewer, SelectionMode)
Action category: command
The alarm queue of the GSCView with the given viewer client number can be remote controlled.

A current alarm is confirmed. The parameter "selection mode" defines which alarm in the queue will be confirmed.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
selection mode	SelectionMode	first = first active alarm in queue next = next active alarm in queue (from current position) previous = previous alarm in queue (from current position) last = last active alarm in queue

VC alarm queue confirm by instance

Action name:VCAlarmQueueConfirmByInstance(Viewer, AlarmID)
Action category: command
The alarm queue of GSCView with the given viewer client number can be remote controlled.

A current alarm is confirmed. It is identified by its alarm instance ID (event instance ID). A unique instance ID is assigned to each alarm /recording event at creation time by the GeViScope server.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
instance ID	AlarmID	The alarm instance ID (event instance ID)

VC alarm queue confirm by type

Action name:VCAlarmQueueConfirmByType(Viewer, TypeID, SelectionMode)
Action category: command
The alarm queue of GSCView with the given viewer client number can be remote controlled.

A current alarm is confirmed. It is identified by its alarm type (event type) which means the name of the alarm (event) in the GeViScope Setup event list and also by the parameter "selection mode". The parameter "selection mode" defines which alarm in the queue will be selected.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
event type	TypeID	The alarm type (event type)
selection mode	SelectionMode	first = first active alarm in queue next = next active alarm in queue (from current position) previous = previous alarm in queue (from current position) last = last active alarm in queue

VC alarm queue remove

Action name:VCAlarmQueueRemove(Viewer, SelectionMode)
Action category: command
The alarm queue of the GSCView with the given viewer client number can be remote controlled.

An alarm is removed from the queue. The parameter "selection mode" defines which alarm in the queue will be removed.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
selection mode	SelectionMode	first = first active alarm in queue next = next active alarm in queue (from current position) previous = previous alarm in queue (from current position) last = last active alarm in queue

VC alarm queue remove by instance

Action name:VCAlarmQueueRemoveByInstance(Viewer, AlarmID)
Action category: command
The alarm queue of the GSCView with the given viewer client number can be remote controlled.

An alarm is removed from the queue. It is identified by its alarm instance ID (event instance ID). A unique instance ID is assigned to each alarm/event recording.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
instance ID	AlarmID	The alarm instance ID (event instance ID)

VC alarm queue remove by type

Action name:VCAlarmQueueRemoveByType(Viewer, TypeID, SelectionMode)
Action category: command
The alarm queue of the GSCView with the given viewer client number can be remote controlled.

An alarm is removed from the queue. It is identified by its alarm type (event type) which means the name of the alarm (event) in the GeViScope Setup event list. The parameter "selection mode" defines which alarm will be removed.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
event type	TypeID	The alarm type (event type)
selection mode	SelectionMode	first = first active alarm in queue next = next active alarm in queue (from current position) previous = previous alarm in queue (from current position) last = last active alarm in queue

VC alarm queue select

Action name:VCAlarmQueueSelect(Viewer, SelectionMode)
Action category: command
The alarm queue of the GSCView with the given viewer client number can be remote controlled.

An alarm of the queue is presented. The parameter "selection mode" defines which alarm in the queue will be presented.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
selection mode	SelectionMode	first = first active alarm in queue next = next active alarm in queue (from current position) previous = previous alarm in queue (from current position) last = last active alarm in queue

VC alarm queue select by instance

Action name:VCAlarmQueueSelectByInstance(Viewer, AlarmID)
Action category: command
The alarm queue of the GSCView with the given viewer client number can be remote controlled.

An alarm of the queue is presented. It is identified by its alarm instance ID (event instance ID). A unique instance ID is assigned to each alarm/event recording.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
instance ID	AlarmID	The alarm instance ID (event instance ID)

VC alarm queue select by type

Action name:VCAlarmQueueSelectByType(Viewer, TypeID, SelectionMode)
Action category: command
The alarm queue of the GSCView with the given viewer client number can be remote controlled.

An alarm of the queue is presented. It is identified by its alarm type (event type) which means the name of the alarm (event) in the GeViScope Setup event list and also by the parameter "selection mode". The parameter "selection mode" defines which alarm in the queue will be selected.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
event type	TypeID	The alarm type (event type)
selection mode	SelectionMode	first = first active alarm in queue next = next active alarm in queue (from current position) previous = previous alarm in queue (from current position) last = last active alarm in queue

VC change scene by name

Action name:VCChangeSceneByName(Viewer, Scene)
Action category: command
The action displays a scene in the GSCView with the given viewer client number.
The scene is identified by its name which is case insensitive. (e.g. "MyScene" equal "myscene")

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
scene	Scene	The name of the scene that should be displayed

VC clear scene by name

Action name:VCClearSceneByName(Viewer, Scene)
Action category: command
The action clears a scene in the GSCView with the given viewer client number.
The scene is identified by its name which is case insensitive. If the scene is currently not active it will be displayed after the action is executed. (e.g. "MyScene" equal "myscene")

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
scene	Scene	The name of the scene that should be cleared

VC full mode

Action name:VCFullMode(Viewer, FullMode, SensitiveAreaEnabled)
Action category: command
The GscView with the given viewer client number can be switched into full mode display or normal mode display.
In full mode display GscView offers the possibility to fade in controls like the tool bar or the side bar if the user moves the mouse cursor in the near of the window borders. This behavior can be controlled by the Parameter "Sensitive area enabled".

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GscView that should be remote controlled
full mode	FullMode	yes = switch to full mode display no = switch to normal mode display
sensitive area enabled	SensitiveAreaEnabled	yes = moving mouse cursor in the near of the window borders causes controls to fade in no = no controls fade in

VC set audio level

Action name:VCSetAudioLevel(Viewer, AudioLevel)
Action category: command
The volume of the audio output of the GSCView with the given viewer client number can be controlled.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
audio level	AudioLevel	0 = audio output off 100 = audio output in maximum volume

VC show viewer text

Action name:VCShowViewerText(Viewer, ShowText)
Action category: command
The text fade-in of all viewers of the GSCView with the given viewer client number can be switched on and off.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
show text	ShowText	yes = switch text fade-in on no = switch text fade-in off

VC stretch mode

Action name:VCStretchMode(Viewer, StretchMode)
Action category: command
The GSCView with the given viewer client number can be switched into stretched mode display or normal mode display.

In the stretched view, the viewers are stretched to the available size in the GSCView main window. In the normal mode display the viewers are sized in 4:3 ratio.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
stretch mode	StretchMode	yes = switch to stretched mode display no = switch to normal mode display

Viewer change scene

Action name:ViewerChangeScene(Viewer)
Action category: command
The action displays the scene where the viewer with the global number on any GSCView in the network belongs to.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled

Viewer clear

Action name:ViewerClear(Viewer)
Action category: command
The action clears the active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network

Viewer clear scene

Action name:ViewerClearScene(Viewer)
Action category: command

The action clears the scene where the active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network belongs to.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network

Viewer clear text output

Action name:ViewerClearTextOutput(Viewer)
Action category: command
The action doesn't display a text in the active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network anymore.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network

Viewer connect

Action name:ViewerConnect(Viewer, Channel, PlayMode)
Action category: command
Display pictures of a video channel on the active viewer of the GscView with the given viewer client number or on the viewer with the global number on some GscView in the network.
The parameter "play mode" defines in which mode the pictures are presented (live, forward, backward).

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GscView that should be remote controlled or Global number of a viewer on some GscView in the network
channel	Channel	Global number of the media channel
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped; if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the actual position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" but with high speed fast backward = like "play backward" but with high speed step forward = like "play forward" but only one picture step backward = like "play backward" but only one picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = like "play forward" but only pictures that belong to event recordings prev event = like "play backward" but only pictures that belong to event recordings peek live picture = display only one actual live picture next detected motion = like "play forward" but only pictures with motion in it

Parameter		Function
		(if no MOS search area is defined in GscView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GscView the whole picture size is used for it) are displayed; the display stops after motion is detected

Viewer connect live

Action name:ViewerConnectLive(Viewer, Channel)
Action category: command
This action displays live pictures of a video channel on the active viewer of the GSCView with the given viewer client number or on the viewer with the global number on any GSCView in the network.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network
channel	Channel	Global number of the media channel

Viewer export picture

Action name:ViewerExportPicture(Viewer, FilePath)
Action category: command
The action exports the current picture of the active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network.

The actual picture is exported as a windows bitmap graphic file in the GSCView directory or in the path (local or UNC) defined via the parameter "file path".

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network
file path	FilePath	Path (local or UNC) where the picture should be exported to

Viewer jump by time

Action name:ViewerJumpByTime(Viewer, Channel, PlayMode, TimeInSec)
Action category: command
The action displays pictures of a video channel on the active viewer of the GSCView with the given viewer client number or on the viewer with the global number on any GSCView in the network.

The parameter "play mode" defines in which mode the pictures are presented (live, forward, backward .).

The parameter "time in sec" defines the time span that the start of the replay should be moved from the actual timestamp.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled

Parameter		Function
		or Global number of a viewer on any GSCView in the network
channel	Channel	Global number of the media channel
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped? if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the current position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" high speed fast backward = like "play backward" high speed step forward = like "play forward" picture by picture step backward = like "play backward" picture by picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = jump to the next event recording prev event = jump to the previous event recording peek live picture = displays only one current live picture next detected motion = like "play forward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected
time in sec	TimeInSec	Time span that the start of the replay should be moved from the actual timestamp

Viewer maximize

Action name:ViewerMaximize(Viewer, Maximize)
Action category: command
The active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network which should be remote controlled.

The parameter "maximize" defines whether the viewer should be displayed in maximized mode or not.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network
maximize	Maximize	yes = display the viewer in maximized mode no = display the viewer in normal mode

Viewer play from time

Action name:ViewerPlayFromTime(Viewer, Channel, PlayMode, Time)
Action category: command
Display pictures of a video channel on the active viewer of the GscView with the given viewer client number or on the viewer with the global number on some GscView in the network.

The parameter "play mode" defines in which mode the pictures are presented (live, forward, backward, .).
The parameter "time" defines the timestamp where the replay of the recorded video should start.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GscView that should be remote controlled or Global number of a viewer on some GscView in the network
channel	Channel	Global number of the media channel
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped? if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the current position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" high speed fast backward = like "play backward" high speed step forward = like "play forward" picture by picture step backward = like "play backward" picture by picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = jump to the next event recording prev event = jump to the previous event recording peek live picture = displays only one current live picture next detected motion = like "play forward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected
time	Time	Timestamp where the replay of the recorded video should start. The parameter should be defined in the following format: "2009/02/13 07:22:00,594 GMT+01:00"

Viewer print picture

Action name:ViewerPrintPicture(Viewer)
Action category: command
The action prints out the current picture of the active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network.

The print out is done on the default printer of the PC on which GSCView is running.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network

Viewer select

Action name:ViewerSelect(Viewer)
Action category: command
The action declares the viewer with the global number on any GSCView in the network to the active viewer of the corresponding GSCView.

Parameter		Function
viewer	Viewer	Global number of a viewer on any GSCView in the network

Viewer set play mode

Action name:ViewerSetPlayMode(Viewer, PlayMode, PlaySpeed)
Action category: command
The action sets the "play mode" of the active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped? if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the current position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" high speed fast backward = like "play backward" high speed step forward = like "play forward" picture by picture step backward = like "play backward" picture by picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = jump to the next event recording prev event = jump to the previous event recording peek live picture = displays only one current live picture next detected motion = like "play forward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected
play speed	PlaySpeed	Speed rate for fast forward/backward (2.)

Viewer show alarm by instance

Action name:ViewerShowAlarmByInstance(Viewer, AlarmID, PlayMode)
Action category: command
The action displays pictures of an alarm on the GSCView with the given viewer client number in the network.

The alarm is identified by its alarm instance ID (event instance ID). Every alarm (event) is assigned a unique instance ID at creation time by the GeViScope server.

The parameter "play mode" defines in which mode the pictures are presented (live replay, replay event pictures, .).

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled

Parameter		Function
instance ID	AlarmID	The alarm instance ID (event instance ID)
play mode	PlayMode	Show alarm using default settings = display alarm pictures using the default settings defined in the GeViScope setup Live replay = display live pictures of the cameras belonging to the alarm configuration replay event pictures = replay the pictures belonging to the alarm (only once) continuous event replay = replay the pictures belonging to the alarm continuously in a loop show first alarm picture only = only display the first picture belonging to the alarm

Viewer show alarm by key

Action name:ViewerShowAlarmByKey(Viewer, ForeignKey, PlayMode)
Action category: command
The action displays pictures of an alarm on the GSCView with the given viewer client number in the network.

The alarm is identified by its "foreign key". The "foreign key" was assigned explicit to the alarm as the alarm was started.

The parameter "play mode" defines in which mode the pictures are presented (live replay, replay event pictures .).

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
foreign key	ForeignKey	The foreign key that was assigned to the alarm as the alarm was started
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped? if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the current position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" high speed fast backward = like "play backward" high speed step forward = like "play forward" picture by picture step backward = like "play backward" picture by picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = jump to the next event recording prev event = jump to the previous event recording peek live picture = displays only one current live picture next detected motion = like "play forward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected

Viewer show alarm by type

Action name:ViewerShowAlarmByType(Viewer, TypeID, ForeignKey, PlayMode)
Action category: command
The action displays pictures of an alarm on the GSCView with the given viewer client number in the network.

The alarm is identified by its alarm type and optional by its foreign key. The alarm type (event name) is defined in the GeViScope setup. The foreign key was assigned explicit to the alarm as the alarm was started. It is optional. If it is not set, the last alarm with the defined alarm type is displayed.

The parameter "play mode" defines in which mode the pictures are presented (live replay, replay event pictures .).

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled
alarm type	TypeID	Type (event name) of the alarm, defined in the GeViScope setup
foreign key	ForeignKey	The foreign key that was assigned to the alarm as the alarm was started
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped? if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the current position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" high speed fast backward = like "play backward" high speed step forward = like "play forward" picture by picture step backward = like "play backward" picture by picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = jump to the next event recording prev event = jump to the previous event recording peek live picture = displays only one current live picture next detected motion = like "play forward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GSCView the whole picture size is used for it) are displayed; the display stops after motion is detected

Viewer change sync audio/video

Action name:ViewerSyncAudioAndVideo(Viewer, EnableSync)
Action category: command
The active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network should be remote controlled.

The parameter "enable sync" defines whether audio and video should be synchronized or not.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network
enable sync	EnableSync	yes = synchronize audio and video no = don't synchronize audio and video

Viewer text output

Action name:ViewerTextOutput(Viewer, Text)
Action category: command
The action displays a text in the active viewer of the GSCView with the given viewer client number or the viewer with the global number on any GSCView in the network.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that should be remote controlled or Global number of a viewer on any GSCView in the network
text string	Text	Text that should be displayed in the picture

Viewer notification actions

Viewer notifications are fired by GSCView while GSCView is remote controlled. To enable remote controlling GSCView the "Remote control" setting in GscProfileManager and a global unique viewer client number has to be configured. To enable GSCView sending viewer notifications the "Send notification actions" settings in GscProfileManager have to be configured.

Image export notification

Action name:ImageExportNotification(User, Destination, DestinationType, TimeStamp, TimeEnd, Channels, ClientHost, ClientType, ClientAccount)
Action category: notification
A single image or a video sequence has been exported by a GSCView in the network.

GSCView has fired this notification because a single picture has been exported via a ViewerExportPicture action while GSCView is remote controlled or because the user has manually exported a picture or a video sequence in GSCView.

Parameter		Function
user	User	GeViScope user, who has done the export
destination	Destination	Path (local or UNC) where the picture or sequence was exported
destination type	DestinationType	0 = single image 1 = backup file (GBF) 2 = video file (MPEG, Video DVD, MPEG4CCTV raw) 3 = snapshot to clip-

Parameter		Function
		board 4 = print picture
time stamp	TimeStamp	Timestamp belonging to the picture exported or belonging to the first picture of the exported video sequence. The parameter is transmitted in the following format: "2009/05/06 14:47:48,359 GMT+02:00"
end time	TimeEnd	Timestamp belonging to the last picture of the exported video sequence. The parameter is transmitted in the following format: "2009/05/06 14:47:48,359 GMT+02:00"
channels	Channels	List of video channels that are included in the export result
client host	ClientHost	Host name of the PC where GSCView is running
client type	ClientType	1 = GSCView All other values are for future use!
client account	ClientAccount	Windows user account under that GSCView is running

Scene store modification

Action name:SceneStoreModification(Viewer, SceneStoreID, SceneStoreName, TimeStamp, ModificationType, User, ClientHost, ClientType, ClientAccount)
Action category: notification

Scene store modification.

Parameter		Function
viewer	Viewer	Global number of a viewer on some GSCView in the network
scene store GUID	SceneStoreID	Scene store GUID.
scene store name	SceneStoreName	Scene store name.
time stamp	TimeStamp	Time stamp.
modification type	ModificationType	Modification type.
user	User	Name of the user.
client host	ClientHost	Host name of the PC where GSCView is running
client type	ClientType	1 = GSCView All other values are for future use!
client account	ClientAccount	Windows user account under that GSCView is running

VC alarm queue notification

Action name:VCAlarmQueueNotification(Viewer, Notification, AlarmID, TypeID, ClientHost, ClientType, ClientAccount)
Action category: notification
The state of the alarm queue of the GSCView with the transmitted viewer client number has been changed.

GSCView has fired this notification because the state of its alarm queue has been changed via a VCAlarmQueue... action while GSCView is remote controlled or because the user has manually changed the state of the alarm queue in GSCView.

An alarm can be identified by its alarm instance ID (event instance ID). Every alarm (event) is assigned a unique instance ID at creation time by the GeViScope server.

Alternatively the alarm can be identified by its alarm type (event type) which means the name of the alarm (event) in the GeViScope Setup event list.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that fired this notification
notification	Notification	New alarm = an new alarm occurred Presented = an alarm was presented Stacked = an alarm was stacked in the queue, because the queue is blocked by an active alarm Confirmed = an alarm was confirmed Removed = an alarm was removed from the queue Last confirmed = the last alarm in the queue was confirmed Last removed = the last alarm was removed from the queue List confirmed = there are no more unconfirmed alarms in the queue List empty = there are no more alarms in the queue
instance ID	AlarmID	The alarm instance ID (event instance ID)
event type	TypeID	The alarm type (event type)
client host	ClientHost	Host name of the PC where GSCView is running
client type	ClientType	1 = GSCView All other values are for future use!

Parameter		Function
client account	ClientAccount	Windows user account under that GSCView is running

VC scene changed

Action name:VCSceneChanged(Viewer, Scene)
Action category: notification
The active scene of the GSCView with the transmitted viewer client number has been changed.

GSCView has fired this notification because its active scene has been changed via a VCChangeSceneByName or ViewerChangeScene action while GSCView is remote controlled or because the user has manually changed the active scene in GSCView.

Parameter		Function
viewer	Viewer	Global viewer client number, identifies the GSCView that fired this notification
scene	Scene	The name of the scene that is displayed after the change

Viewer cleared

Action name:ViewerCleared(Viewer, ClientHost, ClientType, ClientAccount)
Action category: notification
The viewer with the transmitted global number on some GSCView in the network has been cleared.

GSCView has fired this notification because one of its viewers has been cleared via a ViewerClear action while GSCView is remote controlled or because the user has manually cleared the viewer in GSCView.

Parameter		Function
viewer	Viewer	Global number of a viewer on some GSCView in the network
client host	ClientHost	Host name of the PC where GSCView is running
client type	ClientType	1 = GSCView All other values are for future use!
client account	ClientAccount	Windows user account under that GSCView is running

Viewer connected

Action name:ViewerConnected(Viewer, Channel, PlayMode, ClientHost, ClientType, ClientAccount)
Action category: notification
The viewer with the transmitted global number on some GSCView in the network has been connected.

GSCView has fired this notification because one of its viewers has been connected via a ViewerConnect or ViewerConnectLive action while GSCView is remote controlled or because the user has manually connected the viewer in GSCView.

The parameter "play mode" defines in which mode the pictures are presented (live, forward, backward, .).

Parameter		Function
viewer	Viewer	Global number of a viewer on some GSCView in the network
channel	Channel	Global number of the media channel
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped; if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the actual position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" but with high speed fast backward = like "play backward" but with high speed step forward = like "play forward" but only one picture step backward = like "play backward" but only one picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = like "play forward" but only pictures that belong to event recordings prev event = like "play

Parameter		Function
		backward" but only pictures that belong to event recordings peek live picture = display only one actual live picture next detected motion = like "play forward" but only pictures with motion in it (if no MOS search area is defined in GscView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GscView the whole picture size is used for it) are displayed; the display stops after motion is detected
client host	ClientHost	Host name of the PC where GSCView is running
client type	ClientType	1 = GSCView All other values are for future use!
client account	ClientAccount	Windows user account under that GSCView is running

Viewer play mode changed

Action name:ViewerPlayModeChanged(Viewer, Channel, PlayMode, ChannelTime, ClientHost, ClientType, ClientAccount)
Action category: notification
The playmode of the viewer with the transmitted global number on some GSCView in the network has been changed.

GSCView has fired this notification because the playmode of one of its viewers has been changed via a ViewerConnect, ViewerConnectLive, ViewerSetPlayMode, ViewerPlayFromTime, ViewerJumpByTime or one of the ViewerShowAlarmBy. actions while GSCView is remote controlled or because the user has manually changed the playmode of the viewer in GSCView.

Parameter		Function
viewer	Viewer	Global number of a viewer on some GSCView in the network
channel	Channel	Global number of the media channel, displayed in the viewer

Parameter		Function
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped; if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the actual position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" but with high speed fast backward = like "play backward" but with high speed step forward = like "play forward" but only one picture step backward = like "play backward" but only one picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = like "play forward" but only pictures that belong to event recordings prev event = like "play backward" but only pictures that belong to event recordings peek live picture = display only one actual live picture

Parameter		Function
		next detected motion = like "play forward" but only pictures with motion in it (if no MOS search area is defined in GscView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GscView the whole picture size is used for it) are displayed; the display stops after motion is detected
channel time	ChannelTime	Timestamp belonging to the picture presented in the viewer directly after the plamode had changed. The parameter is transmitted in the following format: "2009/05/06 14:47:48,359 GMT+02:00"
client host	ClientHost	Host name of the PC where GSCView is running
client type	ClientType	1 = GSCView All other values are for future use!
client account	ClientAccount	Windows user account under that GSCView is running

Viewer selection changed

Action name:ViewerSelectionChanged(Viewer, Channel, PlayMode, ClientHost, ClientType, ClientAccount)
Action category: notification

The active viewer on some GSCView in the network has been changed.
GSCView has fired this notification because the user has selected one of its viewers by mouse click or by dragging a camera onto one of its viewers.

GSCView only fires the notification, if a camera is displayed on the selected viewer.

Parameter		Function
viewer	Viewer	Global number of a viewer on some GSCView in the network
channel	Channel	Global number of the media channel, displayed in the viewer

Parameter		Function
play mode	PlayMode	play stop = if the viewer is already displaying pictures from that channel, it is stopped; if not the newest picture in the database is displayed play forward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed forward from the actual position; if not display of pictures with normal speed starts at the beginning of the database play backward = if the viewer is already displaying pictures from that channel, it is displaying pictures in normal speed backward from the actual position; if not display of pictures with normal speed starts at the end of the database fast forward = like "play forward" but with high speed fast backward = like "play backward" but with high speed step forward = like "play forward" but only one picture step backward = like "play backward" but only one picture play BOD = display the first (the oldest) picture in the database play EOD = display the last (the newest) picture in the database live = display live pictures next event = like "play forward" but only pictures that belong to event recordings prev event = like "play backward" but only pictures that belong to event recordings peek live picture = display only one actual live picture

Parameter		Function
		next detected motion = like "play forward" but only pictures with motion in it (if no MOS search area is defined in GscView the whole picture size is used for it) are displayed; the display stops after motion is detected prev detected motion = like "play backward" but only pictures with motion in it (if no MOS search area is defined in GscView the whole picture size is used for it) are displayed; the display stops after motion is detected
client host	ClientHost	Host name of the PC where GSCView is running
client type	ClientType	1 = GSCView All other values are for future use!
client account	ClientAccount	Windows user account under that GSCView is running