iView X™ SDK

v3.6

May 2014

Contents

Chapter 1

iView X™ API Documentation

1.1 Introduction

Welcome to the iView X™ SDK Guide v.3.6!

About iView X™ SDK

The iView X™ Software Development Kit ("SDK") provides an Application Interface ("API") for communication between your software application and iView X™, allowing you to create full-featured eye tracking applications that take advantage of the powerful features offered by SensoMotoric Instruments ("SMI") eye tracking devices and the iView X™ platform. Specifically, the SDK was designed for SMI customers who wish to add eye tracking into their own custom applications. Using the functions provided in the SDK you can control SMI eye tracking devices and retrieve eye tracking data online.

About the Guide

The SDK Guide provides a practical introduction to developing applications using the SDK and documentation about major SDK features. It includes instructions for setting up your SDK environment and a function reference, which outlines each available function as well as the supported devices. Additionally, the manual gives a brief overview on the included examples for each major platform.

What's New?

In addition to this document, the SDK includes release notes which may be found in the SMI\iView X SDK\docs directory. In the release notes you can find a complete list of the improvements and bug fixes, helping you get the most from each release.

API Layer Overview:

The figure below shows hard- and software components of the eye tracking system. A customer application connects via the API with the eyetracking server.

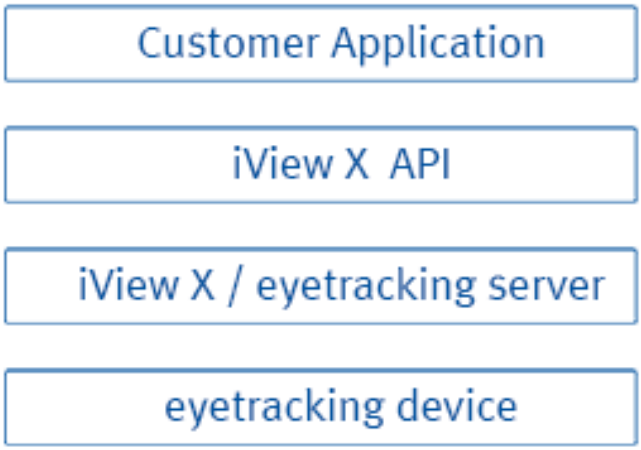

Figure 1.1: API Layers

Customer Application: Custom software using the API to interact with the eye tracking device. You can develop your own application or integrate 3rd party applications into your eye tracking system.

iView X™ API: Programmable interface to provide access to eye tracking device. iView X™ API is part of the iView X™ SDK. A common C Interface is provided, but you can use any programming language to build your own eye tracking application. Please check Supported Programming and Scripting Languages for details.

iView X™ / eyetracking server: eyetracking server application which collects the data from the eye tracking device and provides the data via the iView X™ API. Note: depending on your system, the eyetracking server functionality is provided by different binaries. For Hi-Speed, RED, etc., this is iView X. For RED-m and RED-OEM, this is the eyetracking_server. To improve readabilty the term "eyetracking server" is used as a generic name for this software component.

eye tracking device: eye tracking device by SMI. Please check Supported Eye Tracking Devices for a list of supported devices.

System Requirements

Supported Eye Tracking Devices

The following SMI Eye Tracking Devices are supported in this release:

Supported Eye Tracking Device	Frame Rate [Hz]
iView X™ RED 4 (Firewire)	50 / 60
RED (USB)	60 / 120
RED250	60 / 120 / 250

RED500	60 / 120 / 250 / 500
RED-m	60 / 120
RED-OEM	depends
iView X™ HED	50 / 200
iView X™ HED HT	50 / 200
iView X™ Hi-Speed	240 (mono)
iView X™ Hi-Speed	350 (mono / bin)
iView X™ Hi-Speed	500 (mono / bin)
iView X™ Hi-Speed	1250 (mono)
iView X™ Hi-Speed Primate	500 / 1250 (mono / bin)
iView X™ MRI LR	50
iView X™ MEG	50 / 250

Please note that ETG devices are not supported with this version of iView X™ SDK. Please visit http://www.smivision.com/en/gaze-and-eye-tracking-systems/support/software-download.html for more information.

Supported Programming and Scripting Languages

The iView X™ SDK can be used with most programming and scripting languages that are capable of importing dynamic link libraries (DLLs). These include, but are not limited to,

- C++
- C#
- MATLAB®
- E-Prime
- Python
- NBS Presentation

Supported Operating Systems

This SDK installer contains Windows 32-bit and 64-bit binaries. The SDK application files are installed into

```
C:\Program Files (x86)
```

for Windows 64-bit OS and

```
C:\Program Files
```

for Windows 32-bit OS. The iView X™ SDK for is designed to run on the following operating systems:

Operating System	Notes
Windows XP	Supported
Windows Vista 32/64 bit	Supported
Windows 7 32/64 bit	Supported
Windows 8 32/64 bit	Supported
Linux	Planned
Mac OS X	Planned

Getting Started

In the following sections you will learn how to set up your SDK environment, about the various function available in the SDK, and how to create your first basic eye tracking application based on the provided examples.

Downloading

You can download the latest recommended release of the SDK from the SMI Software Downloads page: `http://www.smivision.com/en/gaze-and-eye-tracking-systems/support/software-download.html`.

Running the Installer

Note: The SDK has to be installed on the same computer as your software application. If you run your eye tracking studies in a single-PC setup, this will be the same computer as your iView X™ software.

After you have downloaded the SDK installer package, execute `SMI iView X SDK.exe` to begin the installation. When the files have been unpacked, the SDK License Agreement will appear — it contains important information about the terms under which we supply the SDK. Agree to it if you would like to proceed with the installation. If you had a previous installation it will first be removed before the new version of the SDK is installed on your computer. Please wait for the installation to complete. The installation process may take a few minutes. Note: The SDK is already included in some RED-OEM Developer Editions, in which case there is no need to install iView X™ SDK.

Running the Demo

Once you have completed installation of the SDK, you are ready to begin developing applications. To learn more about the capabilities of the iView X™ SDK you may start with a demo application that shows most of the features the API provides.

To start the demo application, please

1. Connect your eye tracking device and start the eye tracking software. Depending on your device type, this is usually iView X™ or, iView RED-m or the eyetracking server.
2. Run the `csdemo.exe` application in

```
C:\Program Files\SMI\iView X SDK\Examples\VS C#\Demo Application\
```

or

```
C:\Program Files (x86)\SMI\iView X SDK\Examples\VS C#\Demo Application\
```

csDemo can be used with any SMI eye tracking device that is supported by the iView X™ SDK. Press **Connect** to establish the connection between csDemo and the eyetracking server.

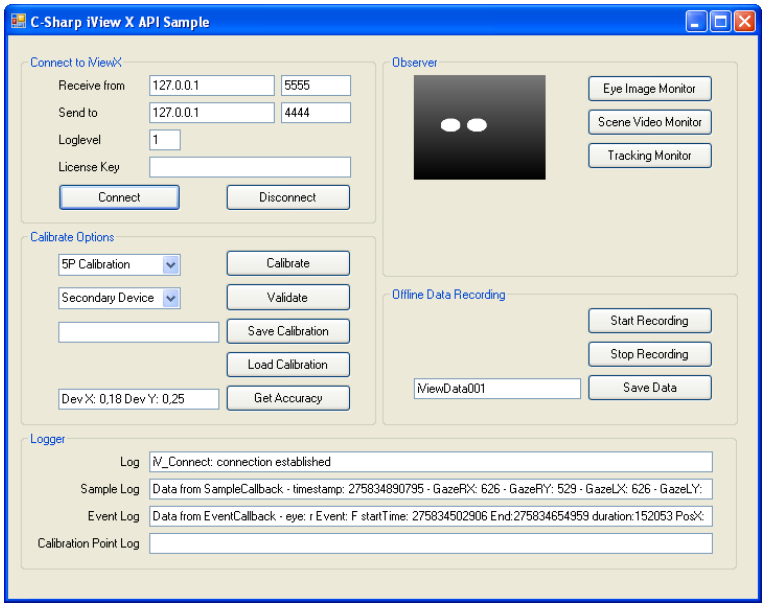

Figure 1.2: Screenshot csDemo

Eye data transmission will start immediately. If a connection has been established, gaze data will be streamed automatically and will be shown in the **Sample Log** text box. If not, please check the connection settings in csDemo and the eye tracking software.

Troubleshooting:

Please Note: In order to exchange data between the eyetracking server and your software application using the SDK, an ethernet connection has to be established. This applies even when running the eyetracking server and your software application on the same PC. If you are unfamiliar with this process, please consult the relevant documentation (e.g. the eyetracking server user manual) on how to establish an ethernet connection between different computers. Please adjust the IP address and port settings in eyetracking server and your application accordingly.

To establish a connection to eyetracking server please set the according IP addresses in the **Connect to iView X** sections of csDemo. If you run csDemo and eyetracking server on the same PC, the **Received from** and **Send to** IP addresses and ports will likely be (127.0.0.1; 5555) and (127.0.0.1; 4444), respectively. Please note that the **Receive from** IP address and Port will be the same as the **Send to** IP address and port set in

- iView X™ (Setup -> Hardware -> Communication -> Ethernet) or

- **Network Settings...** entry from tray menu. You should be sure to verify this, otherwise iView X™ and the example program will not be able to communicate.

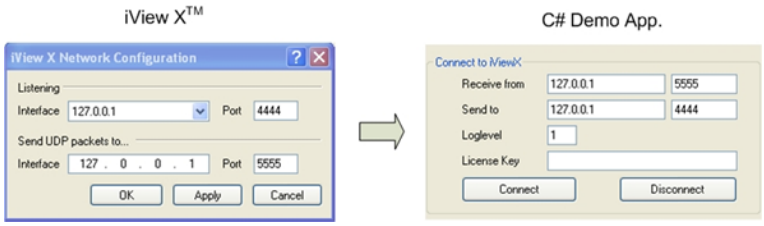

Figure 1.3: Network Settings

After configuring the IP addresses and ports, click the **Connect** button and check again if eye data is available.

For further troubleshooting or to learn more about configuring the connection, please take a look into the section Single PC and Dual PC Setup.

The source code for this demo application is available here:

```
C:\Program Files\SMI\iView X SDK\Examples\VS C#\Demo Project\csdemo
```

Please have a look into the section C# to learn more about using C# and Microsoft Visual Studio to access the iView X™ SDK.

1.2 Developing Applications

The SDK includes sample code and applications for any major environment. Please browse through them in the "Examples" folder. If you want to develop your own eye tracking application we recommend copying the example code into your development environment and use it as a starting point for your own development. They highlight many of the features and capabilities of the iView X™ APIs. They are as follows:

- Remote Control Application: A simple application with the most common features for controlling an SMI eye tracker through iView X™, including establishing a connection to iView X™, performing a calibration, and receiving data from the eye tracker.
- Gaze Contingent Experiment: An example that demonstrates running a calibration session and subsequently recording eye tracking data. In this experiment gaze position data is retrieved from iView X™ in real time and displayed as an overlay on the presented bitmap image. The example illustrates several example functions and commands and is a good starting point for writing your own eye tracking application.

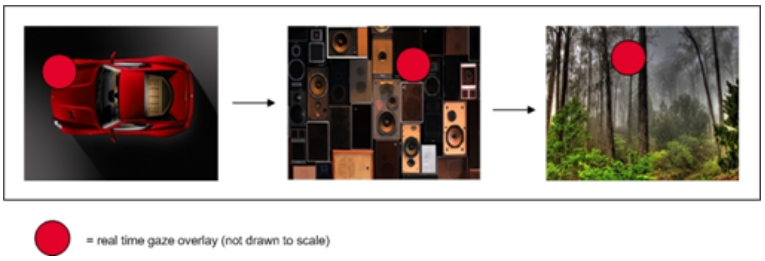

Figure 1.4: Gaze Contingent Example

- Slide Show Experiment: An example that demonstrates running a calibration session and subsequently recording eye tracking data. In this experiment a series of images are presented to a user while eye tracking data is recorded in the background.

The above examples demonstrate concepts that are fundamental to application development. All example programs described in this SDK Guide are also provided in source code form in the examples directory according to programming and scripting language type. The source code will give a more detailed insight into the possibilities of the SDK and its functions.

Tutorials

Common Workflow

This section describes the common workflow of eye tracking applications using the iView X™ API. In the subsequent sections you learn how to realize this workflow in your individual environment or programming language. We recommend to become familiar with the common workflow first and to study the details of your environment afterwards.

A common eye tracking application performs the following steps:

1. Connect to the eyetracking server
2. Run a calibration
3. Present a stimulus and gather eye tracking data
4. Close the connection

For the detailed description we use a C-like programming language syntax to explain the calls to API functions. To learn how to call API functions from your preferred programming language please refer to the corresponding section.

1. Connect to the eyetracking server

To establish a connection call iV_Connect. The parameters shown here connect to eyetracking server running on the same PC as the customer application. They should work with most systems and configurations. For details of the network setup, please see Single PC and Dual PC Setup and your eye tracking device's manual.

```
iV_Connect( "127.0.0.1", 4444, "127.0.0.1", 5555);
```

After the connections have been created, the application can be used to control the eyetracking server's behavior or to retrieve online data for further processing.

2. Run a calibration

The 2nd step in the common workflow is a calibration. A calibration is used to determine participant-specific physiological characteristics to initialize gaze mapping and to optimize eye tracking performance. Usually, a sequence of points is presented where the participant has to gaze at.

```
iV_Calibrate();
```

After the calibration has been performed the system is ready to calculate and provide gaze data.

3. Present a stimulus and gather eye tracking data

There are two ways to handle eye tracking data:

Online Data Analysis: The customer application retrieves and processes eye tracking data online. This can be used for interaction paradigms, e.g. gaze based control of user interfaces. The code snippet shows a loop where gaze data is polled and streamed to a console.

```
while (getchar() != 'q')
{
    SampleStruct sampleData;
    iV_GetSample( &sampleData);
    cout << "Left Eye's Gaze Data X: " << sampleData.leftEye.gazeX << " Y: " << sampleData.
      leftEye.gazeY << endl;
}
```

Gaze coordinates stored in `sampleData` can be used to realize gaze based interaction instead. For details about polling and other ways to retrieve online data please refer to Polling vs. Callbacks.

Offline Data Analysis: The customer application triggers eyetracking server to record eye tracking data into a file, which can be analyzed afterwards. This approach is used if data from a larger set of participants shall be analyzed or compared, or if no gaze based interaction is needed. SMI provides powerful tools for offline data analysis; please check your BeGaze manual for further information.

To start data recording, call

```
iV_StartRecording();
```

When done with recording, call

```
iV_StopRecording();
```

and finally

```
iV_SaveData( "eyedata.idf", "shortDescription", "username", 0);
```

to save the recoded data to a local file. Starting and stopping shall be synchronized with the beginning and end of your stimulus presentation.

4. Close the connection

To shutdown the connection, call

```
iV_Disconnect()
```

before closing the customer application.

E-Prime

The SDK includes several example experiments for E-Prime, two for the Standard version and two for the Professional version. The provided E-Prime sample experiments show you how to use this and other built-in E-Prime capabilities with the SDK functions. The E-Prime examples were created with version 2.0.8.22 and can be converted to newer versions.

Note: The iView X™ SDK provides a package file (.epk2) for E-Prime 2 Professional to simplify the writing of your own experiments. To make the package file available in E-Prime you have to set the package's path in the E-Prime options under "Tools -> Options... -> Packages". In "User Search Folders:" add the following path:

```
C:\[Program Files]\SMI\iView X SDK\bin
```

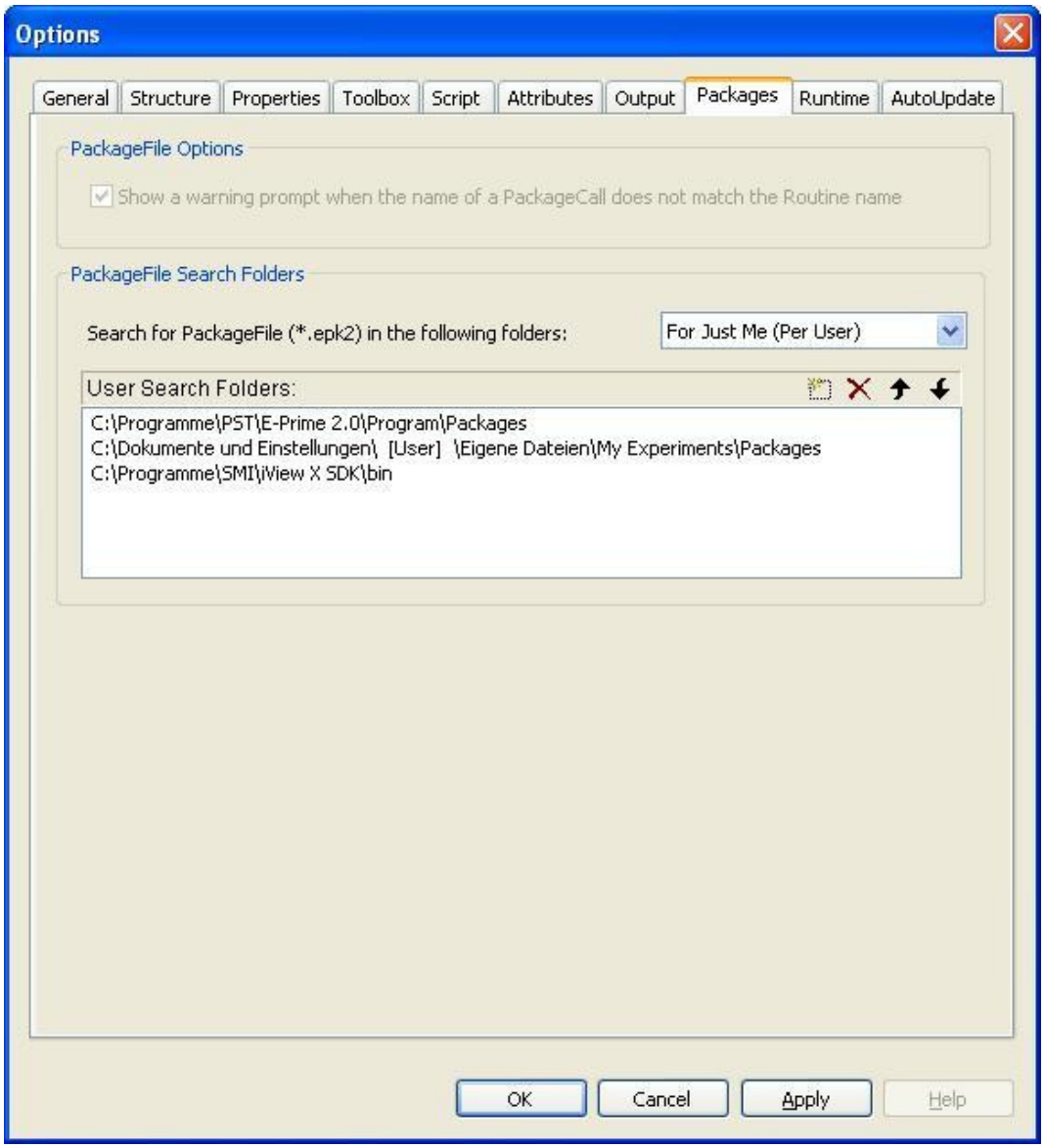

Figure 1.5: Setting up E-Prime

The following code shows how to declare structs and functions from the SDK that are needed for connecting to, getting a sample from and disconnecting from iView X™:

```
Declare Function iV_Connect Lib "iviewxapi.dll" (ByVal sendIPAddress As String, ByVal sendPort As
       Long, ByVal recvIPAddress As String, ByVal readPort As Long) As Long

Declare Function iV_Disconnect Lib "iviewxapi.dll" () As Long

Type EyeDataStruct
    gazeX As Double
    gazeY As Double
    diam As Double
    eyePosX As Double
    eyePosY As Double
    eyePosZ As Double
End Type

Type SampleStruct32
    timestamp As Double
```

```
    leftEye As EyeDataStruct
    rightEye As EyeDataStruct
    planeNumber As Long
End Type

Declare Function iV_GetSample32 Lib "iviewxapi.dll" (ByRef mySampleStruct As
      SampleStruct32) As Long
```

The following code shows how to connect to, get a gaze data sample and disconnect from iView X™:

```
Dim ret As Long

Dim sendIPAddress as String
Dim recvIPAddress as String
Dim sendPort As Long
Dim readPort As Long

sendPort = 4444
readPort = 5555
sendIPAddress = "127.0.0.1"
recvIPAddress = "127.0.0.1"

Dim sample As SampleStruct32

' connect to iView X
ret = iV_Connect (sendIPAddress, sendPort, recvIPAddress, readPort)

ret = iV_GetSample32 (sample)
```

Since E-Prime does not allow other programs to display visualizations, no images may be created by the SDK when used in combination with E-Prime. Instead, E-Prime recommends that you use their scene generation tool to automatically create scenes based on events sent by E-Prime. Additionally, due to an E-Prime limitation in handling callback functions you will need to poll for the required data. See Polling vs. Callbacks for details.

NBS Presentation

NBS Presentation allows interacting with external hardware (such as eye tracking devices) using NBS Presentation extension. This extension (iViewXAPI_NBS.dll) is provided by SMI as a part of the iView X™ SDK and needs to be registered in the operation system before it can be used in the NBS Presentaton experiments. There are two interfaces implemented in the delivered extension (EyeTracker2Impl and PCLLibrary) with individual functionality. While EyeTracker2Impl delivers the standard eye tracking functionality for NBS Presentation, like calibrating, validating, delivery of numerical data set, etc. the PCLLibrary extends this basic functionality by several functions which will be described below.

Registering the extension

Please follow the description below to register the NBS Presentation extension iViewXAPI_NBS.dll in Presentation:

1. In Presentation go to "Tools -> Extension Manager".
2. In Extension Manager press "Select Extension File".

3. In the file browser that opens select the directory where iView X™ SDK is installed. Very likely this is `C:\Program Files\SMI\iView X SDK`. From this directory select subdirectory `bin`.
4. Select file `iViewXAPI_NBS.dll` and press **Open**.
5. In Extension Manager in **Available Extensions** select **EyeTracker2Impl**.

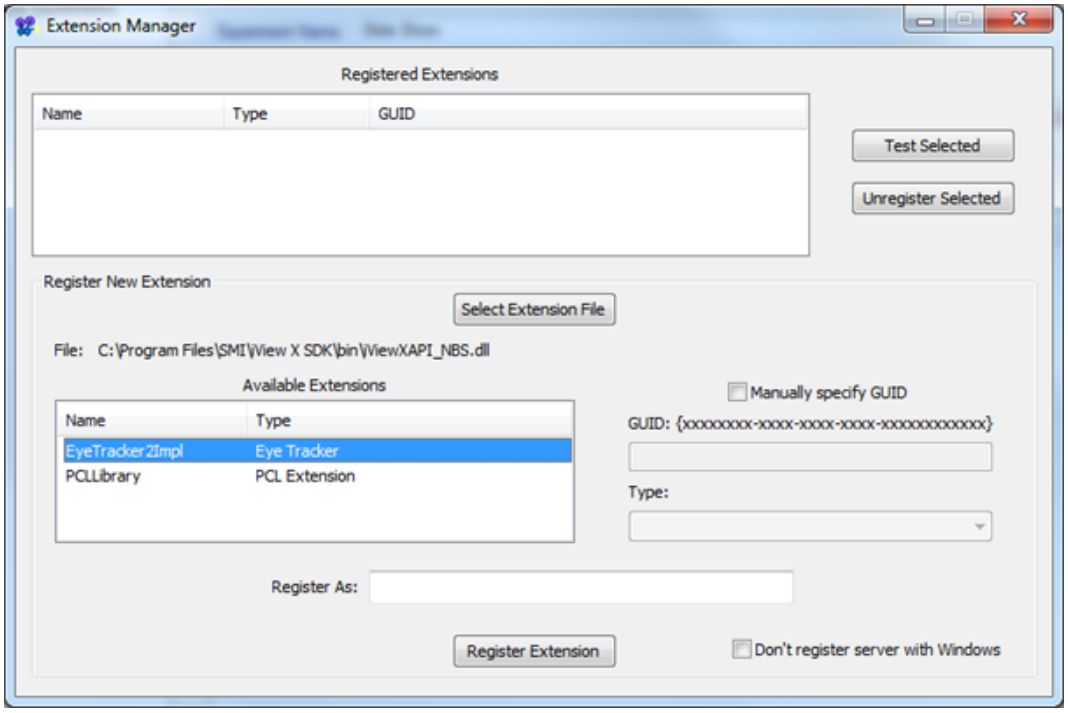

Figure 1.6: Setting up NBS Presentation, Step 5

6. In **Register As:** type "1" (or any other unique name)

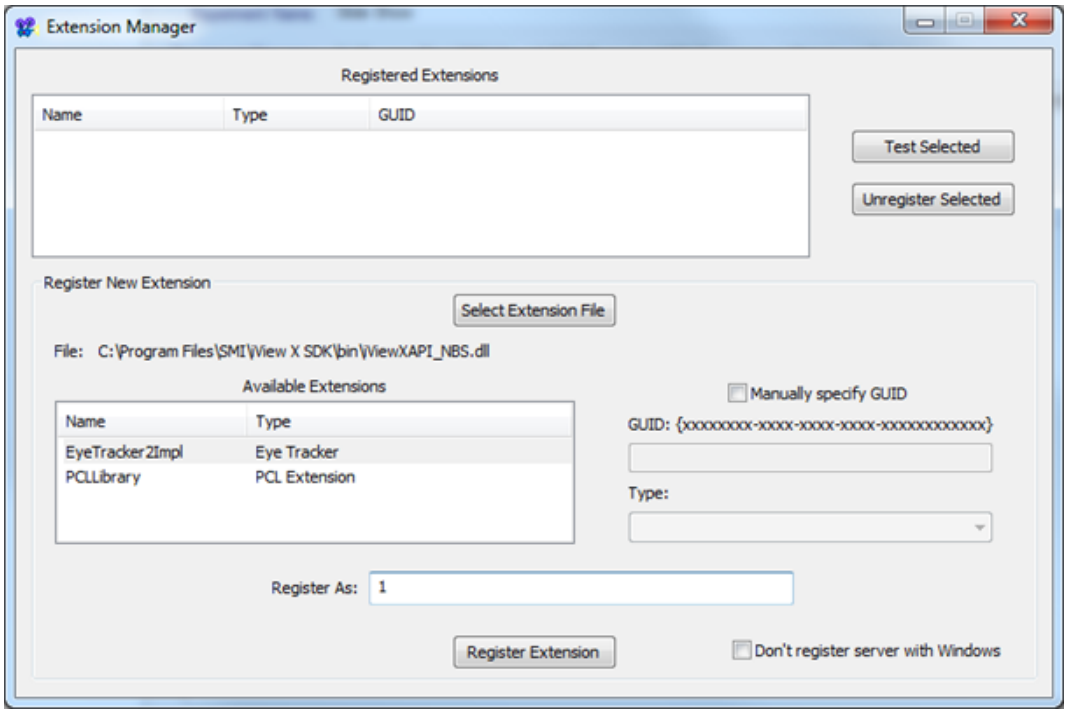

Figure 1.7: Setting up NBS Presentation, Step 6

7. Press **Register Extension**
8. Repeat steps 2 – 4.
9. In Extension Manager in **Available Extensions** select **PCLLibrary**.

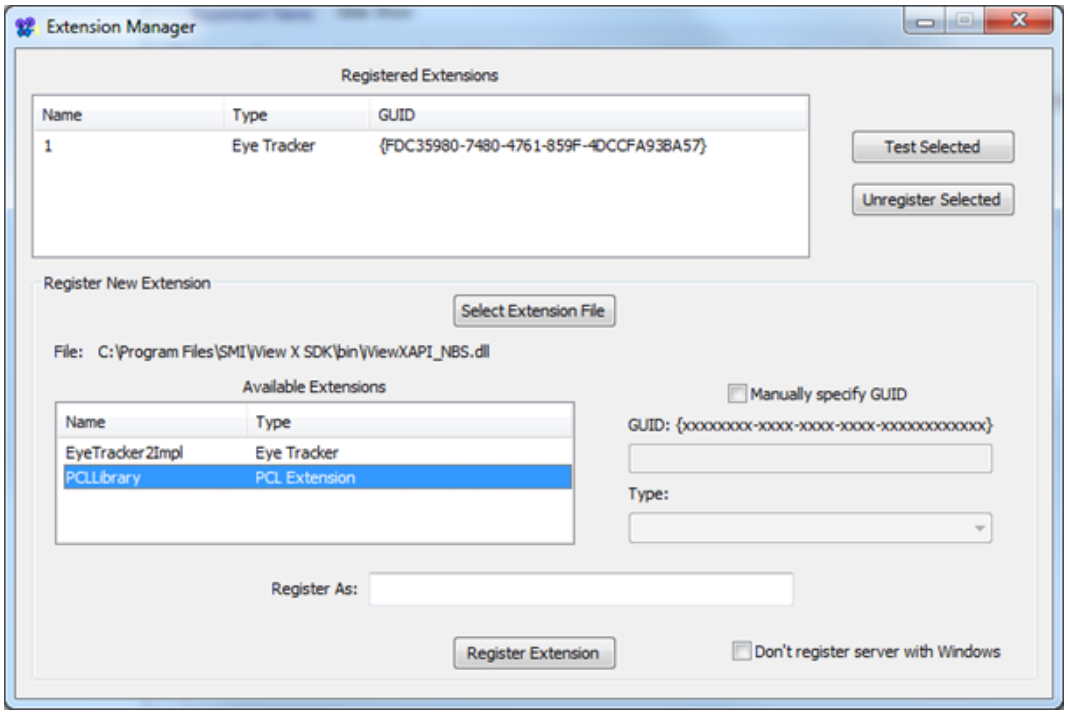

Figure 1.8: Setting up NBS Presentation, Step 9

10. In **Register As:** type "2" (or any other unique name, don't use the same name as in step 6)
11. Press **Register Extension**. Afterwards the Extension Manager should show the situation as given in the picture below:

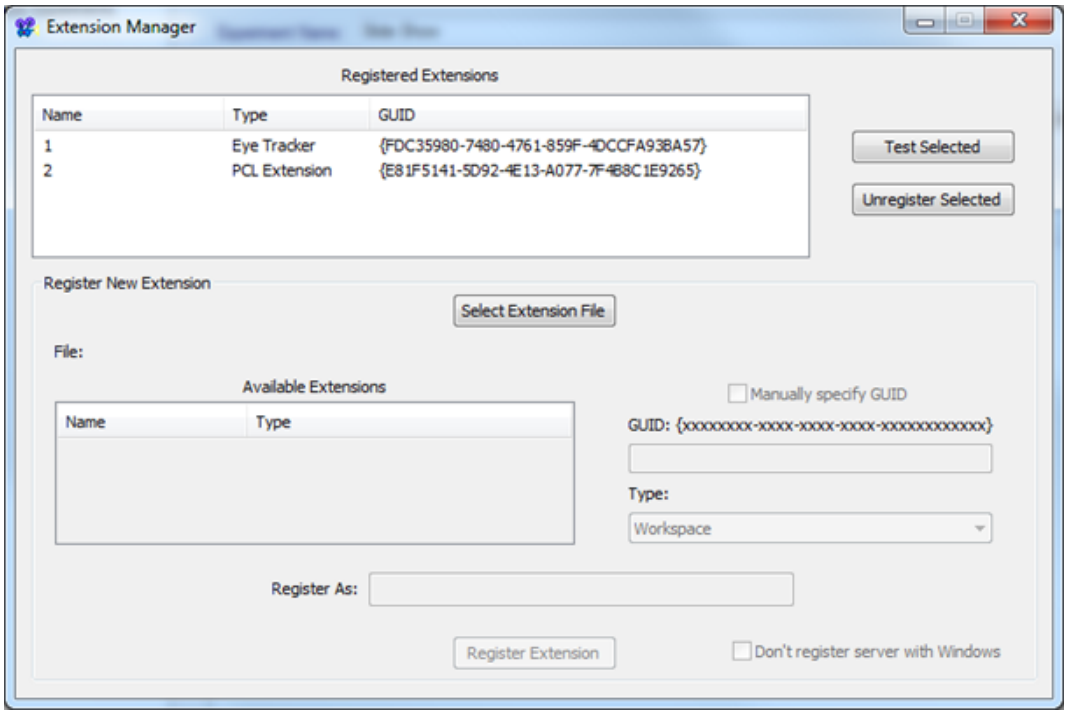

Figure 1.9: Setting up NBS Presentation, Step 11

12. Close Extension Manager. For more information on Presentation extensions and the Extension Manager please visit the NBS website http://www.neurobs.com.

Available Functions

EyeTracker2Impl Functionality

The following list shows which functions will be supported by the SMI EyeTracker2Impl interface. See the Presentation Help 'eye tracker extension' for function description.

Function	Supported
accept_point	-
buffer_position	X
calibration	X
validation (EyeTracker2CalibrationType = 2)	X
clear_buffer	X
event_count	X

get_aoi_event	-
get_blink_event	-
get_calibration_point	-
get_fixation_event	X
get_parameter	-
get_position_data	X
get_pupil_data	X
get_saccade_event	-
get_status	-
get_trigger	-
is_recording	-
last_aoi_event	-
last_blink_event	-
last_fixation_event	X
last_position_data	X
last_pupil_data	X
last_saccade_event	-
new_aoi_events	-
new_blink_events	-
new_fixation_events	X
new_position_data	X
new_pupil_data	X
new_saccade_events	-
send_command	-
send_message	X
send_string	-
send_trigger	-
set_abort_on_error	X
set_aoi_set	-
set_default_data_set	X
set_max_buffer_size	X
set_parameter	-
set_recording	X
start_calibration	-
start_tracking	X
start_data	X
stop_calibration	-

stop_tracking	X
stop_data	X
supports	X
trigger_count	-
version	X

PCLLibrary Functionality

```
# Establishes a connection to iView X (eyetracking-server).
# connect will not return until a connection has been established.
# If no connection can be established, the function will return after the defined time span of 3 seconds.
errorhandle connect(string sendIP, int sendport, string recvIP, int recvport)

# Disconnects from iView X (eyetracking-server).
# disconnect will not return until the connection has been disconnected.
# After this function has been called no other function can communicate with iView X (eyetracking-server).
errorhandle disconnect()

# Writes recorded data buffer to disc.
# The filename can include the path. If the connected eye tracking device is a HED, scene video buffer is
      written too.
# save_data will not return until the data has been saved.
errorhandle save_data()

# Returns horizontal accuracy with validated accuracy results.
# Before accuracy data is accessible the accuracy needs to be validated.
# If both eye data channels are available (binocular systems) the horizontal accuracy will be averaged.
# Invalid data will be marked as -1.
double get_accuracy_x()

# Returns vertical accuracy with validated accuracy results.
# Before accuracy data is accessible the accuracy needs to be validated.
# If both eye data channels are available (binocular systems) the vertical accuracy will be averaged.
# Invalid data will be marked as -1.
double get_accuracy_y()
```

Data handling

Due to consistency, the eye parameter handed over by functions start_data, stop_data should be equal to the parameter which will be handed over to functions like new_position_data, last_position_data, etc. and match the data which will be delivered by the SMI Eye Tracking device. If the parameters do not match the functions new_position_data might not provide any data.

Using NBS Presentation

Since the SMI NBS Presentation extension distributes two different Presentation interfaces, both will be treated as separate objects (eye_tracker and iViewXAPI::eye_tracker2) and needs to be instantiated individually in the script file. The following code shows how to create instances of both extensions and how to use them.

```
# create PCL extension instance and connect to iView X

iViewXAPI::eye_tracker2 tracker2 = new iViewXAPI::eye_tracker2( "{B7A4A7F7-7879-4C95-A3BA-6CCB355AECF6}" );
tracker2.connect(iViewX_IP, Send_Port, Local_IP, Recv_Port);

# create eye tracker extension instance, start tracking and start deliver gaze position data
```

```
eye_tracker tracker = new eye_tracker( "{FDC35980-7480-4761-859F-4DCCFA93BA57}" );
tracker.start_tracking();
tracker.start_data(dt_position);

# start calibration using a predefined calibration method, acceptation and speed setting, and start idf
       data recording

tracker.calibrate( et_calibrate_default, calibration_method, calibration_auto_accept, calibration_speed);
tracker.set_recording (true);

# get the current gaze position data

if( tracker.new_position_data() != 0 ) then
     eyepos = tracker.last_position_data();
end;

# stop idf data recording and save the recorded data to a predefined file

tracker.set_recording(false);
tracker2.save_data("presentation_data.idf", "description", "user", 1);

# disconnect from iView

tracker2.disconnect()
```

Before getting started with the NBS Presentation example experiments included with the SDK, please verify that the following settings match your current setup:

(1) Display Device

The Display Device settings, which may be found under the **Settings** tab and Video Option, should match the actual display output setting of your environment. For example, if you will be displaying your NBS Presentation experiment on your primary monitor, the Primary Display Driver and according display mode must be selected. In the example below the display mode is 1680x1050x32 (60Hz). If you are displaying your experiment on a secondary monitor, select the Secondary Display Driver option from the **Adapter** drop-down menu.

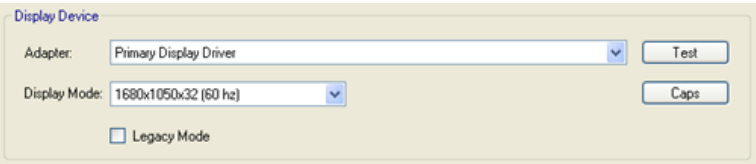

Figure 1.10: Setting up the display

(2) Screen Resolution Settings

The Screen Resolution Settings for the NBS Presentation experiments are set in the .sce file. Please make sure that the values set forth in the Display Device settings illustrated above match those in the .sce file. In the example below, the screen resolution is set to 1680x1050.

(3) Network Connection Settings

The Network Connection Settings for the NBS Presentation experiments are set in the .pcl file. Please verify that settings here match those set forth in iView X™ (Setup -> Hardware -> Communication -> Ethernet). Otherwise, the NBS Presentation experiment will not be able to communicate with iView X. As mentioned previously, if you are configuring your eye tracker to run in a dual PC setup, the connection

settings must reflect such (i.e., the actual IP addresses and ports must be listed).

```
##############################################
#                                            #
#   choose connection settings to            #
#   establish communication with iView X     #
#                                            #
##############################################

# connection settings
string iViewX_IP = "127.0.0.1";
string Local_IP = "127.0.0.1";
int Send_Port = 4444;
int Recv_Port = 5555;
```

Note: The Presentation Interface included with the SMI iTools package does NOT need to be nor should it be used in combination with the SDK to enable communication between iViewX and NBS Presentation. In fact, they are separate packages. Communication may be enabled with NBS Presentation directly through use of the SDK. While the Presentation Interface contains useful commands for start/stop recording and handling of the calibration process, we recommend that you use the SDK due to its more expansive feature set and capabilities.

C#

The SDK includes the source code for the C# example program described in Running the Demo. The C# example was created using Microsoft Visual Studio 2008.

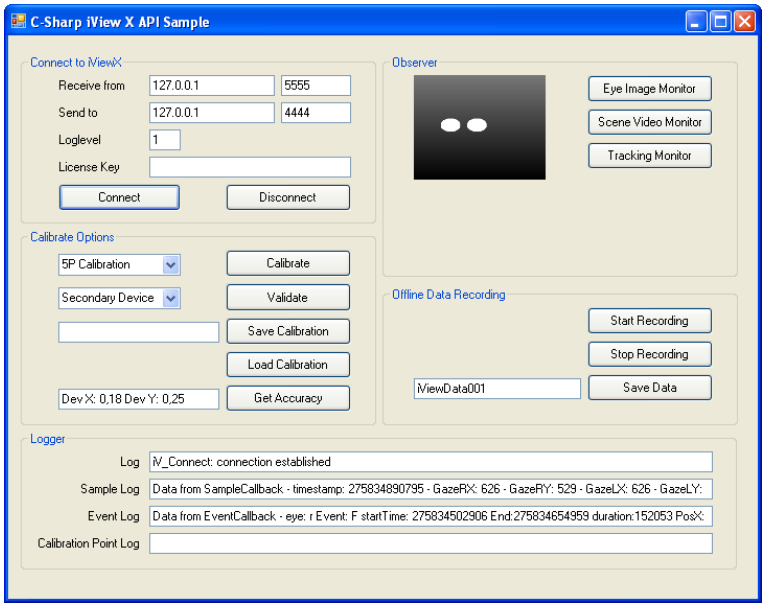

Figure 1.11: csDemo Screenshot

To establish a connection to iView X™ you must first set the according IP addresses in the Connect to iView X™ sections of the User Interface. Please read Single PC and Dual PC Setup for details.

The following code shows how to declare external functions and data structs:

```
[DllImport("iView XAPI.dll")]
public static extern Int32 iV_Connect(StringBiulder sendIP, int sendPort, StringBiulder receiveIP
      , int receivePort);
[DllImport("iView XAPI.dll")]
public static extern Int32 iV_Disconnect();
[DllImport("iView XAPI.dll")]
public static extern Int32 iV_GetSample(ref SampleStruct sampleData);

public struct EyeDataStruct
{
    public double gazeX, gazeY;     // pupil gaze [pixel]
    public double diam;         // pupil diameter [pixel/mm] (mm for RED devices)
    public double eyePositionX      // horizontal eye position relative to camera (only for
       RED)
    public double eyePositionY      // vertical eye position relative to camera (only for RED)
    public double eyePositionZ;     // distance to camera (only for RED)
};

public struct SampleStruct
{
    public Int64 timestamp;         // timestamp of current gaze data sample [microseconds]
    public EyeDataStruct leftEye;       // eye data for left eye
    public EyeDataStruct rightEye;      // eye data for left eye
    public Int32 planeNumber;           // plane number of gaze data sample (only HED HT)
};
```

Using the functions from the DLL:

```
private void connect_Click(object sender, EventArgs e)
{
    iV_Connect(new StringBuilder ("127.0.0.1"), 4444, new StringBuilder ("127.0.0.1"), 5555);
}

private void getsample_Click(object sender, EventArgs e)
{
    iV_GetSample(ref sampleData);

    logger1.Text = "Sample data - Timestamp:" + iV_ sampleData.Timestamp.ToString()
    + " - GazeRX:" + sampleData.GazeRX.ToString()
    + " - GazeRY:" + sampleData.GazeRY.ToString()
    + " - GazeLX:" + sampleData.GazeLX.ToString()
    + " - GazeLY:" + sampleData.GazeLY.ToString()
    + " - DiamRX:" + sampleData.DiamRX.ToString()
    + " - DiamLX:" + sampleData.DiamLX.ToString()
    + " - DistanceR:" + sampleData.DistanceR.ToString()
    + " - DistanceL:" + sampleData.DistanceL.ToString();
}

private void disconnect_Click(object sender, EventArgs e)
{
    iV_Disconnect();
}
```

MATLAB®

The SDK includes three MATLAB® example programs to help you get started with developing your own applications. They will provide you with insights on how to setup experiments using the iView X™ API.

To run the Slideshow and GazeContingent MATLAB® example script enclosed in the iView X™ SDK it's necessary to download and install the "psychophysics toolbox" from http://psychtoolbox.org. The psychophysics toolbox provides MATLAB® specific visualizations being used in this example. Read the "psychophysics toolbox" wiki for more information. Please note though that the toolbox is used for

visualization purposes and is not required for communication with eyetracking server. The examples Slideshow and Gaze Contingent demonstrate how to use the "psychophysics toolbox" in combination with eye tracking. For using the iView X™ SDK without the "psychophysics toolbox" have a look into the DataStreaming example enclosed in the iView X™ SDK. Due to changes in MATLAB® in handing over parameters to dynamic libraries, the MATLAB® examples are available for version 7.0 and version 7.11. If you are using versions in between or a later version it's recommended to try the 7.11 examples first. By default the MATLAB® installer selects the 32bit or 64bit version corresponding to the computer architecture. To avoid compatibility issues with the 32bit or 64bit iView X™ API the example scripts will select it's API versions automatically (iViewXAPI.dll for 32bit or iViewXAPI64.dll for 64bit). Important Note: Using a Windows 64bit version its required to have the Visual Studio C++ 2010 SP1 or 2012 installed, the "X64 Compilers and Tools" option checked during installation and selected the compiler using the "mex -setup" command. For more information please read the MATLAB® compatible compilers support website: http://www.mathworks.de/support/compilers/R2014a/index.html Unlike the C# demo application, the MATLAB® examples do not have a built-in user interface. However, it is still possible to use the same functionality as the C# demo and create a similar user interface programmatically or through use of GUIDE, the MATLAB® graphical user interface development environment.

The following code shows how to load the required 32bit SDK DLL. It also defines a struct which is used to receive online data from the eye tracking device:

```
loadlibrary('iViewXAPI.dll', 'iViewXAPI.h');

Eye.gazeX = int32(0);
Eye.gazeY = int32(0);
Eye.diam = int32(0);
Eye.eyePositionX = int32(0);
Eye.eyePositionY = int32(0);
Eye.eyeDistance = int32(0);
EyeData = libstruct('EyeDataStruct', Eye);
pEyeData = libpointer('EyeDataStruct', Eye);

Sample.Timestamp = int64(0);
Sample.leftEye = EyeData;
Sample.rightEye = EyeData;
Sample.planeNumber = int32(0);
pSample = libpointer('SampleStruct', Sample);
```

The code below illustrates how to connect to iView X™, obtain data samples from the eye tracker, and disconnect from iView X™. After disconnecting, the library has to be unloaded:

```
calllib('iViewXAPI', 'iV_Connect', '127.0.0.1', int32(4444), '127.0.0.1', int32(5555))

calllib('iViewXAPI', 'iV_GetSample', pSample)
get(pSample, 'Value')

calllib('iViewXAPI', 'iV_Disconnect')
unloadlibrary('iViewXAPI');
```

Python

The iView X™ SDK includes four sample experiments for use with Python. To run the experiments "↩ Slideshow" and "GazeContingent", it is necessary to download and install the "Psychopy toolbox" from http://www.psychopy.org/. The Psychopy toolbox is an open source toolbox that allows presentation of stimuli and collection of data for a wide range of neuroscience, psychology and psychophysics

experiments. In particular, the Psychopy toolbox provides Python specific visualizations being used in these examples. Please note that the toolbox is NOT required for communication with iView X™, it is used for stimulus visualisation in the said experiments. These Python examples were written with Python version 2.7.5. and the Psychopy2 toolbox version 1.73.06.

Installing Prerequisites

1. Python 2.7.5 or later versions from http://www.python.org or any other source
2. Optional: PsychoPy Toolbox and additional libraries from http://www.lfd.uci.edu/~gohlke/pythonlibs/ or any other source
   (a) PsychoPy Toolbox 1.73.06
   (b) Numpy
   (c) Pyglet
   (d) Python Imaging library
   (e) wxpython
   (f) wxPython-common
   (g) Dateutil
   (h) Pyparsing

Running Examples

1. Start iView X™, iView RED-m, iView RED-OEM or eye tracking-server
2. Run Python script

Creating a Custom Application

The following code shows how to load the required SDK DLL, connecting to the server, retrieving data and disconnecting from iView X™:

```
from ctypes import *

class CEye(Structure):
    _fields_ = [("gazeX", c_double),
    ("gazeY", c_double),
    ("diam", c_double),
    ("eyePositionX", c_double),
    ("eyePositionY", c_double),
    ("eyePositionZ", c_double)]

class CSample(Structure):
    _fields_ = [("timestamp", c_longlong),
    ("leftEye", CEye),
    ("rightEye", CEye),
    ("planeNumber", c_int)]

leftEye = CEye(0,0,0)
rightEye = CEye(0,0,0)
```

```
sampleData = CSample(0,leftEye,rightEye,0)
iViewXAPI = windll.LoadLibrary("iViewXAPI.dll")
iViewXAPI.iV_Connect(c_char_p('127.0.0.1'), c_int(4444), c_char_p('127.0.0.1'), c_int(5555))
iViewXAPI.iV_GetSample(byref(sampleData))
iViewXAPI.iV_Disconnect()
```

It's recommended to use the following files as wrappers to access the iView X™ SDK.

- iViewXAPI.py demonstrates how to import the iView X™ SDK library and how to declare and initialize data structure that are needed for the use of the iView X™ SDK functions.
- iViewXAPIReturnCodes.py handles iView X™ SDK return codes.

Advanced Usage

Setting up RED and RED-m Geometry

The SDK can be used to configure the monitor attached mode for the RED and the stand alone mode for RED and RED-m.

RED Monitor Attached Mode

For monitor attached mode, the following parameters from the structure REDGeometryStruct are relevant:

Parameter	Value
REDGeometryStruct::redGeometry	REDGeometryEnum::monitorIntegrated
REDGeometryStruct::monitorSize	`19` or `22`

The function iV_SetREDGeometry configures the settings related to the display device. The monitor attached mode is not available for RED-m.

RED Stand Alone

The data structure REDGeometryStruct contains all required geometrical parameters. The function iV_SetREDGeometry configures the stand alone geometry.

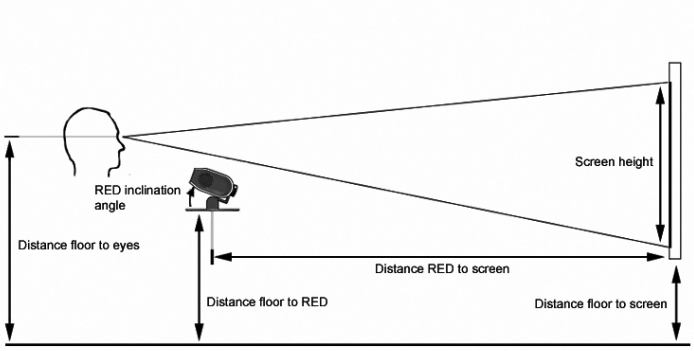

Figure 1.12: RED Stand Alone Mode

The following steps are necessary to setup the RED in stand-alone mode:

1. Remove the RED from the monitor and mount it on the stand-alone foot.
2. Position your external screen (beamer, TV, monitor) as follows:
   - The screen has to be planar
   - The screen has to be at right angle with the floor
   - The screen bottom line has to be parallel to the floor
   - RED is in the horizontal middle of the display device
3. Enter a profile name and the following geometrical dimensions of your setup into REDGeometryStruct
4. Call the function iV_SetREDGeometry including the REDGeometryStruct as parameter to eye-tracking server

Parameter	Value
REDGeometryStruct::redGeometry	REDGeometryEnum::standalone
REDGeometryStruct::setupName	Profile name
REDGeometryStruct::stimX	Screen width [mm]
REDGeometryStruct::stimY	Screen height [mm]
REDGeometryStruct::stimHeightOverFloor	Distance floor to screen [mm]
REDGeometryStruct::redHeightOverFloor	Distance floor to RED [mm]
REDGeometryStruct::redStimDist	Distance RED to screen [mm]
REDGeometryStruct::redInclAngle	RED inclination angle [degree]

RED-m

Note: Although attached to a screen, the geometrical set up has to be regarded as "stand alone" due to advanced options for configuration.

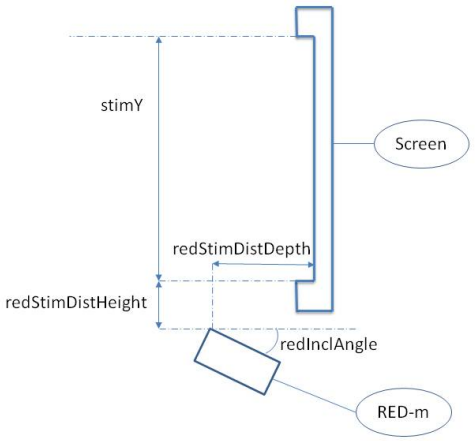

Figure 1.13: RED-m Stand Alone Mode

The following steps are necessary to setup the RED-m in stand alone mode:

1. Position your RED-m and your screen (beamer, TV, monitor) as follows:
   - RED-m is in the horizontal middle of the display device
   - Position and align the RED-m in a way that the user's head is in the middle of the tracking box.
2. Enter a profile name and the following geometrical dimensions of your setup into REDGeometryStruct
3. Call the function iV_SetREDGeometry including the REDGeometryStruct as parameter to eye-tracking server

Parameter	Value
REDGeometryStruct::redGeometry	REDGeometryEnum::standalone
REDGeometryStruct::setupName	Profile name
REDGeometryStruct::stimX	Screen width [mm]
REDGeometryStruct::stimY	Screen height [mm]
REDGeometryStruct::redStimDistHeight	Vertical distance RED-m to stimulus screen [mm]
REDGeometryStruct::redStimDistDepth	Horizontal distance RED-m to stimulus screen [mm]
REDGeometryStruct::redInclAngle	RED-m inclination angle [degree]

Areas of Interest (AOI)

The Area of Interest (AOI) feature allows you to define rectangular objects within the stimulus for high level gaze and fixation analysis. Client application is informed whenever the raw gaze data enters or leaves an AOI, or an online detected fixation event was calculated within an AOI. If idf recording is running a message will be send to the idf data stream. This is useful if you wish to trigger and synchronize other measurement devices with the gaze position. See reference information for iV_DefineAOI and AOIStruct how to define AOIs.

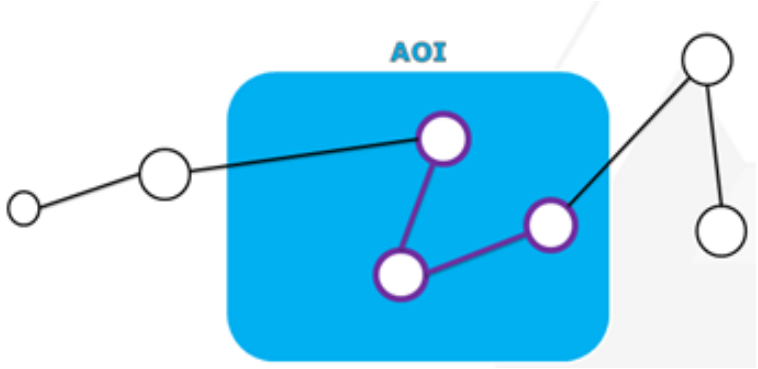

Figure 1.14: Areas of Interest

AOI interaction can be signaled to the LPT port. To define the port in use, call the function iV_DefineAOIPort. `outputValue` from AOIStruct can be used to define the TTL value that is send if the corresponding AOI is hit by gaze or fixation.

Smart Binocular and Monocular Tracking Mode

The iView X™ SDK is able to handle and setup different tracking modes which are supported by SMI RED devices.

The default tracking mode is `SMARTBINOCULAR BOTH` and is aimed to track and calculate the gaze of both eyes of the participant, but will tolerate if just one eye is visible for a certain time span. In this case the system is still able to track the participant, to calculate the gaze cursor, and compensate the head movements. This mode is enabled by default (application start), but to set it during run time, the following function needs to be called:

```
iV_SetTrackingParameter( ET_PARAM_EYE_BOTH, ET_PARAM_SMARTBINOCULAR, 0);
```

In addition to `SMARTBINOCULAR BOTH`, the user can choose between `SMARTBINOCULAR LEFT` and `SMARTBINOCULAR RIGHT` to select the data channel for a specific eye. To setup this mode, the following function needs to be called:

```
iV_SetTrackingParameter( ET_PARAM_EYE_RIGHT, ET_PARAM_SMARTBINOCULAR, 0);
iV_SetTrackingParameter( ET_PARAM_EYE_LEFT, ET_PARAM_SMARTBINOCULAR, 0);
```

Note: The purpose of `LEFT` or `RIGHT` is to track people who have both eyes visible, but only one active eye. E.g. if somebody would have a strong strabism with one eye, the recommended mode would be the `SMARTBINOCULAR LEFT|RIGHT` mode to stop calculating gaze data from the strabism eye. In this case, due to robustness the RED device looks for both eyes, but ignores the strabism eye's data channel.

The `MONOCULAR` mode is designed to track participants with just one visible eye. The tracking of the participant, gaze calculation, and head movement compensation will be calculated just out of one visible eye and ignoring a second one. The active data channel will be written, corresponding to the mode, into the data file. The data of the second channel will be set to zero.

```
iV_SetTrackingParameter( ET_PARAM_EYE_RIGHT, ET_PARAM_MONOCULAR, 0);
iV_SetTrackingParameter( ET_PARAM_EYE_LEFT, ET_PARAM_MONOCULAR, 0);
```

Note: For participants with both eyes visible this mode might have a reduced robustness.

Single PC and Dual PC Setup

iView X™ API handles control flow and data flow between customer application and eyetracking server. Control commands are submitted from the customer application and are addressed to the eyetracking server. Data is produced by the eyetracking server and is sent to the customer application. Therefore, a bidirectional connection is needed. Low level communication between the iView X™ API component itself and eyetracking server is realized via UDP/IP network communication. Therefore, a customer application and eyetracking server have to configure the communication channels. Please refer to your system's manual to learn how to set up network connection at eyetracking server side.

For customer applications, there are two ways to communicate with the eyetracking server via the iView X™ API:

- Single PC Setup
- Dual PC Setup

Both methods are described below.

Single PC Setup

Customer application and eye tracking device are running on the same PC.

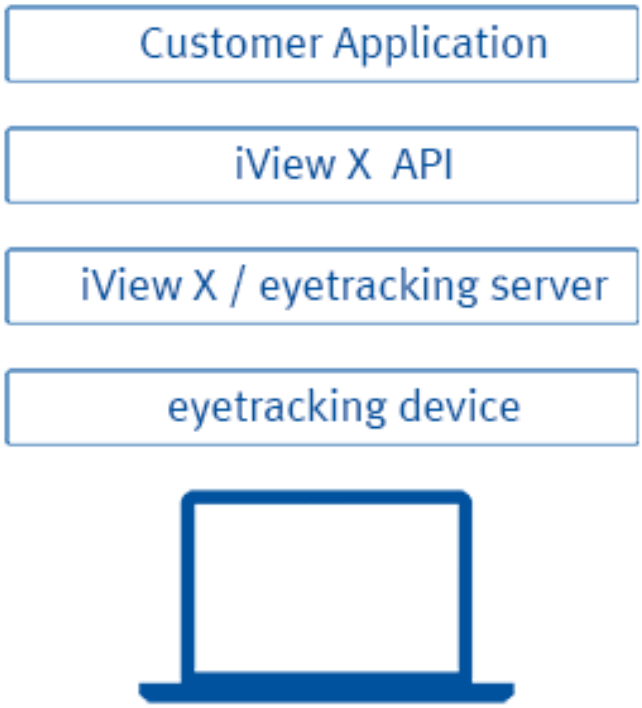

Figure 1.15: Single PC Setup

Although no hardware network connection is used, customer application has to setup a `localhost` network connection to access eyetracking server. Typically, this is realized using the IP address `127.0.0.1`. The port settings have to be mirrored:

- `SendPort` from customer application has to be the `ReceivePort` from eyetracking server. Default port number is `4444`.
- `ReceivePort` from customer application has to be the `SendPort` from eyetracking server. Default port number is `5555`.

Parameters of iV_Connect are:

```
iV_Connect( sendIPAddress, sendPort, recvIPAddress, receivePort);
```

In the described case iV_Connect has to be called from customer application in the following way:

```
iV_Connect( "127.0.0.1", 4444, "127.0.0.1", 5555);
```

Dual PC Setup

Customer application and eyetracking server are running on different PCs. Both PCs are connected via Ethernet.

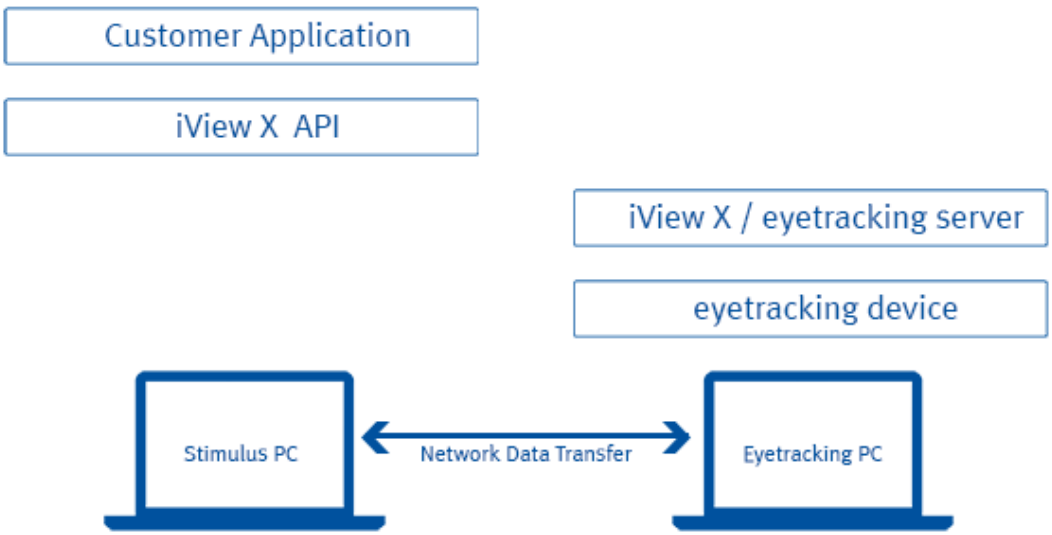

Figure 1.16: Dual PC Setup

For this example we assume the following IP addresses:

PC	IP address
Stimulus PC	192.168.1.1
Eyetracking PC	192.168.1.2

In eyetracking server, the network settings have to be configured as follows:

Direction	IP address	Port
Receive/Listen	192.168.1.2	4444
Send To	192.168.1.1	5555

iV_Connect has to be called from customer application in the following way:

```
iV_Connect( "192.168.1.2", 4444, "192.168.1.1", 5555);
```

Connecting with Multiple Customer Applications

Please Note: This feature is only available for RED-m and RED-OEM devices. It requires iView X™ SDK version 3.4.6 or newer and eyetracking server version 2.11.65 or newer.

To run multiple applications or multiple instances of the same application in parallel, each running instance has to establish its own communication channel.

The mechanism described in Single PC and Dual PC Setup allows configuration of one or at the maximum two communication channels - depending on the underlying eye tracking software's capabilities.

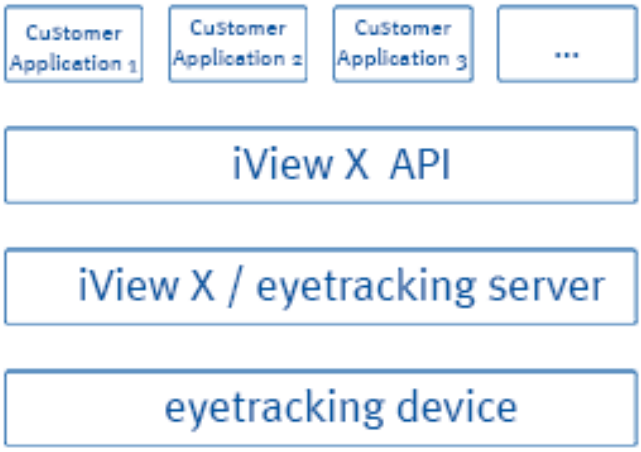

Figure 1.17: Multiple customer applications on a Single PC

iV_ConnectLocal establishes a connection similiar to iV_Connect. With iV_ConnectLocal port settings are handled automatically. There is no need to iV_Disconnect a connection created with iV_ConnectLocal.

Polling vs. Callbacks

iView X™ API provides two ways to access eye tracking data online:

- Polling
- Callbacks

The following table shows the interface functions to be used when realizing certain tasks with polling or callbacks.

Task	Polling	Callbacks
Get event data	iV_GetEvent	iV_SetEventCallback
Get sample data	iV_GetSample	iV_SetSampleCallback
Get current calibration point	iV_GetCurrentCalibrationPoint	iV_SetCalibrationCallback
Get eye images	iV_GetEyeImage	iV_SetEyeImageCallback
Get HED scene images	iV_GetSceneVideo	iV_SetSceneVideoCallback
Get RED Tracking Monitor Image	iV_GetTrackingMonitor	iV_SetTrackingMonitorCallback

Get AOI Hits	iV_GetAOIOutputValue	iV_SetAOIHitCallback

Both methods provide different features, advantages and disadvantages. With **polling** the customer application has full control about the calling frequency of the polling function. Returned data will always contain the latest known values, independently if they have

- not been updated
- updated once
- updated several times

since the last call. **Callback** Functions are called by the API as often as the data is updated by the underlying eyetracking server. Restrictions may apply due to system load.

Please note:

- Callback functions are not called as long as the previously executed callback of the same type has not finished. Therefore, it is recommended to put only very short and fast executing commands into callbacks.
- Callbacks are not available in all programming languages.
- Callback functions are called from different threads. Therefore, the code within callback functions has to be thread safe.
- While Polling for images (Eye Image, Tracking Monitor, Scene Video, Accuracy Image) its recommended to use only one ImageStruct instance for each data set.

Running a Validation

To evaluate the calibration quality the participant may perform a validation after the calibration. For that, iV_Validate has to be called. A sequence of four points is presented to the user, similar to the calibration procedure. The validation calculates the difference between the presented validation points and the measured gaze points. Overall results of the validation can be retrieved with iV_GetAccuracy, iV_GetAccuracyImage or iV_ShowAccuracyMonitor.

Function and Device Overview

The table below provides an overview of the various functions available in the iView X™ SDK along with their corresponding supported SMI eye tracking devices. More detailed information pertaining to these functions follows in the iView X™ SDK Reference section.

Function	RED	RED-m	RED-mx	HiSpeed/↩ Primate	HED	MRI/ MEG

iV_Abort↩ Calibration	X	X	X	X	-	X
iV_↩ Accept↩ Calibration↩ Point	X	X	X	X	-	X
iV_↩ Calibrate	X	X	X	X	-	X
iV_↩ Change↩ Calibration↩ Point	X	X	X	X	X	X
iV_Clear↩ AOI	X	X	X	X	-	X
iV_Clear↩ Recording↩ Buffer	X	X	X	X	X	X
iV_↩ Configure↩ Filter	X	X	X	X	X	X
iV_Connect	X	X	X	X	X	X
iV_↩ Connect↩ Local	-	X	X	-	-	-
iV_↩ Continue↩ Eyetracking	-	X	X	-	-	-
iV_↩ Continue↩ Recording	X	X	X	X	X	X
iV_Define↩ AOI	X	X	X	X	-	X
iV_Define↩ AOIPort	X	X	X	X	-	X
iV_Delete↩ RED↩ Geometry	X	X	X	-	-	-
iV_↩ DisableA↩ OI	X	X	X	X	-	X

iV_DisableAOIGroup	X	X	X	X	-	X
iV_DisableProcessorHighPerformanceMode	-	X	X	-	-	-
iV_DisableGazeDataFilter	X	X	X	X	-	X
iV_Disconnect	X	X	X	X	X	X
iV_EnableAOI	X	X	X	X	-	X
iV_EnableAOIGroup	X	X	X	X	-	X
iV_EnableGazeDataFilter	X	X	X	X	-	X
iV_EnableProcessorHighPerformanceMode	-	X	X	-	-	-
iV_GetAccuracy	X	X	X	X	-	X
iV_GetAccuracyImage	X	X	X	X	-	X
iV_GetAOIOutputValue	X	X	X	X	-	X

iV_Get↩ Calibration↩ Point	X	X	X	X	X	X
iV_Get↩ Calibration↩ Status	X	X	X	X	X	X
iV_Get↩ Current↩ Calibration↩ Point	X	X	X	X	X	X
iV_Get↩ CurrentR↩ ED↩ Geometry	X	X	X	-	-	-
iV_Get↩ Current↩ Timestamp	X	X	X	X	X	X
iV_Get↩ Device↩ Name	-	X	X	-	-	-
iV_Get↩ Event	X	X	X	X	-	X
iV_Get↩ Event32	X	X	X	X	-	X
iV_Get↩ EyeImage	X	X	-	X	X	X
iV_Get↩ License↩ DueDate	X	X	X	X	X	X
iV_GetRE↩ DGeometry	X	X	X	-	-	-
iV_Get↩ Sample	X	X	X	X	X	X
iV_Get↩ Sample32	X	X	X	X	X	X
iV_Get↩ Scene↩ Video	-	-	-	-	X	-
iV_Get↩ Serial↩ Number	-	X	X	-	-	-

iV_GetSystemInfo	X	X	X	X	X	X
iV_GetTrackingMonitor	X	X	X	-	-	-
iV_GetTrackingStatus	X	X	X	X	X	X
iV_HideAccuracyMonitor	X	X	X	X	X	X
iV_HideEyeImageMonitor	X	X	X	X	X	X
iV_HideSceneVideoMonitor	-	-	-	-	X	-
iV_HideTrackingMonitor	X	X	X	-	-	-
iV_IsConnected	X	X	X	X	X	X
iV_LoadCalibration	X	X	X	X	-	X
iV_Log	X	X	X	X	X	X
iV_PauseEyetracking	-	X	X	-	-	-
iV_PauseRecording	X	X	X	X	X	X
iV_Quit	X	X	X	X	X	X
iV_ReleaseAOIPort	X	X	X	X	-	X
iV_RemoveAOI	X	X	X	X	-	X
iV_ResetCalibrationPoints	X	X	X	X	X	X

iV_Save↩ Calibration	X	X	X	X	-	X
iV_Save↩ Data	X	X	X	X	X	X
iV_Select↩ RED↩ Geometry	X	X	X	-	-	-
iV_Send↩ Command	X	X	X	X	X	X
iV_Send↩ Image↩ Message	X	X	X	X	X	X
iV_SetAO↩ IHit↩ Callback	X	X	X	X	-	X
iV_Set↩ Calibration↩ Callback	X	X	X	X	-	X
iV_Set↩ Connection↩ Timeout	X	X	X	X	X	X
iV_Set↩ Event↩ Callback	X	X	X	X	-	X
iV_Set↩ Event↩ Detection↩ Parameter	X	X	X	X	-	X
iV_Set↩ Eye↩ Image↩ Callback	X	X	-	X	X	X
iV_Set↩ License	-	-	-	-	-	-
iV_Set↩ Logger	X	X	X	X	X	X
iV_Set↩ Resolution	X	X	X	X	-	X
iV_Set↩ Sample↩ Callback	X	X	X	X	X	X

iV_Set↩ Scene↩ Video↩ Callback	-	-	-	-	X	-
iV_Set↩ Tracking↩ Monitor↩ Callback	X	X	X	-	-	-
iV_Set↩ Tracking↩ Parameter	-	X	X	X	X	X
iV_Setup↩ Calibration	X	X	X	X	-	X
iV_SetRE↩ DGeometry	X	X	X	-	-	-
iV_Show↩ Accuracy↩ Monitor	X	X	X	X	-	X
iV_Show↩ Eye↩ Image↩ Monitor	X	X	-	X	X	X
iV_Show↩ Scene↩ Video↩ Monitor	-	-	-	-	X	-
iV_Show↩ Tracking↩ Monitor	X	X	X	-	-	-
iV_Start	X	X	X	X	X	X
iV_Start↩ Recording	X	X	X	X	X	X
iV_Stop↩ Recording	X	X	X	X	X	X
iV_TestT↩ TL	X	X	X	X	-	X
iV_Validate	X	X	X	X	-	X

Explanation of Defines

Return Values

The Return values listed in the header defines all possible return codes which can be returned by the API functions.

Return Code	Value	Description
RET_SUCCESS	1	intended functionality has been fulfilled
RET_NO_VALID_DATA	2	no new data available
RET_CALIBRATION_ABOR↩ TED	3	calibration / validation was aborted during progress
RET_SERVER_IS_RUNNING	4	server is running
RET_CALIBRATION_NOT_I↩ N_PROGRESS	5	calibration / validation is not in progress
RET_WINDOW_IS_OPEN	11	window is open
RET_WINDOW_IS_CLOSED	12	window is closed
ERR_COULD_NOT_CONNE↩ CT	100	failed to establish connection
ERR_NOT_CONNECTED	101	no connection established
ERR_NOT_CALIBRATED	102	system is not calibrated
ERR_NOT_VALIDATED	103	system is not validated
ERR_EYETRACKING_APPL↩ ICATION_NOT_RUNNING	104	no eye tracking application running
ERR_WRONG_COMMUNIC↩ ATION_PARAMETER	105	failed to establish connection
ERR_WRONG_DEVICE	111	eye tracking device required for this function is not connected
ERR_WRONG_PARAMETER	112	parameter out of range
ERR_WRONG_CALIBRATIO↩ N_METHOD	113	eye tracking device required for this calibration method is not connected
ERR_BIND_SOCKET	123	the defined port is blocked
ERR_DELETE_SOCKET	124	failed to delete sockets
ERR_NO_RESPONSE_FRO↩ M_IVIEWX	131	iView X (eyetracking-server) application was not able to response to current request
ERR_WRONG_IVIEWX_VE↩ RSION	133	wrong version of iView X (eyetracking-server) application
ERR_ACCESS_TO_FILE	171	failed to access log file
ERR_EMPTY_DATA_BUFF↩ ER	191	recording buffer is empty
ERR_RECORDING_DATA_↩ BUFFER	192	recording is activated

ERR_FULL_DATA_BUFFER	193	data buffer is full
ERR_IVIEWX_IS_NOT_READY	194	iView X (eyetracking-server) application is not ready to record buffer
ERR_PAUSED_DATA_BUFFER	195	recording buffer is paused
ERR_IVIEWX_NOT_FOUND	201	iView X (eyetracking-server) application was not found
ERR_CAMERA_NOT_FOUND	211	failed to access eye tracking device
ERR_WRONG_CAMERA	212	failed to access eye tracking device
ERR_WRONG_CAMERA_PORT	213	failed to access port connected to eye tracking device
ERR_COULD_NOT_OPEN_PORT	220	failed to open port
ERR_COULD_NOT_CLOSE_PORT	221	failed to close port
ERR_AOI_ACCESS	222	failed to access AOI data
ERR_FEATURE_NOT_LICENSED	250	failed to access requested feature
ERR_INITIALIZATION	400	failed to initialize function

Logging Level

With these defines handed over to the function iV_SetLogger it is possible to setup the internal logging status of the API as well as the content which will be logged. Log levels can be combined (e.g. LOG_BUG | LOG_IV_COMMAND | LOG_ETCOM). With iV_Log it is possible to store additional messages in the internal logfile.

Parameter	Value	Description
LOG_LEVEL_BUG	1	logs internal issues
LOG_LEVEL_iV_FCT	2	logs all calls to API functions
LOG_LEVEL_ALL_FCT	4	logs all calls to internal functions
LOG_LEVEL_IV_COMMAND	8	logs the communication from API to iView X (eyetracking-server) application
LOG_LEVEL_RECV_IV_COMMAND	16	logs the communication from iView X (eyetracking-server) application to API

Eye Tracking Parameter

With ET_PARAM_ and function iV_SetTrackingParameter it is possible to change iView X and eyetracking-server tracking parameters, for example pupil threshold and corneal reflex thresholds, eye image contours, and other parameters. Important note: This function can strongly affect tracking stability of your iView X and eyetracking-server system. Only experienced users should use this function.

Parameter	Value	Description
ET_PARAM_EYE_LEFT	0	set parameter for the left eye
ET_PARAM_EYE_RIGHT	1	set parameter for the left eye
ET_PARAM_EYE_BOTH	2	set parameter for both eyes
ET_PARAM_PUPIL_THRESHOLD	0	set pupil threshold parameter
ET_PARAM_REFLEX_THRESHOLD	1	set reflex threshold parameter
ET_PARAM_SHOW_AOI	2	enabling/disabling AOI overlays
ET_PARAM_SHOW_CONTOUR	3	enabling/disabling eye contour overlays
ET_PARAM_SHOW_PUPIL	4	enabling/disabling pupil center overlays
ET_PARAM_SHOW_REFLEX	5	enabling/disabling reflex center overlays
ET_PARAM_DYNAMIC_THRESHOLD	6	enabling/disabling dynamic pupil threashold
ET_PARAM_PUPIL_AREA	11	set pupil area parameter
ET_PARAM_PUPIL_PERIMETER	12	set pupil perimeter
ET_PARAM_PUPIL_DENSITY	13	set pupil density parameter
ET_PARAM_REFLEX_PERIMETER	14	set reflex perimeter
ET_PARAM_REFLEX_PUPIL_DISTANCE	15	set reflex pupil distance parameter
ET_PARAM_MONOCULAR	16	set tracking mode to monocular
ET_PARAM_SMARTBINOCULAR	17	set tracking mode to smart binocular
ET_PARAM_BINOCULAR	18	set tracking mode to binocular

Groups of Functions

Topic	List of Related Functions
System Start and Stop, System Information and Connection	iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start
Calibration	iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration
Validation	iV_GetAccuracy, iV_GetAccuracyImage, iV_HideAccuracyMonitor, iV_ShowAccuracyMonitor, iV_Validate
Data Acquisition	iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback
Eye Data Recording	iV_ClearRecordingBuffer, iV_ContinueRecording, iV_PauseRecording, iV_SaveData, iV_SendImageMessage, iV_StartRecording, iV_StopRecording
Eye Image Handling	iV_GetEyeImage, iV_HideEyeImageMonitor, iV_SetEyeImageCallback, iV_ShowEyeImageMonitor
HED Scene Video	iV_GetSceneVideo, iV_HideSceneVideoMonitor, iV_SetSceneVideoCallback, iV_ShowSceneVideoMonitor
RED Tracking Monitor Handling	iV_GetTrackingMonitor, iV_HideTrackingMonitor, iV_SetTrackingMonitorCallback, iV_ShowTrackingMonitor

AOI Trigger	iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL
Geometry RED	iV_DeleteREDGeometry, iV_GetCurrentREDGeometry, iV_GetGeometryProfiles, iV_GetREDGeometry, iV_SelectREDGeometry, iV_SetREDGeometry
Gaze Data Filter	iV_DisableGazeDataFilter, iV_EnableGazeDataFilter, iV_ConfigureFilter
Logging	iV_Log, iV_SetLogger
Other	iV_SendCommand, iV_SetTrackingParameter

Frequently Asked Questions

- How to link with minGW?
- How to record eye images

How to link with minGW?

As created with Microsoft Visual Studio, currently there is no way to link iView X™ API library with minGW. When using a different compiler, we recommend to use Windows' `GetProcAddress` mechanism to access functions from iViewXAPI.dll.

How to record eye images

Eye image recording functionality is available for certain devices only. Please refer to your eyetracking device's manual to learn details. To start eye image recording call iV_SendCommand and pass a string to it:

```
iV_SendCommand("ET_EVB [type] [prefix] [path]");
```

Please note that depending on your system settings different image file types are available. For storing jpeg images use `type = 0`

```
iV_SendCommand("ET_EVB 0 img d:\\eyeimages\\");
```

This will store eye images to a subfolder (named by time and date) of d:\eyeimages. The image names contain the prefix `img`, a consecutive number and some image aquisition related information. To stop eye image recording recording call

```
iV_SendCommand("ET_EVE");
```

Please note: High CPU load and disk space requirements of eye image recording may impact your system's eyetracking performance. We do not recommend using eye image recording permanently to avoid interference with eyetracking performance.

1.3 Appendix

License Agreement and Warranty for SDK Provided Free of Charge

IMPORTANT – PLEASE READ CAREFULLY: This license agreement ("Agreement") is an agreement between you (either an individual or a company, "Licensee") and SensoMotoric Instruments GmbH ("SMI"). The "Licensed Materials" provided to Licensee free of charge subject to this Agreement include the Software Development Kit (the "SDK") as well as any "on-line" or electronic documentation associated with the SDK, or any portion thereof (the "Documentation"), as well as any updates or upgrades to the SDK and Documentation, if any, or any portion thereof, provided to Licensee at SMI's sole discretion. By installing, downloading, copying or otherwise using the Licensed Materials, you agree to abide by the following provisions. This Agreement is displayed for you to read prior to using the Licensed Materials. If you do not agree with these provisions, do not download, install or use the Licensed Materials.

1. License Subject to the terms of this Agreement, SMI hereby grants and Licensee accepts a non-transferable, non-exclusive, non-assignable license without the right to sublicense to use the Licensed Materials only (i) for Licensee's operations, (ii) with regards to the SMI Eye Tracking application iView X™ and (iii) in accordance with the Documentation. Installation of the SDK is Licensee's sole responsibility.

2. Rights in Licensed Materials Title to and ownership in the Licensed Materials and all proprietary rights with respect to the Licensed Materials and all copies and portions thereof, remain exclusively with SMI. The Agreement does not constitute a sale of the Licensed Materials or any portion or copy of it. Title to and ownership in Licensee's application software that makes calls to but does not contain all or any portion of the SDK remains with Licensee, but such application software may not be licensed or otherwise transferred to third parties without SMI's prior written consent.

3. Confidentiality Licensed Materials are proprietary to SMI and constitute SMI trade secrets. 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 secrets, but in no event less than a reasonable degree of care. Licensee shall not disclose Licensed Materials or any part thereof to anyone for any purpose, other than to its employees and sub-contractors 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.

4. Limited Warranty and Liability a) The SDK is provided "as is". b) SMI's warranty obligations are limited to fraudulently concealed defects of the Licensed Material. c) SMI is only liable for damages caused by gross negligence or intent. d) With the exception of liability under the Product Liability Law, for defects after having given a guarantee, for fraudulently concealed defects and for personal injury, the above limitations of liability shall apply to all claims, irrespective of their legal basis, in particular to all claims based on breach of contract or tort. e) The above limitations of liability also apply in case of Licensee's claims for damages against SMI's employees or agents.

5. Licensee Indemnity Licensee will defend and indemnify SMI, and hold it harmless from all costs, including attorney's fees, arising from any claim that may be made against SMI by any third party as a result of Licensee's use of Licensed Materials.

6. Export Restriction Licensee will not remove or export from Germany or from the country Licensed Materials were originally shipped to by SMI or re-export from anywhere any part of the Licensed Materials or any direct product of the SDK except in compliance with all applicable export laws and regulations, including without limitation, those of the U.S. Department of Commerce.

7. Non-Waiver; Severability; Non-Assignment. The delay or failure of either party to exercise any right provided in this Agreement shall not be deemed a waiver. If any provision of this Agreement is held invalid, all others shall remain in force. Licensee may not, in whole or in part, assign or otherwise transfer this Agreement or any of its rights or obligations hereunder.

8. Termination This Agreement may be terminated (i) by Licensee without cause on 30 days notice; (ii) by SMI, in addition to other remedies, if Licensee fails to cure any breach of its obligations hereunder within 30 days of notice thereof; (iii) on notice by SMI if there is a transfer of twenty-five percent (25%) or more of the ownership interest in Licensee, which in good faith is not acceptable to SMI, and on notice by either party if the other party ceases to do business in the normal course, becomes insolvent, or becomes subject to any bankruptcy, insolvency, or equivalent proceedings. Upon termination by either party for any reason, Licensee shall at SMI's instructions immediately destroy or return the Licensed Materials and all copies thereof to SMI and delete the SDK and all copies thereof from any computer on which the SDK had been installed.

9. Entire Agreement; Written Form Requirement. There are no separate oral agreements; any supplementary agreements or modifications hereto must be made in writing. This also applies to any waiver of this requirement of written form.

10. Notices All notices under the Agreement must be in writing and shall be delivered by hand or by overnight courier to the addresses of the parties set forth above.

11. Applicable Law and Jurisdiction German law applies with the exception of its conflict of laws rules. The application of the United Nations Convention on Contracts for the International Sale of Goods (CISG) is expressly excluded. The courts of Berlin, Germany, shall have exclusive jurisdiction for any action brought under or in connection with this Agreement. © Teltow, Germany, 2004-2014 SensoMotoric Instruments GmbH

Technical Support

Due to the complex nature of SDK's in general and the wide variety of applications that may be created using the iView X™ SDK, it is not always possible to provide in-depth support. However, if you feel there is an error or omission in the iView X™ SDK, please fill out a support request on the SMI website (http://www.smivision.com/en/gaze-and-eye-tracking-systems/support/support-request.html) and we will research the issue. Please note that if you should require technical assistance relating to the SDK and your application, SMI may request or require a copy of your application and elements of your source code. If you are new to programming, we would highly recommend that you consult a general programming guide for your desired language before attempting to use the iView X™ SDK to write

your own eyetracking application. The provided examples are included to help you in getting started with developing your software application, but they are not a substitute for programming knowledge.

About SMI

SensoMotoric Instruments (SMI) is a world leader in dedicated computer vision applications, developing and marketing eye & gaze tracking systems and OEM solutions for a wide range of applications. Founded in 1991 as a spin-off from academic research, SMI was the first company to offer a commercial, vision-based 3D eye tracking solution. We now have 20 years of experience in developing application-specific solutions in close collaboration with our clients. We serve our customers around the globe from our offices in Teltow, near Berlin, Germany and Boston, USA, backed by a network of trusted local partners in many countries. Our products combine a maximum of performance and usability with the highest possible quality, resulting in high-value solutions for our customers. Our major fields of expertise are: • Eye & gaze tracking systems in research and industry • High speed image processing, and • Eye tracking and registration solutions in ophthalmology. More than 4,000 of our systems installed worldwide are testimony to our continuing success in providing innovative products and outstanding services to the market. While SMI has won several awards, the largest reward for us each year is our trusted business relationships with academia and industry.

Please contact us:

Europe, Asia, Africa, South America, Australia
SensoMotoric Instruments GmbH (SMI)
Warthestraße 21
D-14513 Teltow
Germany
Phone: +49 3328 3955 0
Fax: +49 3328 3955 99
Email: info@smi.de

North American Headquarters
SensoMotoric Instruments, Inc.
28 Atlantic Avenue
236 Lewis Wharf
Boston, MA 02110
USA
Phone: +1 - 617 - 557 - 0010
Fax: +1 - 617 - 507 - 83 19
Toll-Free: 888 SMI USA1
Email: info@smivision.com

Please also visit our home page: http://www.smivision.com

Last updated: December 2013

Chapter 2

Reference

2.1 Data Types and Enumerations

Data Structures

- struct AccuracyStruct
- struct AOIRectangleStruct
- struct AOIStruct
- struct CalibrationPointStruct
- struct CalibrationStruct
- struct DateStruct
- struct EventStruct
- struct EventStruct32
- struct EyeDataStruct
- struct EyePositionStruct
- struct ImageStruct
- struct REDGeometryStruct
- struct SampleStruct
- struct SampleStruct32
- struct SystemInfoStruct
- struct TrackingStatusStruct

Typedefs

- typedef int(CALLBACK * pDLLSetAOIHit)(int digitalOutoutValue)
- typedef int(CALLBACK * pDLLSetCalibrationPoint)(CalibrationPointStruct calibrationPoint)
- typedef int(CALLBACK * pDLLSetEvent)(EventStruct eventDataSample)
- typedef int(CALLBACK * pDLLSetEyeImage)(ImageStruct eyeImage)
- typedef int(CALLBACK * pDLLSetSample)(SampleStruct rawDataSample)
- typedef int(CALLBACK * pDLLSetSceneVideo)(ImageStruct sceneVideo)
- typedef int(CALLBACK * pDLLSetTrackingMonitor)(ImageStruct trackingMonitor)

Enumerations

- enum CalibrationStatusEnum { calibrationUnknown = 0, calibrationInvalid = 1, calibrationValid = 2, calibrationInProgress = 3 }
- enum ETApplication { iViewX = 0, iViewXOEM = 1 }
- enum ETDevice { NONE = 0, RED = 1, REDm = 2, HiSpeed = 3, MRI = 4, HED = 5, Custom = 7 }
- enum FilterAction { Query = 0, Set = 1 }
- enum FilterType { Average = 0 }
- enum REDGeometryEnum { monitorIntegrated = 0, standalone = 1 }

Detailed Description

Data Structure Documentation

struct AccuracyStruct

This struct provides information about the last validation. Therefore a valid validation must be successfully completed before the AccuracyStruct can be updated. To update information in AccuracyStruct use function iV_GetAccuracy.

Data Fields

double	deviationLX	horizontal calculated deviation for left eye [degree]
double	deviationLY	vertical calculated deviation for left eye [degree]
double	deviationRX	horizontal calculated deviation for right eye [degree]
double	deviationRY	vertical calculated deviation for right eye [degree]

struct AOIRectangleStruct

Use this struct to customize the AOI position on screen. AOIRectangleStruct is a part of AOIStruct and can be defined with iV_DefineAOI.

Data Fields

int	x1	x-coordinate of left border of the AOI [pixel]
int	x2	x-coordinate of right border of the AOI [pixel]
int	y1	y-coordinate of upper border of the AOI [pixel]
int	y2	y-coordinate of lower border of the AOI [pixel]

struct AOIStruct

Use this struct to customize trigger AOIs. To define AOIs on screen, trigger parameter and trigger values use iV_DefineAOIPort and iV_DefineAOI functions.

Data Fields

char	aoiGroup[256]	group name of AOI
char	aoiName[256]	name of AOI
int	enabled	enable/disable trigger functionality [1: enabled, 0: disabled]
char	eye	['l', 'r']
int	fixationHit	uses fixations or raw data as trigger [1: fixation hit, 0: raw data hit]
char	outputMessage[256]	message in idf data stream
int	outputValue	TTL output value.
struct AOIRectangleStruct	position	position of AOI

struct CalibrationPointStruct

This struct provides information about the position of calibration points. To update information in CalibrationPointStruct during a calibration or validation use function iV_GetCurrentCalibrationPoint. Before or after the calibration use iV_GetCalibrationPoint.

Data Fields

int	number	number of calibration point
int	positionX	horizontal position of calibration point [pixel]
int	positionY	vertical position of calibration point [pixel]

struct CalibrationStruct

Use this struct to customize the calibration and validation behavior. To set calibration parameters with CalibrationStruct use function iV_SetupCalibration before a calibration or validation is started.

Data Fields

int	autoAccept	set calibration/validation point acceptance [0: manual, 1: semi automatic (default), 2: full automatic]
int	backgroundBrightness	set calibration/validation background brightness [0..255] (default: 220)
int	displayDevice	set display device [0: primary device (default), 1: secondary device]

int	foreground↩ Brightness	set calibration/validation target brightness [0..255] (default: 250)
int	method	select calibration method (default: 5)
int	speed	set calibration/validation speed [0: slow (default), 1: fast]
char	target↩ Filename[256]	select custom calibration/validation target (only if targetShape = 0)
int	targetShape	set calibration/validation target shape [IMAGE = 0, CIRCLE1 = 1, CIRCLE2 = 2 (default), CROSS = 3]
int	targetSize	set calibration/validation target size (default: 20 pixels)
int	visualization	draw calibration/validation [0: no API visualization , 1: API visualization (default)]

struct DateStruct

Use this struct to get the license due date of the device. Use the function iV_GetLicenseDueDate to update information in DateStruct.

Data Fields

int	day	day of license expiration
int	month	month of license expiration
int	year	year of license expiration

struct EventStruct

This struct provides information about the last eye event that has been calculated. To update information in EventStruct use function iV_GetEvent or set the event callback with with iV_SetEventCallback.

Data Fields

long long	duration	duration of the event [microseconds]
long long	endTime	end time of the event [microseconds]
char	eventType	type of eye event, 'F' for fixation (only fixations are supported)
char	eye	related eye, 'l' for left eye, 'r' for right eye
double	positionX	horizontal position of the fixation event [pixel]
double	positionY	vertical position of the fixation event [pixel]
long long	startTime	start time of the event [microseconds]

struct EventStruct32

This struct provides information about the last eye event that has been calculated. The difference to EventStruct is that the timestamp will be stored in milliseconds instead of microseconds and the order

of the components are different. To update information in EventStruct32 use function iV_GetEvent32.

Data Fields

double	duration	duration of the event [milliseconds]
double	endTime	end time of the event [milliseconds]
char	eventType	type of eye event, 'F' for fixation (only fixations are supported)
char	eye	related eye, 'l' for left eye, 'r' for right eye
double	positionX	horizontal position of the fixation event [pixel]
double	positionY	vertical position of the fixation event [pixel]
double	startTime	start time of the event [milliseconds]

struct EyeDataStruct

This struct provides numerical information about eye data. EyeDataStruct is part of SampleStruct. To update information in SampleStruct use function iV_GetSample or set the sample callback with iV_Set↩SampleCallback.

Data Fields

double	diam	pupil diameter [mm]
double	eyePositionX	horizontal eye position relative to camera [mm]
double	eyePositionY	vertical eye position relative to camera [mm]
double	eyePositionZ	distance to camera [mm]
double	gazeX	horizontal gaze position on screen [pixel]
double	gazeY	vertical gaze position on screen [pixel]

struct EyePositionStruct

This value represents the relative position of the eye in the tracking box. The 0 is defined at the center position. The value +1 defines the upper/right/far maximum while the value -1 the lower/left/near maximum. The position rating is related to the tracking monitor and represents how critical the tracking and the position is, related to the border of the tracking box. The 0 is defined as the best eye position to be tracked while the value +1 defines that the eye is almost not being tracked due to extreme upper/right/far position. The value -1 defines that the eye is almost not being tracked due to extreme lower/left/near position. If the eye isn't tracked at all the validity flag goes to 0 and all values for the represented eye will be set to 0.

Data Fields

double	position↩RatingX	horizontal rating [-1; +1]
double	position↩RatingY	vertical rating [-1; +1]

double	positionRatingZ	distance rating [-1; +1]
double	relativePositionX	horizontal position [-1; +1]
double	relativePositionY	vertical position [-1; +1]
double	relativePositionZ	depth/distance position [-1; +1]
int	validity	confidence of position and rating values [0; 1]

struct ImageStruct

Use this struct to get raw eye image, raw scene video image, or raw tracking monitor image. For receiving raw eye image (format: monochrome 8bpp) use iV_GetEyeImage, or set the eye image callback with iV_SetEyeImageCallback. For receiving raw scene video image (format: RGB 24bpp) use iV_GetSceneVideo, or set the scene video callback with iV_SetSceneVideoCallback. For receiving raw tracking monitor image (format: RGB 24bpp) use iV_GetTrackingMonitor, or set the tracking monitor callback with iV_SetTrackingMonitorCallback.

Data Fields

char *	imageBuffer	pointer to image data
int	imageHeight	vertical size of the image [pixel]
int	imageSize	image data size [byte]
int	imageWidth	horizontal size of the image [pixel]

struct REDGeometryStruct

Use this struct to customize the RED and RED-m geometry. See chapter Setting up RED and RED-m Geometry in the iView X SDK Manual for details. For setting up the RED or RED-m geometry parameters with REDGeometryStruct use function iV_SetREDGeometry.

Data Fields

int	monitorSize	monitor size [inch] can be set to 19 or 22 used if redGeometry is set to monitorIntegrated only applicable for RED devices only
enum REDGeometryEnum	redGeometry	defines which parameter is used.
int	redHeightOverFloor	distance floor to RED [mm] used if redGeometry is set to standalone only applicable for RED only
int	redInclAngle	RED or RED-m inclination angle [degree] used if redGeometry is set to standalone only applicable for RED and RED-m devices

int	redStimDist	distance RED to stimulus screen [mm] used if redGeometry is set to standalone only applicable for RED only
int	redStimDist↩ Depth	horizontal distance RED-m to stimulus screen [mm] used if red↩ Geometry is set to standalone only applicable for RED-m only
int	redStimDist↩ Height	vertical distance RED-m to stimulus screen [mm] used if red↩ Geometry is set to standalone only applicable for RED-m only
char	setup↩ Name[256]	name of the profile used if redGeometry is set to standalone only applicable for RED and RED-m devices
int	stimHeight↩ OverFloor	distance floor to stimulus screen [mm] used if redGeometry is set to standalone only applicable for RED only
int	stimX	horizontal stimulus calibration size [mm] used if redGeometry is set to standalone only applicable for RED and RED-m devices
int	stimY	vertical stimulus calibration size [mm] used if redGeometry is set to standalone only applicable for RED and RED-m devices

struct SampleStruct

This struct provides information about an eye data sample. To update information in SampleStruct use the function iV_GetSample or set the sample callback with iV_SetSampleCallback.

Data Fields

struct EyeDataStruct	leftEye	stores information of the left eye (see EyeDataStruct for more information)
int	planeNumber	plane number of gaze data sample (only for HED HT)
struct EyeDataStruct	rightEye	stores information of the right eye (see EyeDataStruct for more information)
long long	timestamp	timestamp of current gaze data sample [microseconds]

struct SampleStruct32

This struct provides information about a eye data samples. To update information in SampleStruct32 use the function iV_GetSample32. The difference to SampleStruct is that the timestamp will be stored in milliseconds instead of microseconds.

Data Fields

struct EyeDataStruct	leftEye	stores information of the left eye (see EyeDataStruct for more information)
int	planeNumber	plane number of gaze data sample
struct EyeDataStruct	rightEye	stores information of the right eye (see EyeDataStruct for more information)
double	timestamp	timestamp of current gaze data sample [milliseconds]

struct SystemInfoStruct

This struct provides information about the iView X (eyetracking-server) version and the API version in use. To update data in SystemInfoStruct use the function iV_GetSystemInfo.

Data Fields

int	API_Buildnumber	build number of iView X SDK in use
int	API_MajorVersion	major version number of iView X SDK in use
int	API_MinorVersion	minor version number of iView X SDK in use
int	iV_Buildnumber	build number of iView X (eyetracking-server) in use
enum ETDevice	iV_ETDevice	type of eye tracking device
int	iV_MajorVersion	major version number of iView X (eyetracking-server) in use
int	iV_MinorVersion	minor version number of iView X (eyetracking-server) in use
int	samplerate	sample rate of eye tracking device in use

struct TrackingStatusStruct

This struct provides information about the relative eye ball position within the tracking box. The information will be provided for each eye individually as well as for the geographical center between both eyes. To update information in TrackingStatusStruct use the function iV_GetTrackingStatus.

Data Fields

struct EyePositionStruct	leftEye	stores information of the left eye (see EyePositionStruct for more information)
struct EyePositionStruct	rightEye	stores information of the right eye (see EyePositionStruct for more information)
long long	timestamp	timestamp of current tracking status sample [microseconds]
struct EyePositionStruct	total	stores information of the geometric average of both eyes (see EyePositionStruct for more information)

Enumeration Type Documentation

enum CalibrationStatusEnum

This enum provides information about the eyetracking-server calibration status. If the device is not calibrated the eyetracking-server won't deliver valid gaze data. Use the functions iV_GetCalibration↩Status to retrieve the calibration status and iV_Calibrate to perform a calibration.

Enumerator

calibrationUnknown calibration status is unknown (i.e. if the connection is not established)

calibrationInvalid the device is not calibrated and will not deliver valid gaze data

calibrationValid the device is calibrated and will deliver valid gaze data

calibrationInProgress the device is currently performing a calibration

enum ETApplication

ETApplication can be used to start iView X or iView X OEM (eyetracking-server) application dependent to the used eye tracking device. Set this as a parameter in iV_Start function.

Enumerator

iViewX for iView X based devices like RED, HiSpeed, MRI, HED

iViewXOEM for RED-OEM based devices like RED-m or other customized RED-OEM devices

enum ETDevice

The enumeration ETDevice can be used in connection with iV_GetSystemInfo to get information about which type of device is connected to iView X or eyetracking-server. It is part of the SystemInfoStruct.

Enumerator

NONE if no device is set up while running iView X application

RED iView X based remote eye tracking devices

REDm eyetracking-server based remote eye tracking devices

HiSpeed iView X based hi speed eye tracking devices

MRI iView X based MRI eye tracking devices

HED iView X based head mounted eye tracking devices

Custom iView X based custom devices like the mouse grabber

enum FilterAction

FilterType can be used to select the action that is performed when calling iV_ConfigureFilter.

Enumerator

Query query the current filter status

Set configure filter parameters

enum FilterType

FilterType can be used to select the filter that is used with iV_ConfigureFilter.

Enumerator

Average left and right gaze data channels are averaged the type of the parameter data from iV_ConfigureFilter has to be converted to int* The value of data can be [0;1] where 0 means averaging is disabled and 1 means averaging is enabled

enum REDGeometryEnum

uses to the define the content of REDGeometryStruct

Enumerator

monitorIntegrated use monitor integrated mode

standalone use standalone mode

2.2 Functions

Functions

- int iV_AbortCalibration ()
- int iV_AcceptCalibrationPoint ()
- int iV_Calibrate ()
- int iV_ChangeCalibrationPoint (int number, int positionX, int positionY)
- int iV_ClearAOI ()
- int iV_ClearRecordingBuffer ()
- int iV_ConfigureFilter (FilterType filter, FilterAction action, void *data)
- int iV_Connect (char *sendIPAddress, int sendPort, char *recvIPAddress, int receivePort)
- int iV_ConnectLocal ()
- int iV_ContinueEyetracking ()
- int iV_ContinueRecording (char *etMessage)
- int iV_DefineAOI (AOIStruct *aoiData)
- int iV_DefineAOIPort (int port)
- int iV_DeleteREDGeometry (char *setupName)
- int iV_DisableAOI (char *aoiName)
- int iV_DisableAOIGroup (char *aoiGroup)
- int iV_DisableGazeDataFilter ()
- int iV_DisableProcessorHighPerformanceMode ()
- int iV_Disconnect ()
- int iV_EnableAOI (char *aoiName)
- int iV_EnableAOIGroup (char *aoiGroup)
- int iV_EnableGazeDataFilter ()
- int iV_EnableProcessorHighPerformanceMode ()
- int iV_GetAccuracy (AccuracyStruct *accuracyData, int visualization)
- int iV_GetAccuracyImage (ImageStruct *imageData)
- int iV_GetAOIOutputValue (int *aoiOutputValue)
- int iV_GetCalibrationParameter (CalibrationStruct *calibrationData)
- int iV_GetCalibrationPoint (int calibrationPointNumber, CalibrationPointStruct *calibrationPoint)
- int iV_GetCalibrationStatus (CalibrationStatusEnum *calibrationStatus)
- int iV_GetCurrentCalibrationPoint (CalibrationPointStruct *currentCalibrationPoint)
- int iV_GetCurrentREDGeometry (REDGeometryStruct *redGeometry)
- int iV_GetCurrentTimestamp (long long *currentTimestamp)
- int iV_GetDeviceName (char deviceName[64])
- int iV_GetEvent (EventStruct *eventDataSample)
- int iV_GetEvent32 (EventStruct32 *eventDataSample)
- int iV_GetEyeImage (ImageStruct *imageData)
- int iV_GetFeatureKey (long long *featureKey)
- int iV_GetGeometryProfiles (int maxSize, char *profileNames)
- int iV_GetLicenseDueDate (DateStruct *licenseDueDate)

- int iV_GetREDGeometry (char *profileName, REDGeometryStruct *redGeometry)
- int iV_GetSample (SampleStruct *rawDataSample)
- int iV_GetSample32 (SampleStruct32 *rawDataSample)
- int iV_GetSceneVideo (ImageStruct *imageData)
- int iV_GetSerialNumber (char serialNumber[64])
- int iV_GetSystemInfo (SystemInfoStruct *systemInfoData)
- int iV_GetTrackingMonitor (ImageStruct *imageData)
- int iV_GetTrackingStatus (TrackingStatusStruct *trackingStatus)
- int iV_GetUseCalibrationKeys (int *enableKeys)
- int iV_HideAccuracyMonitor ()
- int iV_HideEyeImageMonitor ()
- int iV_HideSceneVideoMonitor ()
- int iV_HideTrackingMonitor ()
- int iV_IsConnected ()
- int iV_LoadCalibration (char *name)
- int iV_Log (char *logMessage)
- int iV_PauseEyetracking ()
- int iV_PauseRecording ()
- int iV_Quit ()
- int iV_ReleaseAOIPort ()
- int iV_RemoveAOI (char *aoiName)
- int iV_ResetCalibrationPoints ()
- int iV_SaveCalibration (char *name)
- int iV_SaveData (char *filename, char *description, char *user, int overwrite)
- int iV_SelectREDGeometry (char *profileName)
- int iV_SendCommand (char *etMessage)
- int iV_SendImageMessage (char *etMessage)
- int iV_SetAOIHitCallback (pDLLSetAOIHit pAOIHitCallbackFunction)
- int iV_SetCalibrationCallback (pDLLSetCalibrationPoint pCalibrationCallbackFunction)
- int iV_SetConnectionTimeout (int time)
- int iV_SetEventCallback (pDLLSetEvent pEventCallbackFunction)
- int iV_SetEventDetectionParameter (int minDuration, int maxDispersion)
- int iV_SetEyeImageCallback (pDLLSetEyeImage pEyeImageCallbackFunction)
- int iV_SetLicense (const char *licenseKey)
- int iV_SetLogger (int logLevel, char *filename)
- int iV_SetREDGeometry (REDGeometryStruct *redGeometry)
- int iV_SetResolution (int stimulusWidth, int stimulusHeight)
- int iV_SetSampleCallback (pDLLSetSample pSampleCallbackFunction)
- int iV_SetSceneVideoCallback (pDLLSetSceneVideo pSceneVideoCallbackFunction)
- int iV_SetTrackingMonitorCallback (pDLLSetTrackingMonitor pTrackingMonitorCallbackFunction)
- int iV_SetTrackingParameter (int ET_PARAM_EYE, int ET_PARAM, int value)
- int iV_SetupCalibration (CalibrationStruct *calibrationData)
- int iV_SetUseCalibrationKeys (int enableKeys)

- int iV_ShowAccuracyMonitor ()
- int iV_ShowEyeImageMonitor ()
- int iV_ShowSceneVideoMonitor ()
- int iV_ShowTrackingMonitor ()
- int iV_Start (enum ETApplication etApplication)
- int iV_StartRecording ()
- int iV_StopRecording ()
- int iV_TestTTL (int value)
- int iV_Validate ()

Detailed Description

Function Documentation

int iV_AbortCalibration ( )

Aborts a calibration or validation if one is in progress. If the calibration or validation function is visualizing the calibration area the iV_Calibrate or iV_Validate function will return with RET_CALIBRATION_ABORTED. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_CALIBRATION_NOT_IN_PROGRESS	calibration or validation is not in progress
ERR_NOT_CONNECTED	no connection established

int iV_AcceptCalibrationPoint ( )

Accepts a calibration or validation point if the calibration or validation is in progress. The participant needs to be tracked and has to fixate the calibration or validation point. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_CALIBRATION_NOT_IN_PROGRESS	calibration or validation is not in progress

ERR_NOT_CONNECTED	no connection established

int iV_Calibrate ( )

Starts a calibration process. To proceed, the participant needs to be tracked and has to fixate the calibration point. Depending on the calibration settings (which can be changed using iV_SetupCalibration and by iV_SetUseCalibrationKeys) the user can accept the calibration points manually (by pressing SPACE or calling iV_AcceptCalibrationPoint) or abort the calibration (by pressing ESC or calling iV_AbortCalibration)
If the calibration is visualized by the API (CalibrationStruct::visualization is set to 1) the function won't return until the calibration has been finished (closed automatically) or aborted.
If the CalibrationStruct::visualization is set to 0, iV_Calibrate returns immediately. The user has to care about the visualization of calibration points. Information about the current calibration point can be retrieved with iV_GetCurrentCalibrationPoint or with setting up the calibration callback using iV_SetCalibrationCallback.
See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_CALIBRATION_ABORTED	calibration was aborted during progress
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected
ERR_WRONG_CALIBRATION_METHOD	eye tracking device required for this calibration method is not connected

int iV_ChangeCalibrationPoint (int *number,* int *positionX,* int *positionY*)

Changes the position of a calibration point. This has to be done before the calibration process is started. The parameter number refers to the calibration method used. If this function is used with a RED or RED-m device, the change is applied to the currently selected profile. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

number	selected calibration point
positionX	new X position on screen [pixel]
positionY	new Y position on screen [pixel]

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range

int iV_ClearAOI ( )

Removes all trigger AOIs. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_AOI_ACCESS	failed to access AOI data

int iV_ClearRecordingBuffer ( )

Clears the recorded data buffer. The recording buffer needs to be stopped using "iV_StopRecording" before it can be cleared. If you are using an "HED", the scene video buffer is cleared, too. See also iV_ClearRecordingBuffer, iV_ContinueRecording, iV_PauseRecording, iV_SaveData, iV_SendImageMessage, iV_StartRecording, iV_StopRecording.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected
ERR_EMPTY_DATA_BUFFER	recording buffer is empty
ERR_RECORDING_DATA_BUFFER	recording is activated
ERR_PAUSED_DATA_BUFFER	recording buffer is paused

int iV_ConfigureFilter (FilterType *filter*, FilterAction *action*, void * *data*)

Queries or sets filter parameters. The usage of the parameter data depends on the parameter type.

Parameters

filter	filter type which will be is configured. See FilterType
action	type of action to modify the corresponding filter. See FilterAction
data	a void pointer that can be casted to a data type depending on filter type. Please refer to FilterType for details. Content of the parameter depends on filter action, see FilterType. FilterAction::Query data is filled with current filter settings. FilterAction::Set data is passed to cofigure the filter

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range
ERR_WRONG_IVIEWX_VERSION	wrong version of iView X (eyetracking-server)

int iV_Connect (char * *sendIPAddress*, int *sendPort*, char * *recvIPAddress*, int *receivePort*)

Establishes a connection to iView X (eyetracking-server). iV_Connect will not return until a connection has been established. If no connection can be established, the function will return after the time span defined by iV_SetConnectionTimeout. Default time span is 3 seconds. See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Parameters

sendIPAddress	IP address of iView X (eyetracking-server) computer
sendPort	port being used by iView X SDK for sending data to iView X (eyetracking-server)
recvIPAddress	IP address of local computer
receivePort	port being used by iView X SDK for receiving data from iView X (eyetracking-server)

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_EYETRACKING_APPLICATION_NOT_RUNNING	no eye tracking application running

ERR_WRONG_PARAMETER	parameter out of range
ERR_COULD_NOT_CONNECT	failed to establish connection

int iV_ConnectLocal ( )

Establishes a connection to eyetracking-server only. iV_ConnectLocal will not return until a connection has been established. If no connection can be established the function will return after the time span defined by iV_SetConnectionTimeout. Default time span is 3 seconds.
iV_ConnectLocal can only connect with RED-m or RED-OEM devices plugged in to the same PC. See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_EYETRACKING_APPLICATION_NOT_RUNNING	no eye tracking application running
ERR_COULD_NOT_CONNECT	failed to establish connection

int iV_ContinueEyetracking ( )

Wakes up and enables the eye tracking application from suspend mode to continue processing gaze data. The application can be set to suspend mode by calling iV_PauseEyetracking.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

int iV_ContinueRecording (char * *etMessage*)

Continues gaze data recording. A HED video recording can neither be paused nor continued. iV_ContinueRecording does not return until gaze recording is continued. Before it can be continued, the data needs to be paused using. iV_PauseRecording. Additionally this function allows a message to be stored inside the idf data buffer. See also iV_ClearRecordingBuffer, iV_ContinueRecording, iV_PauseRecording, iV_SaveData, iV_SendImageMessage, iV_StartRecording, iV_StopRecording.

Parameters

etMessage	text message that will be written to data file

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected
ERR_EMPTY_DATA_BUFFER	recording buffer is empty

int iV_DefineAOI (AOIStruct * *aoiData*)

Defines an AOI. The API can handle up to 20 AOIs. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

aoiData	see reference information for AOIStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_DefineAOIPort (int *port*)

Selects a port for sending out TTL trigger. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

port	port address

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

ERR_COULD_NOT_OPEN_PORT	failed to open port

int iV_DeleteREDGeometry (char * *setupName*)

Deletes the RED-m geometry setup with the given name. It is not possible to delete a geometry profile if it is currently in use. See chapter Setting up RED and RED-m Geometry in the iView X SDK Manual.

Parameters

setupName	name of the geometry setup which will be deleted

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_DisableAOI (char * *aoiName*)

Disables all AOIs with the given name. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

aoiName	name of the AOI which will be disabled

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no data available
ERR_AOI_ACCESS	failed to access AOI data

int iV_DisableAOIGroup (char * *aoiGroup*)

Disables an AOI group. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

aoiGroup	name of the AOI group which will be disabled

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no data available
ERR_AOI_ACCESS	failed to access AOI data

int iV_DisableGazeDataFilter ( )

Disables the raw data filter. The gaze data filter can be enabled using iV_EnableGazeDataFilter.

Return values

RET_SUCCESS	intended functionality has been fulfilled

int iV_DisableProcessorHighPerformanceMode ( )

Disables a CPU high performance mode allowing the CPU to reduce the performance. See also iV_EnableProcessorHighPerformanceMode.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

int iV_Disconnect ( )

Disconnects from iView X (eyetracking-server). iV_Disconnect will not return until the connection has been disconnected. After this function has been called no other function or device can communicate with iView X (eyetracking-server). See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_DELETE_SOCKET	failed to delete sockets

int iV_EnableAOI (char * *aoiName*)

Enables all AOIs with the given name. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

aoiName	name of the AOI which will be enabled

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no data available
ERR_AOI_ACCESS	failed to access AOI data

int iV_EnableAOIGroup (char * *aoiGroup*)

Enables an AOI group See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

aoiGroup	name of the AOI group which will be enabled

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no data available
ERR_AOI_ACCESS	failed to access AOI data

int iV_EnableGazeDataFilter ( )

Enables a gaze data filter. This API bilateral filter was implemented due to special HCI application requirements. The gaze data filter can be disabled using iV_DisableGazeDataFilter.

Return values

RET_SUCCESS	intended functionality has been fulfilled

int iV_EnableProcessorHighPerformanceMode ( )

Enables a CPU high performance mode to prevent the CPU from reducing the performance. See also iV_DisableProcessorHighPerformanceMode.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

int iV_GetAccuracy (AccuracyStruct * *accuracyData*, int *visualization*)

Updates AccuracyStruct accuracyData with validated accuracy results. Before accuracy data is accessible the accuracy needs to be validated with iV_Validate. If the parameter `visualization` is set to `1` the accuracy data will be visualized in a dialog window. See also iV_GetAccuracy, iV_GetAccuracyImage, iV_HideAccuracyMonitor, iV_ShowAccuracyMonitor, iV_Validate and the chapter Running a Validation in the iView X SDK Manual.

Parameters

accuracyData	see reference information for AccuracyStruct
visualization	0: no visualization 1: accuracy data will be visualized in a dialog window

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_NOT_VALIDATED	system is not validated
ERR_WRONG_PARAMETER	parameter out of range

int iV_GetAccuracyImage (ImageStruct * *imageData*)

Updates `imageData` struct with drawn accuracy results. Before accuracy data is accessible the accuracy needs to be validated with iV_Validate. See also iV_GetAccuracy, iV_GetAccuracyImage, iV_HideAccuracyMonitor, iV_ShowAccuracyMonitor, iV_Validate and the chapter Running a Validation in the iView X SDK Manual.

Parameters

imageData	see reference information for ImageStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_NOT_VALIDATED	system is not validated

int iV_GetAOIOutputValue (int * *aoiOutputValue*)

Gives back the AOI value See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

aoiOutputValue	provides the AOI output value

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

int iV_GetCalibrationParameter (CalibrationStruct * *calibrationData*)

Updates stored `calibrationData` information with currently selected parameters. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

calibrationData	see reference information for CalibrationStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

int iV_GetCalibrationPoint (int *calibrationPointNumber,* CalibrationPointStruct * *calibrationPoint*)

Delivers information about a calibration point. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

calibrationPointNumber	number of requested calibration point
calibrationPoint	information of requested calibration point, stored in CalibrationPointStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range

int iV_GetCalibrationStatus (CalibrationStatusEnum * *calibrationStatus*)

Updates `calibrationStatus` information. The client needs to be connected to the eyetracking-server. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

calibrationStatus	see reference information for CalibrationStatusEnum

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established

int iV_GetCurrentCalibrationPoint (CalibrationPointStruct * currentCalibrationPoint)

Updates data in CalibrationPointStruct `currentCalibrationPoint` with current calibration point data. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

currentCalibrationPoint	information of requested calibration point, stored in CalibrationPointStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established

int iV_GetCurrentREDGeometry (REDGeometryStruct * redGeometry)

Gets the currently loaded RED geometry. See chapter Setting up RED and RED-m Geometry in the iView X SDK Manual and iV_DeleteREDGeometry, iV_GetCurrentREDGeometry, iV_GetGeometryProfiles, iV_GetREDGeometry, iV_SelectREDGeometry, iV_SetREDGeometry.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_GetCurrentTimestamp (long long * currentTimestamp)

Provides the current eye tracker timestamp in microseconds. See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback.

Parameters

currentTimestamp	requested time stamp

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established

int iV_GetDeviceName (char *deviceName[64]*)

Updated the device name information of the connected device.

Parameters

deviceName	the name of the requested device

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_IVIEWX_VERSION	wrong version of iView X (eyetracking-server)

int iV_GetEvent (EventStruct * *eventDataSample*)

Updates data from EventStruct `eventDataSample` with current event data. See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback.

Parameters

eventDataSample	see reference information for EventStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established

int iV_GetEvent32 (EventStruct32 * *eventDataSample*)

Updates data from EventStruct32 `eventDataSample` with current event data. See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback.

Parameters

eventDataSample	see reference information for EventStruct32

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established

int iV_GetEyeImage (ImageStruct * *imageData*)

Updates `imageData` with current eye image.

Parameters

imageData	see reference information for ImageStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_GetFeatureKey (long long * *featureKey*)

Gets the device specific feature key. Used for RED-OEM devices only.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

ERR_WRONG_IVIEW↩ X_VERSION	wrong version of iView X (eyetracking-server)

int iV_GetGeometryProfiles (int *maxSize*, char * *profileNames*)

Gets all available profiles by name. They will be written comma-separated in the char buffer. The user needs to be sure that the buffer is not smaller than the needed buffer length. See chapter Setting up RED and RED-m Geometry in the iView X SDK Manual and iV_DeleteREDGeometry, iV_GetCurrent↩ REDGeometry, iV_GetGeometryProfiles, iV_GetREDGeometry, iV_SelectREDGeometry, iV_SetRE↩ DGeometry.

Parameters

maxSize	the length of the string profileNames
profileNames	an empty string where profile names will be put in

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECT↩ ED	no connection established
ERR_WRONG_PARA↩ METER	parameter out of range
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_GetLicenseDueDate (DateStruct * *licenseDueDate*)

Gets the system license expiration date. The license will not expire if the license is set to 00.00.0000. See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDue↩ Date, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_↩ SetConnectionTimeout, iV_SetLicense, iV_Start.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECT↩ ED	no connection established

int iV_GetREDGeometry (char * *profileName,* REDGeometryStruct * *redGeometry*)

Gets the geometry data of a requested profile without selecting them. See chapter Setting up RED and RED-m Geometry in the iView X SDK Manual and iV_DeleteREDGeometry, iV_GetCurrentREDGeometry, iV_GetGeometryProfiles, iV_GetREDGeometry, iV_SelectREDGeometry, iV_SetREDGeometry.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_GetSample (SampleStruct * *rawDataSample*)

Updates data in SampleStruct `rawDataSample` with current eye tracking data. See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback.

Parameters

rawDataSample	see reference information for SampleStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established

int iV_GetSample32 (SampleStruct32 * *rawDataSample*)

Updates data in SampleStruct32 `rawDataSample` with current eye tracking data sample. See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback.

Parameters

rawDataSample	see reference information for SampleStruct32

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established

int iV_GetSceneVideo (ImageStruct * *imageData*)

Updates ImageStruct `imageData` with current scene video image. This functions is available for HED only.

Parameters

imageData	see reference information for ImageStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_GetSerialNumber (char *serialNumber[64]*)

Updated the serial number information of the connected device. See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Parameters

serialNumber	the serial number of the requested device

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no data available
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_IVIEWX_VERSION	wrong version of iView X

int iV_GetSystemInfo (SystemInfoStruct * *systemInfoData*)

Updates SystemInfoStruct systemInfoData with current system information. See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Parameters

systemInfoData	see reference information for SystemInfoStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no data available

int iV_GetTrackingMonitor (ImageStruct * *imageData*)

Updates ImageStruct `imageData` with current tracking monitor image.

Parameters

imageData	see reference information for ImageStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_GetTrackingStatus (TrackingStatusStruct * *trackingStatus*)

Updates TrackingStatusStruct `trackingStatus` with current tracking status.

Parameters

trackingStatus	see reference information for TrackingStatusStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no new data available

ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_HideAccuracyMonitor ( )

Hides accuracy monitor window which can be opened by iV_ShowAccuracyMonitor.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_WINDOW_IS_CLOSED	window is closed
ERR_NOT_CONNECTED	no connection established

int iV_HideEyeImageMonitor ( )

Hides eye image monitor window which can be opened by iV_ShowEyeImageMonitor.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_WINDOW_IS_CLOSED	window is closed
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_HideSceneVideoMonitor ( )

Hides scene video monitor window which can be opened by iV_ShowSceneVideoMonitor.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_WINDOW_IS_CLOSED	window is closed
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_HideTrackingMonitor ( )

Hides tracking monitor window which can be opened by iV_ShowTrackingMonitor.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_WINDOW_IS_CLOSED	window is closed
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_IsConnected ( )

Checks if connection to iView X (eyetracking-server) is still established. See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

int iV_LoadCalibration (char * *name*)

Loads a previously saved calibration. A calibration has to be saved by using iV_SaveCalibration. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

name	calibration name / identifier

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_IVIEWX_VERSION	wrong version of iView X

ERR_WRONG_DEVICE	eye tracking device required for this function is not connected
ERR_WRONG_PARAMETER	parameter out of range

int iV_Log (char * *logMessage*)

Writes `logMessage` into log file.

Parameters

logMessage	message that shall be written to the log file

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_ACCESS_TO_FILE	failed to access log file

int iV_PauseEyetracking ( )

Suspends the eye tracking application and disables calculation of gaze data. The application can be reactivated by calling iV_ContinueEyetracking. See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

int iV_PauseRecording ( )

Pauses gaze data recording. A HED video recording can neither be paused nor continued. iV_PauseRecording does not return until gaze recording is paused. See also iV_ClearRecordingBuffer, iV_ContinueRecording, iV_PauseRecording, iV_SaveData, iV_SendImageMessage, iV_StartRecording, iV_StopRecording.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

ERR_WRONG_DEVICE	eye tracking device required for this function is not connected
ERR_EMPTY_DATA_BUFFER	recording buffer is empty
ERR_FULL_DATA_BUFFER	data buffer is full

int iV_Quit ( )

Disconnects and closes iView X (eyetracking-server). After this function has been called no other function or application can communicate with iView X (eyetracking-server). See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_DELETE_SOCKET	failed to delete sockets
ERR_WRONG_IVIEWX_VERSION	wrong version of iView X

int iV_ReleaseAOIPort ( )

Releases the port for sending TTL trigger. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Return values

RET_SUCCESS	intended functionality has been fulfilled

int iV_RemoveAOI (char * *aoiName*)

Removes all AOIs with the given name. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

aoiName	name of the AOI which will be removed

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_NO_VALID_DATA	no data available
ERR_AOI_ACCESS	failed to access AOI data

int iV_ResetCalibrationPoints ( )

Resets all calibration points to its default position. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_SaveCalibration (char * *name*)

Saves a calibration with a custom name. To save a calibration it is required that a successful calibration already has been completed. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

name	calibration name / identifier

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_IVIEWX_VERSION	wrong version of iView X
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

ERR_WRONG_PARAMETER	parameter out of range

int iV_SaveData (char * *filename*, char * *description*, char * *user*, int *overwrite*)

Writes recorded data buffer to disc. The data recording needs to be stopped using iV_StopRecording before the data buffer can be saved to given location. The `filename` can include the path. If the connected eye tracking device is a HED, scene video buffer is written, too. iV_SaveData will not return until the data has been saved.

Parameters

filename	filename (incl. path) of data files being created (.idf: eyetracking data, .avi: scene video data) If no path is provided, eyetracking server tries to save the file into it's binary path, e.g. `C:\Program Files\SMI\iViewREDm`.
description	Optional experiment description tag stored in the idf file. This tag is available in BeGaze and in the text export from an idf file.
user	Optional name of test person. This tag is available in BeGaze and in the text export from an idf file.
overwrite	Overwriting policy. 0: do not overwrite file `filename` if it already exists 1: overwrite file `filename` if it already exists

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range
ERR_EMPTY_DATA_BUFFER	recording buffer is empty
ERR_RECORDING_DATA_BUFFER	recording is activated
ERR_PAUSED_DATA_BUFFER	recording buffer is paused

int iV_SelectREDGeometry (char * *profileName*)

Selects a predefined geometry profile. See chapter Setting up RED and RED-m Geometry in the iView X SDK Manual and iV_DeleteREDGeometry, iV_GetCurrentREDGeometry, iV_GetGeometryProfiles, iV_GetREDGeometry, iV_SelectREDGeometry, iV_SetREDGeometry.

Parameters

profileName	name of the selected profile which should be selected

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_SendCommand (char * *etMessage*)

Sends a remote command to iView X (eyetracking-server). Please refer to the iView X help file for further information about remote commands. Important Note: This function is temporary and will not be supported in subsequent versions. See also iV_ClearRecordingBuffer, iV_ContinueRecording, iV_PauseRecording, iV_SaveData, iV_SendImageMessage, iV_StartRecording, iV_StopRecording.

Parameters

etMessage	iView X remote command

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range

int iV_SendImageMessage (char * *etMessage*)

Sends a text message to iView X idf recording data file. If the `etMessage` has the suffix ".jpg", ".bmp", ".png", or ".avi" BeGaze will separate the data buffer automatically into according trials. See also iV_ClearRecordingBuffer, iV_ContinueRecording, iV_PauseRecording, iV_SaveData, iV_SendImageMessage, iV_StartRecording, iV_StopRecording.

Parameters

etMessage	Any text message to separate trials (image name containing extensions) or any idf data marker

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

int iV_SetAOIHitCallback (pDLLSetAOIHit *pAOIHitCallbackFunction*)

Sets a callback function for the AOI hit functions. The function will be called if the iView X (eyetracking-server) has calculated an AOI hit. For usage of this function AOI's needs to be defined. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Important note: Algorithms with high processor usage and long calculation time shouldn't run within this callback due to a higher probability of data loss See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback

Parameters

pAOIHitCallbackFunction	pointer to AOIHitCallbackFunction

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetCalibrationCallback (pDLLSetCalibrationPoint *pCalibrationCallbackFunction*)

Sets a callback function for the calibration and validation process. The callback function will be called after a calibration or validation point was moved and needs to be redrawn on the new position, or if the calibration or validation was finished successfully or unsuccessfully. It's possible to disable the callback by hand over NULL as a parameter. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

pCalibrationCallbackFunction	pointer to CalibrationCallbackFunction

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetConnectionTimeout (int *time*)

Defines a customized timeout for how long iV_Connect tries to connect to iView X (eyetracking-server). See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Parameters

time	the time [sec] iV_Connect is waiting for iView X response

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetEventCallback (pDLLSetEvent *pEventCallbackFunction*)

Sets a callback function for the event data. The function will be called if a real-time detected fixation has been started or ended. Important note: Algorithms with high processor usage and long calculation time shouldn't run within this callback due to a higher probability of data loss. See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback.

Parameters

pEventCallbackFunction	pointer to EventCallbackFunction

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetEventDetectionParameter (int *minDuration*, int *maxDispersion*)

Defines the detection parameter for online fixation detection algorithm. See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback.

Parameters

minDuration	minimum fixation duration [ms]
maxDispersion	maximum dispersion [pixel] for head tracking systems or [deg] for non head tracking systems

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetEyeImageCallback (pDLLSetEyeImage *pEyeImageCallbackFunction*)

Sets a callback function for the eye image data. The function will be called if a new eye image is available. The image format is monochrome 8bpp. Important note: Algorithms with high processor usage and long calculation time shouldn't run within this callback due to a higher probability of data loss.

Parameters

pEyeImageCallbackFunction	pointer to EyeImageCallbackFunction

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetLicense (const char * *licenseKey*)

Validates the customer license (only for OEM devices). See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Parameters

licenseKey	provided license key

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetLogger (int *logLevel*, char * *filename*)

Defines the logging behavior of iView X SDK.

Parameters

logLevel	see Logging Level in subsection Explanations for Defines in this manual for further information
filename	filename of log file

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range
ERR_ACCESS_TO_FILE	failed to access log file

int iV_SetREDGeometry (REDGeometryStruct * *redGeometry*)

Define the RED and RED-m stand alone and monitor integrated geometry. See chapter Setting up RED and RED-m Geometry in the iView X SDK Manual and iV_DeleteREDGeometry, iV_GetCurrentREDGeometry, iV_GetGeometryProfiles, iV_GetREDGeometry, iV_SelectREDGeometry, iV_SetREDGeometry for details.

Parameters

redGeometry	see reference information for REDGeometryStruct

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established

ERR_WRONG_PARAMETER	parameter out of range
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_SetResolution (int *stimulusWidth*, int *stimulusHeight*)

iV_SetResolution function defines a fixed resolution independent to the screen resolution of chosen display device defined in iV_SetupCalibration function.

Parameters

stimulusWidth	horizontal resolution of stimulus screen [pixel]
stimulusHeight	vertical resolution of stimulus screen [pixel]

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetSampleCallback (pDLLSetSample *pSampleCallbackFunction*)

Sets a callback function for the raw sample data. The function will be called if iView X (eyetracking-server) has calculated a new data sample. Important note: Algorithms with high processor usage and long calculation time shouldn't run within this callback due to a higher probability of data loss. See also iV_GetCurrentTimestamp, iV_GetEvent, iV_GetEvent32, iV_GetSample, iV_GetSample32, iV_GetTrackingStatus, iV_SetEventCallback, iV_SetEventDetectionParameter, iV_SetSampleCallback.

Parameters

pSampleCallbackFunction	pointer to SampleCallbackFunction

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetSceneVideoCallback (pDLLSetSceneVideo *pSceneVideoCallbackFunction*)

Sets a callback function for the scene video image data. The function will be called if a new scene video image is available. The image format is RGB 24bpp. Important note: Algorithms with high processor usage and long calculation time shouldn't run within this callback due to a higher probability of data loss.

Parameters

pSceneVideo↩ Callback↩ Function	pointer to SceneVideoCallbackFunction

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARA↩ METER	parameter out of range

int iV_SetTrackingMonitorCallback (pDLLSetTrackingMonitor *pTrackingMonitorCallbackFunction*)

Sets a callback function for the tracking monitor image data. The function will be called if a new tracking monitor image was calculated. The image format is RGB 24bpp. Important note: Algorithms with high processor usage and long calculation time shouldn't run within this callback due to a higher probability of data loss.

Parameters

pTracking↩ Monitor↩ Callback↩ Function	pointer to TrackingMonitorCallbackFunction

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARA↩ METER	parameter out of range

int iV_SetTrackingParameter (int *ET_PARAM_EYE,* int *ET_PARAM,* int *value*)

Sets iView X (eyetracking-server) tracking parameters. See Eye Tracking Parameter subsection and iView X (eyetracking-server) manual for further explanations.

Parameters

ET_PARAM_EYE	select specific eye
ET_PARAM	select parameter that shall be set
value	new value for selected parameter

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_PARAMETER	parameter out of range

int iV_SetupCalibration (CalibrationStruct * *calibrationData*)

Sets the calibration and validation visualization parameter. See also iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Parameters

calibrationData	see reference information for "CalibrationStruct"

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected
ERR_WRONG_CALIBRATION_METHOD	eye tracking device required for this calibration method is not connected

int iV_ShowAccuracyMonitor ( )

The validated accuracy results will be visualized in a dialog window. Before the image can be drawn the calibration needs to be performed with iV_Calibrate and validated with iV_Validate. See also iV_GetAccuracy, iV_GetAccuracyImage, iV_HideAccuracyMonitor, iV_ShowAccuracyMonitor, iV_Validate.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_WINDOW_IS_OPEN	window is open
ERR_NOT_CONNECTED	no connection established
ERR_NOT_VALIDATED	system is not validated

int iV_ShowEyeImageMonitor ( )

Visualizes eye image in a separate window while the participant will be tracked.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_WINDOW_IS_OPEN	window is open
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_ShowSceneVideoMonitor ( )

Visualizes scene video in separate window (available for HED devices only). See also iV_GetSceneVideo, iV_HideSceneVideoMonitor, iV_SetSceneVideoCallback, iV_ShowSceneVideoMonitor.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_WINDOW_IS_OPEN	window is open
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_ShowTrackingMonitor ( )

Visualizes RED tracking monitor in a separate dialog window. It shows the position of the participant related to the eye tracking device and indicates (using arrows) if the participant is not positioned in the center of the tracking head box.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_WINDOW_IS_OPEN	window is open
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

int iV_Start (enum ETApplication *etApplication*)

Starts the iView X (eyetracking-server) application. Depending on the PC, it may take several seconds to start the iView X (eyetracking-server) application. The connection needs to be established separately using iV_Connect. The connection timeout can be extended using iV_SetConnectionTimeout. See also iV_Connect, iV_ConnectLocal, iV_ContinueEyetracking, iV_Disconnect, iV_GetLicenseDueDate, iV_GetSerialNumber, iV_GetSystemInfo, iV_IsConnected, iV_PauseEyetracking, iV_Quit, iV_SetConnectionTimeout, iV_SetLicense, iV_Start.

Parameters

etApplication	the iView X (eyetracking-server) application which will be started

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_SERVER_IS_RUNNING	server is running
ERR_IVIEWX_NOT_FOUND	failed to start iView X (eyetracking-server) application
ERR_CAMERA_NOT_FOUND	failed to access device
ERR_WRONG_CAMERA	failed to access device
ERR_WRONG_CAMERA_PORT	failed to access device

int iV_StartRecording ( )

Starts gaze data recording and scene video recording (if connected eye tracking device is "HED"). iV_StartRecording does not return until gaze and scene video recording is started. The data streaming needs to be stopped by using iV_StopRecording before it can be saved using iV_SaveData. See also iV_ClearRecordingBuffer, iV_ContinueRecording, iV_PauseRecording, iV_SaveData, iV_SendImageMessage, iV_StartRecording, iV_StopRecording.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected
ERR_RECORDING_DATA_BUFFER	recording is activated

int iV_StopRecording ( )

Stops gaze data recording and scene video recording (if connected eye tracking device is "HED"). iV_StopRecording does not return until gaze and scene video recording is stopped. This function needs to be called before the data can be saved using iV_SaveData. See also iV_ClearRecordingBuffer, iV_ContinueRecording, iV_PauseRecording, iV_SaveData, iV_SendImageMessage, iV_StartRecording, iV_StopRecording.

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_NOT_CONNECTED	no connection established
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected
ERR_EMPTY_DATA_BUFFER	recording buffer is empty
ERR_FULL_DATA_BUFFER	data buffer is full

int iV_TestTTL (int *value*)

Sends a TTL value to defined port. Define a port with iV_DefineAOIPort. See also iV_ClearAOI, iV_DefineAOI, iV_DefineAOIPort, iV_DisableAOI, iV_DisableAOIGroup, iV_EnableAOI, iV_EnableAOIGroup, iV_GetAOIOutputValue, iV_ReleaseAOIPort, iV_RemoveAOI, iV_SetAOIHitCallback. iV_TestTTL.

Parameters

value	value which will be sends out as TTL signal

Return values

RET_SUCCESS	intended functionality has been fulfilled
ERR_WRONG_PARAMETER	parameter out of range

int iV_Validate ( )

Starts a validation process. To proceed, the participant needs to be tracked and has to fixate on the validation point. Depending on the validation settings (which can be changed using iV_SetupCalibration and iV_SetUseCalibrationKeys) the user can accept the validation points manually (by pressing SPACE or calling iV_AcceptCalibrationPoint) or abort the validation (by pressing ESC or calling iV_AbortCalibration). If the validation will be visualized by the API (CalibrationStruct::visualization is set to 1) the function won't return until the validation has been finished (closed automatically) or aborted (ESC). If the the CalibrationStruct::visualization is set to 0 iV_Validate returns immediately. The user has to care about the visualization of validation points. Information about the current validation point can be retrieved with iV_GetCurrentCalibrationPoint or with setting up the calibration callback using iV_SetCalibrationCallback.

See also iV_GetAccuracy, iV_GetAccuracyImage, iV_HideAccuracyMonitor, iV_ShowAccuracyMonitor, iV_Validate iV_AbortCalibration, iV_AcceptCalibrationPoint, iV_Calibrate, iV_ChangeCalibrationPoint, iV_GetCalibrationParameter, iV_GetCalibrationPoint, iV_GetCalibrationStatus, iV_GetCurrentCalibrationPoint, iV_LoadCalibration, iV_ResetCalibrationPoints, iV_SaveCalibration, iV_SetCalibrationCallback, iV_SetResolution, iV_SetupCalibration.

Return values

RET_SUCCESS	intended functionality has been fulfilled
RET_CALIBRATION_ABORTED	validation was aborted during progress
ERR_NOT_CONNECTED	no connection established
ERR_NOT_CALIBRATED	system is not calibrated
ERR_WRONG_DEVICE	eye tracking device required for this function is not connected

Index