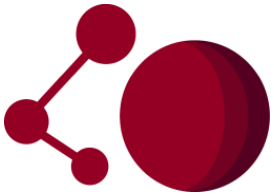

EyeLogic SDK

v 1.1.9

September 2023

1 EyeLogic SDK Documentation (C#) 1

1.1 Introduction . . . 1

1.1.1 About . . . 1

1.1.2 System Requirements . . . 1

1.2 Installation and Getting Started . . . 1

1.2.1 Download Software . . . 1

1.2.1.1 Compatibility . . . 2

1.2.2 Install EyeLogic SDK on Windows . . . 2

1.2.3 Getting Started with the Sample Code . . . 2

1.3 Concepts . . . 3

1.3.1 Server-Client Setup . . . 3

1.3.2 Set Up a Project for your Application . . . 3

1.3.3 Control Flow between API and server . . . 4

1.3.4 Dual PC Setup . . . 4

1.3.5 Example Program . . . 5

1.3.6 GazeSamples . . . 6

1.3.7 Shipping your Application . . . 6

1.4 Appendix . . . 6

1.4.1 License Agreement and Warranty for SDK . . . 6

1.5 About EyeLogic . . . 8

1.5.1 Contact and Support . . . 8

2 Namespace Index 9

2.1 Namespace List . . . 9

3 Hierarchical Index 11

3.1 Class Hierarchy . . . 11

4 Class Index 13

4.1 Class List . . . 13

5 File Index 15

5.1 File List . . . 15

6 Namespace Documentation 17

6.1 eyelogic Namespace Reference . . . 17

6.1.1 Detailed Description . . . 17

6.1.2 Enumeration Type Documentation . . . 17

6.1.2.1 EventType . . . 17

7 Class Documentation 19

7.1 ELCsApi.DeviceConfig Class Reference . . . 19

7.1.1 Detailed Description . . . 19

7.1.2 Member Data Documentation . . . 19

7.1.2.1 deviceSerial . . . 20
7.2 ELCsApi.DeviceGeometry Class Reference . . . 20
7.2.1 Detailed Description . . . 20
7.3 ELCsApi Class Reference . . . 20
7.3.1 Detailed Description . . . 22
7.3.2 Constructor & Destructor Documentation . . . 22
7.3.2.1 ELCsApi() . . . 22
7.3.3 Member Function Documentation . . . 22
7.3.3.1 calibrate() . . . 22
7.3.3.2 connect() . . . 23
7.3.3.3 connectRemote() . . . 23
7.3.3.4 getActiveScreen() . . . 23
7.3.3.5 getAvailableScreens() . . . 23
7.3.3.6 getDeviceConfig() . . . 24
7.3.3.7 requestServerList() . . . 24
7.3.3.8 requestTracking() . . . 24
7.3.3.9 setActiveScreen() . . . 25
7.3.3.10 unrequestTracking() . . . 25
7.3.3.11 validate() . . . 25
7.4 ELException Class Reference . . . 26
7.4.1 Detailed Description . . . 26
7.4.2 Member Enumeration Documentation . . . 26
7.4.2.1 ErrorType . . . 26
7.5 GazeSample Class Reference . . . 27
7.5.1 Detailed Description . . . 27
7.5.2 Member Data Documentation . . . 28
7.5.2.1 eyePositionLeft . . . 28
7.5.2.2 eyePositionRight . . . 28
7.6 Point2d Class Reference . . . 28
7.6.1 Detailed Description . . . 29
7.7 Point3d Class Reference . . . 29
7.7.1 Detailed Description . . . 29
7.8 ELCsApi.ScreenConfig Class Reference . . . 29
7.8.1 Detailed Description . . . 30
7.9 ELCsApi.ServerInfo Class Reference . . . 30
7.9.1 Detailed Description . . . 30
7.10 ELCsApi.ValidationPointResult Class Reference . . . 30
7.10.1 Detailed Description . . . 31
7.11 ELCsApi.ValidationResult Class Reference . . . 31
7.11.1 Detailed Description . . . 31

8 File Documentation **33**

8.1 ELCsApi.cs File Reference 33
8.1.1 Detailed Description 34

Index **35**

Chapter 1

EyeLogic SDK Documentation (C#)

1.1 Introduction

1.1.1 About

The EyeLogic Software Development Kit (SDK) is a free software package for building custom applications which use an EyeLogic eye tracking device. It offers the possibility to connect with your device via an application programming interface (API) from any custom application. The EyeLogic SDK is available for the programming languages C++, C#, C, and Python. It is also usable with any other programming language that is capable of importing dynamic link libraries (DLLs), e.g. Visual Basic or Matlab.

For each directly supported language, there is a short and simple sample program to help you get started with the development of your first EyeLogic application.

This guide describes the use of the EyeLogic API for C# and gives a step-by-step introduction on how to start with your own C# program.

1.1.2 System Requirements

For the system requirements of the EyeLogic Server and an installation guide, please refer to the Server's documentation.

The SDK has no additional requirements. The included C# sample project is built for Microsoft Windows only (64 bit) and uses Microsoft Visual Studio 2019 or newer. Any other compilers are not yet supported.

1.2 Installation and Getting Started

1.2.1 Download Software

In order to use an EyeLogic eye tracking device from within your application, you need the EyeLogic Server and the EyeLogic SDK. Check the download-page to get the latest release of both packages: https://www.eyelogicsolutions.com/downloads/

1.2.1.1 Compatibility

The software is written to support backwards-compatibility, i.e. an update of the EyeLogic Server software will not break support for your device, irregardless of the model. The actual guide assumes that you are installing the newest version of the EyeLogic Server. Please always update to the newest server version before reporting an error to the EyeLogic support.

On the other hand, updating the SDK and API-DLLs is not always neccessary. Since you as a programmer would have to recompile your application with every SDK-update, we designed the SDK such that the server is able to communicate with older API versions. Therefore, when shipping your application, just add the EyeLogic API DLLs of the actual version to your package. It is compatible with servers of the actual and newer releases.

See Shipping your Application for a tutorial on how to ship your application.

1.2.2 Install EyeLogic SDK on Windows

The EyeLogic SDK does not need to be installed. It ships as .zip file which just needs to be extracted to some directory on your hard disk. Be sure, that you have user-rights to that directory, e.g. any directory inside C:\Program Files or similar is problematic, since it requires admin rights to access those files every time you start your program. It is recommended to use a user-local directory.

Note: The SDK has to be installed on the same computer as the server. Please see the main server manual for help on installing the server.

After extracting the .zip file, the directory contains one subfolder for each supported programming language. Open the cs folder, the content should be:

- bin - contains the binary DLLs to link against
- example - contains the sample code

1.2.3 Getting Started with the Sample Code

In the directory, into which you unpacked the SDK `EyeLogicSDK`, navigate to the sub-directory `cs/example` and open the solution file `AllDemoClients.sln` in Visual Studio. Note, you will need Visual Studio 2019 or newer to open this file.

You may want to choose your destination compile level (Debug/Release) in the drop down list on top of the screen. Set it to "Debug" while developing your app. When your app is finished, set it to "Release" to create an optimized application binary. Then compile from the menu with Build->Build Solution. You should see an output similarly to:

```
Erstellen gestartet...
1>------ Erstellen gestartet: Projekt: DemoClient, Konfiguration: Debug x64 ------
2>------ Erstellen gestartet: Projekt: DualPC, Konfiguration: Debug x64 ------
3>------ Erstellen gestartet: Projekt: Validation, Konfiguration: Debug x64 ------
3>  Validation -> cs\example\Validation\bin\x64\Debug\DemoClientCs.exe
2>  DualPC -> cs\example\DualPC\bin\x64\Debug\DemoClientCs.exe
1>  DemoClient -> cs\example\DemoClient\bin\x64\Debug\DemoClientCs.exe
1>  ...\EyeLogic_SDK\cs\example\..\bin\concrt140.dll
1>  ...\EyeLogic_SDK\cs\example\..\bin\ELApi.dll
1>  ...\EyeLogic_SDK\cs\example\..\bin\ELCApi.dll
1>  ...\EyeLogic_SDK\cs\example\..\bin\ELCsApi.dll
1>  ...\EyeLogic_SDK\cs\example\..\bin\msvcp140.dll
1>  ...\EyeLogic_SDK\cs\example\..\bin\msvcp140_1.dll
1>  ...\EyeLogic_SDK\cs\example\..\bin\msvcp140_2.dll
1>  ...\EyeLogic_SDK\cs\example\..\bin\vccorlib140.dll
1>  ...\EyeLogic_SDK\cs\example\..\bin\vcruntime140.dll
1>          9 Datei(en) kopiert.
========== Erstellen: 1 erfolgreich, 0 fehlerhaft, 0 aktuell, 0 übersprungen ==========
```

Before running the application check that the EyeLogic Server is running (see the EyeLogic Server manual). If the server is running, there is an EyeLogic icon in the windows tray bar.

In the left part of the editor, there is a list of all projects / democlients. The active one is marked in bold (DemoClient). You might make any other demo client active (e.g. DualPC or Validation) by right-click on the desired name in the list and set it as the Startup Project.

Press F5 to compile and run the application.

Note that your firewall might block the connection between your program and the server. In this case, add a rule to your firewall to allow your application to open TCP/UDP ports to an application on localhost (for the windows defender, just click "accept").

If you reached this point, you have properly set up your EyeLogic SDK. You may now start with the development of you own application. See the next section Concepts for the basic programming concepts and for a tutorial on how to deploy and ship your application.

1.3 Concepts

1.3.1 Server-Client Setup

The EyeLogic software consists of two main parts: The `server` and the `API`. The server is the neccessary driver for your eye tracking device. It detects your device and handles the communication. The API is part of the EyeLogic Standard Development Kit (SDK). It consists of .dll files which can be used by your application to set up a connection to the EyeLogic Server, start tracking and receive eye tracking data.

The server is designed to run permanently on your computer as a background process. While not actively tracking the server requires an insignificant portion of your computer's resources. Once an EyeLogic eye tracking device is plugged in, the server application detects it automatically and allows the user to set it up via the servers' configuration dialog (see the server icon in the windows tray bar). If for any reason the server background process is not running (the tray icon is missing), you may start the server manually via the windows start menu.

The API is a set of .dll files which can be used by any custom program (called the `user application`). Using those DLLs the user application can establish a connection to the (running) server. Note that it the EyeLogic Server may run on the same computer than the user application, or they may run on different PCs. See Dual PC Setup for how to set up the setting with running the server and the user application on different computers.

1.3.2 Set Up a Project for your Application

For an easy start to develop a new application it is recommended to copy the existing sample folder to a new location (e.g. EyeLogic_SDK\cs with all its contents). The sample source file already provides a fully functional implementation. Starting from this sample code, you can easily modify and extend the code to suit your customized experiment.

Alternatively you can start a new Visual Studio project from scratch. In that case be sure the compiler is able to find the EyeLogic dll files. Therefore, apply the following changes to the project properties of your Visual Studio project:

- Select x64 as your target platform (it might require to add a new platform in your configuration manager)
- In your project explorer, right click "references", click "add". In the following window select "browse" and browse to the file `<Location of your EyeLogic_SDK>\cs\bin\ELCsApi.dll`.
- After compiling, copy all dll files from `<Location of your EyeLogic_SDK>\cs\bin` to your execution directory.

1.3.3 Control Flow between API and server

The usual control flow between the custom application/API and the server is characterized by the following steps:

1. **initialize:** Before calling any other function the API DLLs need initializing.
2. **connect to server:** Establish a connection to the server via TCP.
3. **find eye tracking device:** Obtain the information on connected eye trackers, otherwise wait until an eye tracker is plugged in.
4. **start tracking:** Request tracking. If successful, the device will start tracking and the server sends `GazeSamples` to the user application, see also GazeSamples.
5. **perform calibration:** Request a calibration. The screen will show a calibration point animated to be moving across the screen. The user must fixate on this point until the calibration screen diappears. The system is calibrated and ready to use once this process is completed sucessfully.
6. **shut down:** At the end of your experiment either stop the tracking or simply shutdown the API.

All information which is passed from the server to the user application will be transmitted via `asyncroneous callbacks`. The application has to register it's own implementations of those callback functions with the API (see Example Program for an example implementation).

Note that you need to calibrate in order to obtain valid gaze samples (see GazeSamples). All gaze samples which are reported before the system is calibrated contain no valid eye data.

1.3.4 Dual PC Setup

The Dual-PC setup is a special setting where the EyeLogic server runs on a different computer than the user application.

The most common use-case for the Dual PC Setup would be the following. Running an experiment with an operator who constrols the eye tracking device and a participant who has to perform a task. The participant uses a different PC (showing the experiment) than the operator (who can control the eye tracker via the EyeLogic Server software).

The computer of the operator (called Operator PC) needs to have the EyeLogic driver software (the EyeLogic Server) installed and running. The eye tracker is physically mounted to a screen which is connected to the computer of the participant (called Experiment PC). The USB cable of the eye tracker is plugged into the USB port of the Operator PC!

Now, the operator can use the server to detect the eye tracking device. On the Experiment PC, any custom application which shows an experiment to the participant, can use the EyeLogic API to connect to the server remotely. In order to do that, the application should use the API calls:

1. `requestServerList()` to obtain a list of all EyeLogic servers in the local network (LAN/WLAN) which are running and are configured to allow remote connections
2. `connectRemote()` to conntect to a specific server from that list
3. `setActiveScreen()` to set the screen connected to the Experiment PC as the active screen for eye tracking (replacing the default main screen of the Operator PC)

Note, that a server has to allow remote connections in order to be found. You can enable that in the settings of the server window.

If connected successfully, the client can operate as usual as if it would be connected to a local server. See the demo application "dualpc" in the SDK for an example.

1.3.5 Example Program

In this section, the code of the C# example program is explained in some detail.

The file starts with an include section. The important include is

```
using eyelogic;
```

which is needed to find all neccessary definitions of the EyeLogic API.

In the `run( )` method the application implements its control flow. It consists of the following code lines:

```
m_api = new ELCsApi( "C# Client" );
```

This constructs a new instance of the ELCsApi class. The instanciation will automatically initialize the library. The API needs to be initialized only once throughout the whole program. At the end, when finished, deinitialize the API with

```
m_api.destroy( );
```

The next two lines

```
m_api.OnEvent += onEvent;
m_api.OnGazeSample += onGazeSample;
```

register the callback functions which are invoked from the EyeLogic software whenever an event occurs and whenever a new gaze sample is incoming. Those functions are defined further below. The example code simply prints an incoming event to the main console, resp. count the incoming gaze samples.

```
m_api.connect( );
```

Connects to the EyeLogic server. If the connection fails, an ELException is thrown. If the method exits without an exception, then the connection is established.

```
ELCsApi.DeviceConfig deviceConfig = m_api.getDeviceConfig( );
```

obtains information about the connected eye tracking device. If there is no device connected, the method returns null.

```
ELCsApi.ScreenConfig screenConfig = m_api.getActiveScreen( );
```

obtains information about the active screen.

```
m_api.requestTracking( 0 );
```

Tells the device to start tracking and the server to begin sample processing. The parameter 0 specifies the frame rate mode. If your device is capable of multiple frame rate modes (60Hz, 120Hz or 250Hz), you can also enter a different number. The list of available frame rate modes is part of the DeviceConfig and can be obtained by calling getDeviceConfig(). The first frame rate mode (DeviceConfig.frameRates[0]) is the default mode, which usually is the highest available speed mode of your system.

```
const auto retCalibrate = api.calibrate( 0 );
```

Performs a calibration. This method blocks until the calibration ends - i.e. completed or aborted. The parameter 0 denotes the type of calibration. A list of available calibration methods is part of the DeviceConfig and can be obtained by calling getDeviceConfig( ).

The example program waits for 5 seconds and then closes the connection:

```
m_api.unrequestTracking( );
m_api.disconnect( );
m_api.destroy( );
```

1.3.6 GazeSamples

GazeSamples are the most essential data which is generated by the eye tracker. The eye tracker delivers one GazeSample per frame. Each sample contains information on the time of measurement, the position of the eyes, the pupil radius and the point where the user looks at on some stimulus plane (usually a computer monitor).

1.3.7 Shipping your Application

When you want to ship your application, be sure to include all relevant files so that it may run on different computers. The EyeLogic functionality will only work on computers which have the EyeLogic Server installed. The installed server needs to at least be of the same version as the shipped API DLLs (a newer server version is permissible).

Beside the relevant files of your application, you need to ship the content of the bin/ folder of your language (typically including some .dll files). Place the content of the bin/ folder inside the working directory of your application and ship them together.

1.4 Appendix

1.4.1 License Agreement and Warranty for SDK

IMPORTANT – PLEASE READ CAREFULLY:

The License Agreement is a legal agreement between you and EyeLogic GmbH and its affiliates (“EyeLogic”, “we”, or “us”). This license agreement governs your use of the EyeLogic software and any third party software that may be distributed therewith (collectively the “software”). EyeLogic agrees to license the software to you (personally and/or on behalf of you employer) (collectively “you” or “your”) only if you accept all the terms contained in this license agreement. By installing, using, copying, or distributing all or any portion of the software, you accept and agree to be bound by all of the terms and conditions of this license agreement.

If you do not agree with any of the terms of this license agreement, do no install or use the software.

1. **License Grant:** EyeLogic grants you a revocable, nonexclusive, non-transferable, limited right to install and use the application on a device owned and controlled by you, and to access and use the application on such mobile device strictly in accordance with the terms and conditions of this licenses, the usage rules and any service agreement associated with your device. The Software includes third party software and other copyrighted material. Acknowledgements, licensing terms and disclaimers for such Third Party Software are provided with the Software or contained in the Documentation, and your use of such Third Party Software is governed by their respective terms (collectively “Related Agreements”).
2. **Restriction on Use:** You shall use the application strictly in accordance with the terms of the related agreements and shall not:
   (a) decompile, reverse engineer, disassemble, attempt to derive the source code of, or decrypt the application,
   (b) make any modification, adaption, improvement, enhancement, translation or derivative work from the application,
   (c) violate any applicable laws, rules or regulations in connection with your access or use of the application,
   (d) remove, alter or obscure any proprietary notice (including any notice of copyright or trademark) of Eye↩ Logic or its affiliates, partners, suppliers or the licensors of the application,
   (e) use the application for any revenue generating endeavor, commercial enterprise or other purpose for which it is not designed or intended,

(f) make the application publicly available over a network or other environment permitting access or use by others without the written permission of EyeLogic,

(g) use the application for creating a product, service or software that is, directly or indirectly, competitive with or I any way substitute for any services, product or software offered by EyeLogic,

(h) use any proprietary information or interfaces of EyeLogic or other intellectual property of EyeLogic in the design, development, manufacture, licensing or distribution of any applications, accessories or devices for use with the application.

3. **Termination:** EyeLogic may, in its sole and absolute discretion, at any time and for any or no reason, suspend or terminate this license and the rights afforded to you hereunder with or without prior notice. Furthermore, if you fail to comply with any terms and conditions of this license, then this license and any rights afforded to you hereunder shall terminate automatically, without any notice or other action by EyeLogic. Upon the termination of this license, you shall cease all use of the application and uninstall the application.

4. **Disclaimer of Warranties:** You acknowledge and agree that the application is provided on an "as is" and "as available" basis, and that your use of or reliance upon the application and any third party content and services accessed thereby is at you sole risk and discretion. EyeLogic and its affiliates, partners suppliers and licensors hereby disclaim any and all representations, warranties and guaranties regarding the application and third party content and services, whether express, implied or statutory, and including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, furthermore, EyeLogic and its affiliates, partners, suppliers and licensors make no warranty that

   (a) The application or third party content and services will meet your requirements,

   (b) The application or third party content and services will be uninterrupted, accurate, reliable timely secure or error-free,

   (c) The quality of any products, services, information or other material accessed or obtained by you through the application will be as represented or meet your expectations, or

   (d) Any errors in the application or third party content and services will be corrected.

   No advice or information whether oral or written, obtained by you from EyeLogic or from the application will create any warranty not expressly made herein or create any liability on the part of EyeLogic.

   If the licensee modifies or replaces any of the third party open source software included in the software, Eye↩ Logic is not obligated to provide any updates, maintenance, warranty, technical or other support or services for the resultant modified Software. You expressly acknowledge that any failure or damage to any hardware, software or systems as a result of such modification to the open source components of the software is excluded from the terms of any EyeLogic warranty.

5. **Limitation of liability:** Under no circumstances shall EyeLogic or its affiliates, partners, suppliers or licensors be liable for any indirect, incidental, consequential, special or exemplary damages arising out of or in connection with your access or use of or inability to access or use the application and any third party content and services, whether or not the damages ere foreseeable and whether or not EyeLogic was advices of the possibility of such damages. Without limiting the generality of the foregoing, EyeLogic's aggregate liability to you (whether under contract, tort, statue or otherwise) shall not exceed the amounts actually paid by licensee for the licensed materials. The foregoing limitations will apply even if the above stated remedy fails of its essential purpose.

6. **Confidentiality:** Licensed materials are proprietary to EyeLogic and constitute EyeLogic trade and business secrets. The licensee shall maintain licensed materials in confidence and prevent their disclosure using at least the same degree of care it uses for its own trade and business secrets, but in no event less than a reasonable degree of care. The licensee shall not disclose licensed materials or any part thereof to anyone for any purpose, other than to its employees and sub-contractors, if any, for the purpose of exercising the rights expressly granted under this agreement, provided they have in writing agreed to confidentiality obligations at least equivalent to the obligations stated herein. The foregoing does not apply to information that a. is or becomes generally known or available to the public without any breach of the confidentiality obligation by licensee, b. was already known to licensee prior to the disclosure by EyeLogic, or c. was rightfully acquired by licensee from a third party without a breach of a confidentiality obligation towards EyeLogic. In case of a dispute, the licensee has the burden of proof that the licensed materials and/or any portion thereof fall under one of these exceptions. Should the licensee be legally compelled to disclose any licensed materials to a third party, such as pursuant to a mandatory order by a court or authority or any comparable action, the licensee

shall, to the extent permitted under applicable law, inform EyeLogic without undue delay and undertake all possible measures to safeguard secrecy.

1.5 About EyeLogic

EyeLogic is a manufacturer of high precision and high quality eye tracking devices, mainly for scientific and research use cases. EyeLogic GmbH is a spin-off of the Free University of Berlin, faculty of mathematics and computer science and has a vast experience in image processing and computer vision.

1.5.1 Contact and Support

For technical support questions contact us via mail at: support@eyelogicsolutions.com

EyeLogic GmbH
Schlesische Str. 28
10997 Berlin Germany
www: https://www.eyelogicsolutions.com

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

eyelogic
Namespace for C# API calls 17

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ELCsApi.DeviceConfig 19
ELCsApi.DeviceGeometry 20
ELCsApi 20
Exception
    ELException 26
GazeSample 27
Point2d 28
Point3d 29
ELCsApi.ScreenConfig 29
ELCsApi.ServerInfo 30
ELCsApi.ValidationPointResult 30
ELCsApi.ValidationResult 31

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ELCsApi.DeviceConfig
: Device configuration . . . 19

ELCsApi.DeviceGeometry
: Geometric position of the device related to the active monitor . . . 20

ELCsApi
: Main class for communication with the EyeLogic server . . . 20

ELException
: EyeLogic Exception class. API functions may throw this exception, catch it for error handling . . . 26

GazeSample
: EyeLogic GazeSample . . . 27

Point2d
: 2D point . . . 28

Point3d
: 3D point . . . 29

ELCsApi.ScreenConfig
: Screen configuration . . . 29

ELCsApi.ServerInfo
: Connection information for an EyeLogic server . . . 30

ELCsApi.ValidationPointResult
: ValidationPointResult holds the results of the validation (total deviation between true point position and calculated POR of the left and right eye POR in [px] and [deg]) of the validation point at position (validationPointPxX, validationPointPxY) [px] . . . 30

ELCsApi.ValidationResult
: ValidationResult contains one ValidationPointResult struct per validation stimulus point of the performed valdation. ValidationPointResult data fields may be ELCsLib.InvalidValue 31

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

ELCsApi.cs
The file contains the C# definitions which are neccessary to control the EyeLogic software from an API client 33

Chapter 6

Namespace Documentation

6.1 eyelogic Namespace Reference

namespace for C# API calls

Classes

- class ELCsApi
  main class for communication with the EyeLogic server
- class ELException
  EyeLogic Exception class. API functions may throw this exception, catch it for error handling.
- class GazeSample
  EyeLogic GazeSample.
- class Point2d
  2D point
- class Point3d
  3D point

Enumerations

- enum EventType {
  SCREEN_CHANGED, CONNECTION_CLOSED, DEVICE_CONNECTED, DEVICE_DISCONNECTED,
  TRACKING_STOPPED }
  EyeLogic events.

6.1.1 Detailed Description

namespace for C# API calls

6.1.2 Enumeration Type Documentation

6.1.2.1 EventType

```
enum EventType  [strong]
```

EyeLogic events.

Enumerator

SCREEN_CHANGED	a new screen has been set as active
CONNECTION_CLOSED	connection to EyeLogic Server has closed
DEVICE_CONNECTED	a new device has connected
DEVICE_DISCONNECTED	device disconnected
TRACKING_STOPPED	tracking has stopped

Chapter 7

Class Documentation

7.1 ELCsApi.DeviceConfig Class Reference

device configuration

Public Member Functions

- string formatDeviceSerial ()

  get device serial number as formatted string

Public Attributes

- ulong deviceSerial

  serial number of the device as unsigned 64-bit int for a verbose format, call
- List< int > frameRates

  list of available framerates [Hz]
- List< int > calibrationMethods

  list of available calibration methods [number of calibration points]

7.1.1 Detailed Description

device configuration

7.1.2 Member Data Documentation

7.1.2.1 deviceSerial

```
ulong deviceSerial
```

serial number of the device as unsigned 64-bit int for a verbose format, call

See also

formatDeviceSerial

7.2 ELCsApi.DeviceGeometry Class Reference

Geometric position of the device related to the active monitor.

Public Attributes

- double mmBelowScreen

  vertical distance between the lowest pixel on the display and the upper edge of the eye tracker
- double mmTrackerInFrontOfScreen

  horizontal distance between the front of the screen and the front edge of the eye tracker

7.2.1 Detailed Description

Geometric position of the device related to the active monitor.

7.3 ELCsApi Class Reference

main class for communication with the EyeLogic server

Classes

- class DeviceConfig

  device configuration
- class DeviceGeometry

  Geometric position of the device related to the active monitor.
- class ScreenConfig

  screen configuration
- class ServerInfo

  connection information for an EyeLogic server
- class ValidationPointResult

  ValidationPointResult holds the results of the validation (total deviation between true point position and calculated POR of the left and right eye POR in [px] and [deg]) of the validation point at position (validationPointPxX, validation↩ PointPxY) [px].
- class ValidationResult

  ValidationResult contains one ValidationPointResult struct per validation stimulus point of the performed valdation. ValidationPointResult data fields may be ELCsLib.InvalidValue

Public Member Functions

- delegate void ELEvent (EventType id)

  Event type.
- delegate void ELGazeSample (GazeSample sample)

  GazeSample event type.
- delegate void ELEyeImage (Bitmap eyeImage)

  EyeImage event type.
- ELCsApi (string clientName)

  constructor
- void destroy ()

  destroys the ELCsApi. Call this once before shutting down.
- void connect ()

  initialize connection to the server (method is blocking until connection established). The connection is only established for a local server (running on this machine). For connections to a remote server,
- void connectRemote (ServerInfo server)

  initialize connection to a remote server (method is blocking until connection established)
- ServerInfo[ ] requestServerList (int blockingDurationMS, int maxNumServer)

  Ping all running EyeLogic servers in the local network and wait some time for their response.
- void disconnect ()

  closes connection to the server
- bool isConnected ()

  whether a connection to the server is established
- ScreenConfig getActiveScreen ()

  obtain configuration of active screen
- ScreenConfig[ ] getAvailableScreens ()

  Get a list of screens connected to the local machine. If there are more screens than 'numScreenConfigs' found, then only the first 'numScreenConfigs' ones are filled.
- void setActiveScreen (string screenID, DeviceGeometry deviceGeometry)

  Make a screen connected to this machine to the active screen.
- DeviceConfig getDeviceConfig ()

  obtain configuration of active device
- void **streamEyeImages** (bool enable)
- void requestTracking (int frameRateModeInd)

  request tracking
- void unrequestTracking ()

  unrequest tracking
- void calibrate (int calibrationModeInd)

  perform calibration (method is blocking until calibration finished)
- ValidationResult validate ()

  perform validation (method is blocking until validation finished) - calibration must be performed prior

Static Public Attributes

- static double InvalidValue = Double.MinValue

  marker for an invalid double value

Events

- ELEvent OnEvent = delegate { }

  Event, add your event handler to become notified on new events.
- ELGazeSample OnGazeSample = delegate { }

  GazeSample event, add your event handler to become notified on new gaze samples.
- ELEyeImage OnEyeImage = delegate { }

  EyeImage event, add your event handler to become notified on new eye images.

7.3.1 Detailed Description

main class for communication with the EyeLogic server

7.3.2 Constructor & Destructor Documentation

7.3.2.1 ELCsApi()

```
ELCsApi (
            string clientName )  [inline]
```

constructor

Parameters

clientName	string identifier of the client (shown in the server tool window), may be null

7.3.3 Member Function Documentation

7.3.3.1 calibrate()

```
void calibrate (
            int calibrationModeInd )  [inline]
```

perform calibration (method is blocking until calibration finished)

Parameters

calibrationModeInd	index of the requested calibration method in DeviceConfig.calibrationMethods

7.3.3.2 connect()

```
void connect ( )  [inline]
```

initialize connection to the server (method is blocking until connection established). The connection is only established for a local server (running on this machine). For connections to a remote server,

See also

connectRemote( ).

7.3.3.3 connectRemote()

```
void connectRemote (
            ServerInfo server )  [inline]
```

initialize connection to a remote server (method is blocking until connection established)

Parameters

server	Server to connect to

See also

acquireServerList( ) to obtain IP address and port of a remote server

7.3.3.4 getActiveScreen()

```
ScreenConfig getActiveScreen ( )  [inline]
```

obtain configuration of active screen

Returns

ScreenConfig returns the config of the active screen

7.3.3.5 getAvailableScreens()

```
ScreenConfig [ ] getAvailableScreens ( )  [inline]
```

Get a list of screens connected to the local machine. If there are more screens than 'numScreenConfigs' found, then only the first 'numScreenConfigs' ones are filled.

Parameters

screenConfig	pre-allocated array, will be filled with screen configurations
numScreenConfigs	number of entries of screenConfig

Returns

number of filled screen configurations. will be <= numScreenConfigs

7.3.3.6 getDeviceConfig()

```
DeviceConfig getDeviceConfig ( )  [inline]
```

obtain configuration of active device

Returns

DeviceConfig returns the config of the connected device

7.3.3.7 requestServerList()

```
ServerInfo [ ] requestServerList (
            int blockingDurationMS,
            int maxNumServer )  [inline]
```

Ping all running EyeLogic servers in the local network and wait some time for their response.

Parameters

blockingDurationMS	waiting duration in milliseconds. Method returns at the latest after this time.
maxNumServer	Maximum number of server to be searched. Method returns immediately when that number of server is found.

Returns

List of all responding EyeLogic servers

7.3.3.8 requestTracking()

```
void requestTracking (
            int frameRateModeInd )  [inline]
```

request tracking

If tracking is not yet ongoing, tracking is started in the device. If tracking is already running (e.g. started from another client) with the same frame-rate as requested, all gaze samples are reported to this client as well.

Parameters

frameRateModeInd	index of the requested frame rate mode in DeviceConfig.frameRates

7.3.3.9 setActiveScreen()

```
void setActiveScreen (
            string screenID,
            DeviceGeometry deviceGeometry )  [inline]
```

Make a screen connected to this machine to the active screen.

Recording is from now on performed on the new active screen. Remember to perform a calibration on the new screen, otherwise it remains in an uncalibrated state.

Parameters

screenID	ID of the new active screen on *this* machine (even works if the connection to the server is remote). If null, the primary screen of this machine is set as active.
deviceGeometry	Geometry of the device which is mounted to the screen.

Returns

success/error code

7.3.3.10 unrequestTracking()

```
void unrequestTracking ( )  [inline]
```

unrequest tracking

Note that the tracking device may continue if other processes still request tracking. Check the EyeLogic server window to observe the actual state.

7.3.3.11 validate()

```
ValidationResult validate ( )  [inline]
```

perform validation (method is blocking until validation finished) - calibration must be performed prior

Returns

ValidationResult

7.4 ELException Class Reference

EyeLogic Exception class. API functions may throw this exception, catch it for error handling.

Public Types

- enum ErrorType {
  UNKNOWN_ERROR, ALREADY_INITED, NOT_INITED, VERSION_MISMATCH,
  CONNECTION_FAILED, NOT_CONNECTED, DEVICE_MISSING, INVALID_FRAMERATE_MODE,
  ALREADY_TRACKING, NOT_TRACKING, INVALID_CALIBRATION_MODE, ALREADY_CALIBRATING_OR_VALIDATING,
  NOT_CALIBRATED, SCREEN_NOT_FOUND, SCREEN_FAILURE }

  Error type.

Public Member Functions

- ELException ()

  Default constructor.
- ELException (ErrorType error, string message)

  Constructor with error type and message string.

Public Attributes

- ErrorType Error

  error type

7.4.1 Detailed Description

EyeLogic Exception class. API functions may throw this exception, catch it for error handling.

7.4.2 Member Enumeration Documentation

7.4.2.1 ErrorType

```
enum ErrorType  [strong]
```

Error type.

Enumerator

UNKNOWN_ERROR	not specified
ALREADY_INITED	cannot initialize library: was already initialized before
NOT_INITED	library not correctly initialized
VERSION_MISMATCH	connection failed: API is build on a newer version than the server. Update the EyeLogicServer to the newest version.

Enumerator

CONNECTION_FAILED	connection failed: the server can not be found or is not responding
NOT_CONNECTED	not connected to the server
DEVICE_MISSING	cannot start tracking: no device found
INVALID_FRAMERATE_MODE	cannot start tracking: framerate mode is invalid or not supported
ALREADY_TRACKING	tracking already ongoing, but frame rate mode is different
NOT_TRACKING	cannot calibrate: no device found or tracking not started
INVALID_CALIBRATION_MODE	cannot start calibration: calibration mode is invalid or not supported
ALREADY_CALIBRATING_OR_VALIDATING	cannot start calibration or validation: a calibration or validation is already in progress
NOT_CALIBRATED	cannot start validation: device must be calibrated
SCREEN_NOT_FOUND	cannot set active screen: given screen was not found
SCREEN_FAILURE	cannot set active screen

7.5 GazeSample Class Reference

EyeLogic GazeSample.

Public Attributes

- long timestampMicroSec
  timepoint when data was acquired in microseconds after EPOCH
- int index
  increasing GazeSample index
- Point2d porRaw
  binocular point of regard on the stimulus plane, check porRaw.x != ELCsAPI.InvalidValue before using it
- Point2d porFiltered
  filtered binocular point of regard on the stimulus plane, check porFiltered.x != ELCsAPI.InvalidValue before using it
- Point2d porLeft
  monocular point of regard of the left eye, check porLeft.x != ELCsAPI.InvalidValue before using it
- Point3d eyePositionLeft
  position of the left eye in device coordinates, unit is mm
- double pupilRadiusLeft
  radius of the left pupil in mm or ELCsAPI.InvalidValue if eye was not found
- Point2d porRight
  monocular point of regard of the right eye, check porRight.x != ELCsAPI.InvalidValue before using it
- Point3d eyePositionRight
  position of the right eye in device coordinates, unit is mm
- double pupilRadiusRight
  radius of the right pupil in mm or ELCsAPI.InvalidValue if eye was not found

7.5.1 Detailed Description

EyeLogic GazeSample.

7.5.2 Member Data Documentation

7.5.2.1 eyePositionLeft

```
Point3d eyePositionLeft
```

position of the left eye in device coordinates, unit is mm

- (0, 0, 0) is in the center of the device
- x-coordinate: positive towards the right side of the screen
- y-coordinate: positive towards the top of the screen
- z-coordinate: distance in front of the screen

check eyePositionLeft.x != ELCsAPI.InvalidValue before using it

7.5.2.2 eyePositionRight

```
Point3d eyePositionRight
```

position of the right eye in device coordinates, unit is mm

- (0, 0, 0) is in the center of the device
- x-coordinate: positive towards the right side of the screen
- y-coordinate: positive towards the top of the screen
- z-coordinate: distance in front of the screen

check eyePositionRight.x != ELCsAPI.InvalidValue before using it

7.6 Point2d Class Reference

2D point

Public Member Functions

- Point2d (double x, double y)

  Constructor.

Public Attributes

- double x

  x coordinate of the point
- double y

  y coordinate of the point

7.6.1 Detailed Description

2D point

7.7 Point3d Class Reference

3D point

Public Member Functions

- Point3d (double x, double y, double z)

  Constructor.

Public Attributes

- double x

  x coordinate of the point
- double y

  y coordinate of the point
- double z

  z coordinate of the point

7.7.1 Detailed Description

3D point

7.8 ELCsApi.ScreenConfig Class Reference

screen configuration

Public Attributes

- bool localMachine

  whether the screen is connected to the this machine
- string id

  identifier name of the screen
- string name

  descriptive name of the screen
- int resolutionX

  screen resolution [px]
- int resolutionY

  screen resolution [px]
- double physicalSizeX_mm

  physical dimension of the screen [mm]
- double physicalSizeY_mm

  physical dimension of the screen [mm]

7.8.1 Detailed Description

screen configuration

7.9 ELCsApi.ServerInfo Class Reference

connection information for an EyeLogic server

Public Attributes

- string ip

  IP address of server as 0-terminated string.
- ushort port

  port of server

7.9.1 Detailed Description

connection information for an EyeLogic server

7.10 ELCsApi.ValidationPointResult Class Reference

ValidationPointResult holds the results of the validation (total deviation between true point position and calculated POR of the left and right eye POR in [px] and [deg]) of the validation point at position (validationPointPxX, validationPointPxY) [px].

Public Attributes

- double validationPointPxX

  x-coordinate of stimulus point position
- double validationPointPxY

  y-coordinate of stimulus point position
- double meanDeviationLeftPx

  InvalidValue or mean deviation between left eye POR and stimulus position in [px] in the stimulus plane.
- double meanDeviationLeftDeg

  InvalidValue or mean deviation of left eye gaze direction in [deg] in the 3-D world system.
- double meanDeviationRightPx

  InvalidValue or mean deviation between right eye POR and stimulus position in [px] in the stimulus plane.
- double meanDeviationRightDeg

  InvalidValue or mean deviation of right eye gaze direction in [deg] in the 3-D world system.

7.10.1 Detailed Description

ValidationPointResult holds the results of the validation (total deviation between true point position and calculated POR of the left and right eye POR in [px] and [deg]) of the validation point at position (validationPointPxX, validationPointPxY) [px].

The stimulus point position and deviation [px] are given in the 2D stimulus coordinate system originating in the top left corner of the stimulus.

The deviation [deg] corresponds to the total angular deviation between the measured gaze direction from the ground truth gaze direction as determined according to the measured eye position.

Note: meanDeviation∗ data fields may be ELCsApi.InvalidValue. meanDeviationLeftDeg/-Px and meanDeviation↩ RightDeg-/Px are always either both valid or both ELCsApi.InvalidValue.

7.11 ELCsApi.ValidationResult Class Reference

ValidationResult contains one ValidationPointResult struct per validation stimulus point of the performed valdation. ValidationPointResult data fields may be ELCsLib.InvalidValue

Public Attributes

- List< ValidationPointResult > **pointsData**

7.11.1 Detailed Description

ValidationResult contains one ValidationPointResult struct per validation stimulus point of the performed valdation. ValidationPointResult data fields may be ELCsLib.InvalidValue

Chapter 8

File Documentation

8.1 ELCsApi.cs File Reference

The file contains the C# definitions which are neccessary to control the EyeLogic software from an API client.

Classes

- class Point2d

  2D point
- class Point3d

  3D point
- class GazeSample

  EyeLogic GazeSample.
- class ELException

  EyeLogic Exception class. API functions may throw this exception, catch it for error handling.
- class ELCsApi

  main class for communication with the EyeLogic server
- class ELCsApi.ServerInfo

  connection information for an EyeLogic server
- class ELCsApi.ScreenConfig

  screen configuration
- class ELCsApi.DeviceGeometry

  Geometric position of the device related to the active monitor.
- class ELCsApi.DeviceConfig

  device configuration
- class ELCsApi.ValidationPointResult

  ValidationPointResult holds the results of the validation (total deviation between true point position and calculated POR of the left and right eye POR in [px] and [deg]) of the validation point at position (validationPointPxX, validation↩ PointPxY) [px].
- class ELCsApi.ValidationResult

  ValidationResult contains one ValidationPointResult struct per validation stimulus point of the performed valdation. ValidationPointResult data fields may be ELCsLib.InvalidValue

Namespaces

- namespace eyelogic

  namespace for C# API calls

Enumerations

- enum EventType {
  SCREEN_CHANGED, CONNECTION_CLOSED, DEVICE_CONNECTED, DEVICE_DISCONNECTED,
  TRACKING_STOPPED }

  EyeLogic events.

8.1.1 Detailed Description

The file contains the C# definitions which are neccessary to control the EyeLogic software from an API client.

Index

ALREADY_CALIBRATING_OR_VALIDATING
    ELException, 27
ALREADY_INITED
    ELException, 26
ALREADY_TRACKING
    ELException, 27

calibrate
    ELCsApi, 22
connect
    ELCsApi, 22
CONNECTION_CLOSED
    eyelogic, 18
CONNECTION_FAILED
    ELException, 27
connectRemote
    ELCsApi, 23

DEVICE_CONNECTED
    eyelogic, 18
DEVICE_DISCONNECTED
    eyelogic, 18
DEVICE_MISSING
    ELException, 27
deviceSerial
    ELCsApi.DeviceConfig, 19

ELCsApi, 20
    calibrate, 22
    connect, 22
    connectRemote, 23
    ELCsApi, 22
    getActiveScreen, 23
    getAvailableScreens, 23
    getDeviceConfig, 24
    requestServerList, 24
    requestTracking, 24
    setActiveScreen, 25
    unrequestTracking, 25
    validate, 25
ELCsApi.cs, 33
ELCsApi.DeviceConfig, 19
    deviceSerial, 19
ELCsApi.DeviceGeometry, 20
ELCsApi.ScreenConfig, 29
ELCsApi.ServerInfo, 30
ELCsApi.ValidationPointResult, 30
ELCsApi.ValidationResult, 31
ELException, 26
    ALREADY_CALIBRATING_OR_VALIDATING, 27
    ALREADY_INITED, 26
    ALREADY_TRACKING, 27
    CONNECTION_FAILED, 27
    DEVICE_MISSING, 27
    ErrorType, 26
    INVALID_CALIBRATION_MODE, 27
    INVALID_FRAMERATE_MODE, 27
    NOT_CALIBRATED, 27
    NOT_CONNECTED, 27
    NOT_INITED, 26
    NOT_TRACKING, 27
    SCREEN_FAILURE, 27
    SCREEN_NOT_FOUND, 27
    UNKNOWN_ERROR, 26
    VERSION_MISMATCH, 26
ErrorType
    ELException, 26
EventType
    eyelogic, 17
eyelogic, 17
    CONNECTION_CLOSED, 18
    DEVICE_CONNECTED, 18
    DEVICE_DISCONNECTED, 18
    EventType, 17
    SCREEN_CHANGED, 18
    TRACKING_STOPPED, 18
eyePositionLeft
    GazeSample, 28
eyePositionRight
    GazeSample, 28

GazeSample, 27
    eyePositionLeft, 28
    eyePositionRight, 28
getActiveScreen
    ELCsApi, 23
getAvailableScreens
    ELCsApi, 23
getDeviceConfig
    ELCsApi, 24

INVALID_CALIBRATION_MODE
    ELException, 27
INVALID_FRAMERATE_MODE
    ELException, 27

NOT_CALIBRATED
    ELException, 27
NOT_CONNECTED
    ELException, 27

NOT_INITED
ELException, 26
NOT_TRACKING
ELException, 27

Point2d, 28
Point3d, 29

requestServerList
ELCsApi, 24
requestTracking
ELCsApi, 24

SCREEN_CHANGED
eyelogic, 18
SCREEN_FAILURE
ELException, 27
SCREEN_NOT_FOUND
ELException, 27
setActiveScreen
ELCsApi, 25

TRACKING_STOPPED
eyelogic, 18

UNKNOWN_ERROR
ELException, 26
unrequestTracking
ELCsApi, 25

validate
ELCsApi, 25
VERSION_MISMATCH
ELException, 26