



iView X™ SDK

v3.1.0

June 2012

Table of Contents

Introduction.....	6
System Requirements	7
Supported Eye Tracking Devices	7
Supported Programming and Scripting Languages	8
Supported Operating Systems.....	8
Function and Device Overview.....	8
Getting Started	11
Downloading	11
Running the Installer	11
Next Steps.....	12
Getting started with the SDK Examples	13
Using C#.....	14
Using MATLAB®	17
Using Python	18
Using E-Prime	19
Using NBS Presentation.....	21
iView X SDK Reference	23
Header File	23
Defines.....	23
Enumerations	23
Structs.....	23
Functions	24
Explanations for Defines.....	25
Explanations for Enumerations	26
Explanations for Data Structures.....	27
AccuracyStruct Reference	27
CalibrationPointStruct Reference.....	27
EventStruct Reference.....	27
EventStruct32 Reference.....	28
EyeDataStruct Reference.....	28
SampleStruct Reference	29
SampleStruct32 Reference	29
SystemInfoStruct Reference.....	30
CalibrationStruct Reference	30

MonitorAttachedGeometryStruct Reference	31
StandAloneModeGeometryStruct Reference	31
ImageStruct Reference	32
AOIRectangleStruct Reference	32
AOIStruct Reference	33
Function Reference	33
int iV_AbortCalibration ()	33
int iV_AcceptCalibrationPoint ()	34
int iV_Calibrate ()	34
int iV_ChangeCalibrationPoint (int number, int positionX, int positionY)	34
int iV_ClearAOI ()	35
int iV_ClearRecordingBuffer ()	35
int iV_Connect (char sendIPAddress[16], int sendPort, char recvIPAddress[16], int receivePort)	35
int iV_ContinueEyetracking ()	36
int iV_ContinueRecording (char etMessage[256])	36
int iV_DefineAOI(struct AOIStruct * aoiData)	36
int iV_DefineAOIPort(int portNumber)	37
int iV_DisableAOI (char aoiName[256])	37
int iV_DisableAOIGroup (char aoiGroup[256])	37
int iV_DisableGazeDataFilter()	37
int iV_Disconnect ()	38
int iV_EnableAOI (char aoiName[256])	38
int iV_EnableAOIGroup (char aoiGroup[256])	38
int iV_EnableGazeDataFilter()	38
int iV_GetAccuracy (struct AccuracyStruct * accuracyData, int visualization)	39
int iV_GetAccuracyImage (struct ImageStruct * imageData)	39
int iV_GetCurrentCalibrationPoint (struct CalibrationPointStruct * currentCalibrationPoint)	39
int iV_GetCurrentTimestamp (int64* currentTimestamp)	40
int iV_GetEvent (struct EventStruct * eventDataSample)	40
int iV_GetEvent32 (struct EventStruct32 * eventDataSample)	40
int iV_GetEyelImage (struct ImageStruct* image)	40
int iV_GetSample (struct SampleStruct * rawDataSample)	41
int iV_GetSample32 (struct SampleStruct32 * rawDataSample)	41
int iV_GetSceneVideo(struct ImageStruct* image)	41
int iV_GetSystemInfo (struct SystemInfoStruct * systemInfoData)	42

int iV_GetTrackingMonitor (struct ImageStruct* image).....	42
int iV_IsConnected ()	42
int iV_LoadCalibration (char name [256])	42
int iV_Log (char logMessage[256]).....	43
int iV_PauseEyetracking ().....	43
int iV_PauseRecording ()	43
int iV_Quit().....	44
int iV_ReleaseAOIPort ()	44
int iV_RemoveAOI (char aoiName[256]).....	44
int iV_ResetCalibrationPoints()	44
int iV_SaveCalibration (char name [256])	45
int iV_SaveData (char filename [256], char description [64], char user [64], int overwrite).....	45
int iV_SendCommand (char etMessage[256])	46
int iV_SendImageMessage (char etMessage[256]).....	46
void iV_SetCalibrationCallback (pDLLSetCalibrationPoint pCalibrationPoint).....	46
void iV_SetEventCallback (pDLLSetEvent pEvent)	47
int iV_SetEventDetectionParameter (int minDuration, int maxDispersion)	47
void iV_SetEyeImageCallback (pDLLSetEyeImage pEyeImage).....	47
int iV_SetLicense (char key[16])	47
int iV_SetLogger (int logLevel, char filename[256])	48
void iV_SetResolution (int stimulusWidth, int stimulusHeight).....	48
void iV_SetSampleCallback (pDLLSetSample pSample).....	48
void iV_SetSceneVideoCallback (pDLLSetSceneVideo pSceneVideo)	49
void iV_SetTrackingMonitorCallback (pDLLSetTrackingMonitor pTrackingMonitor)	49
int iV_SetTrackingParameter (int ET_PARAM_EYE, int ET_PARAM, int value)	49
int iV_SetupCalibration(struct CalibrationStruct *calibrationData).....	49
int iV_SetupMonitorAttachedGeometry (struct MonitorAttachedGeometryStruct *attachedModeGeometry)	50
int iV_SetupStandAloneMode (struct StandAloneModeGeometryStruct *standAloneModeGeometry)	50
int iV_ShowEyeImageMonitor ()	50
int iV_ShowSceneVideoMonitor().....	51
int iV_ShowTrackingMonitor ()	51
int iV_Start(int etApplication)	51
int iV_StartRecording ()	51

int iV_StopRecording ().....	52
int iV_Validate ().....	52
RED Stand Alone Mode	53
RED Monitor Attached Mode.....	54
Areas of Interest (AOI).....	55
Return Codes	56
Technical Support.....	58
License Agreement and Warranty for SDK Provided Free of Charge.....	58
1. License.....	58
2. Rights in Licensed Materials	58
3. Confidentiality	58
4. Limited Warranty and Liability	59
5. Licensee Indemnity.....	59
6. Export Restriction.....	59
7. Non-Waiver; Severability; Non-Assignment.....	59
8. Termination	59
9. Entire Agreement; Written Form Requirement.	59
10. Notices.....	60
11. Applicable Law and Jurisdiction	60
About SMI.....	61

Introduction

Welcome to the *iViewX SDK* Guide v.3.0.29!

About iViewX 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. The SDK uses UDP over Ethernet communication to provide maximum speed and minimum latency for data transfer. Additionally, the SDK supports a growing number of programming languages and environments including, but not limited to, MATLAB®, C/C++, C#, Visual Basic, E-Prime, NBS Presentation, and Python. Several example programs are provided for helping you get started with your application development.

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?

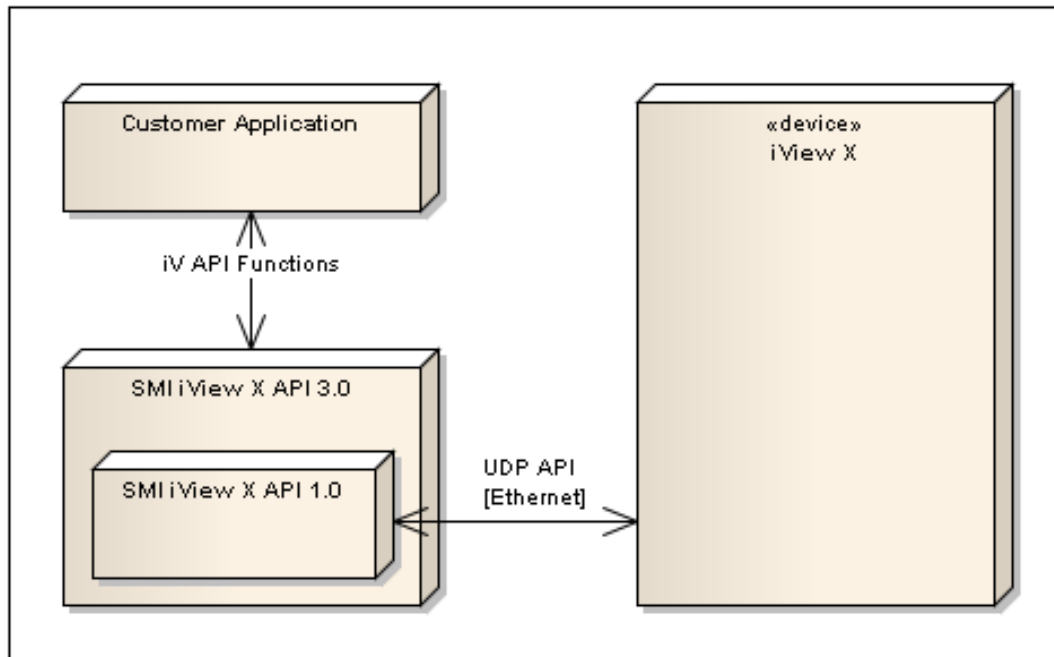
In addition to this document, the SDK includes release notes, which may be found in the SMI\iView X SDK\docs directory. In the release notes you can find a complete list of the improvements and bug fixes we have made, helping you get the most from each release.

Important Notes:

PLEASE NOTE that in order to exchange data between iView X™ and your software application using the SDK, an Ethernet (IEEE 802.3) connection *must* be established. This applies even when running iView X™ and your software application on the same PC. If unfamiliar with such process, please consult relevant documentation (e.g. the iView X™ user manual) on how to establish an Ethernet connection between different computers. Please adjust the IP address and port settings in iView X™ *and* your application accordingly.

API layer overview:

Shown below is a graphical overview of the iView X™ API.



iView X™ SDK installer contains 32-bit DLLs

NOTE: This SDK installer contains Windows 32bit DLLs. Although you can install and run the iView X™ SDK binaries on a Windows 64-bit OS given that 64-bit processors can run both 64-bit and 32-bit applications natively, please note that it is not possible to implement the binaries in Windows 64-bit applications. 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.

7

System Requirements

The sections below describe the system requirements for developing applications using the iView X™ SDK.

Supported Eye Tracking Devices

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

Supported Eye Tracking Systems	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
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

In general, it is always best to make sure that you are running the latest available version of iView X™ and firmware for your eye tracker model with the SDK. The most recent iView X™ software is always provided on the SMI Support Software Downloads page:

<http://www.smivision.com/en/gaze-and-eye-tracking-systems/support/software-download.html>.

Please note that the SDK does not support iView X™ releases prior to v.2.0.

Supported Programming and Scripting Languages

The iView X™ SDK can be used with most programming and scripting languages that are capable of importing C dynamic link libraries (DLLs). These include, but are not limited to, C++, C#, Matlab, E-Prime, Python, and NBS Presentation. The SDK includes several programming examples to help you get started in your application development. They are as follows:

Languages	Example
C++	Remote Control Application
C#	Remote Control Application
MATLAB®	Slide show and Gaze contingent Experiment
E-Prime	Slide show and Gaze contingent Experiment
Python	Slide show and Gaze contingent Experiment
NBS Presentation	Slide show and Gaze contingent Experiment

These examples applications are included in the SDK folder in the /examples directory. They highlight many of the features and capabilities of the iView X™ libraries and APIs.

8

Supported Operating Systems

The iView X™ SDK for is designed to run on the following operating systems:

Supported Operating Systems	Notes
Windows XP 32 bit	Supported
Windows XP 64 bit	Supported
Windows Vista 32 bit	Supported
Windows Vista 64 bit	Supported
Windows 7 32 bit	Supported
Windows 7 64 bit	Supported
Linux	Planned
Mac OS X	Planned

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	HiSpeed / Primate	HED	MRI / MEG
1	iV_AbortCalibration	X	X	X	-	X
2	iV_AcceptCalibrationPoint	X	X	X	-	X
3	iV_Calibrate	X	X	X	-	X
4	iV_ChangeCalibrationPoint	X	X	X	X	X
5	iV_ClearAOI	X	X	X	-	X
6	iV_ClearRecordingBuffer	X	X	X	X	X
7	iV_Connect	X	X	X	X	X
8	iV_ContinueEyetracking	X	X	-	-	-
9	iV_ContinueRecording	X	X	X	X	X
10	iV_DefineAOI	X	X	X	-	X
11	iV_DefineAOIPort	X	X	X	-	X
12	iV_DeleteMonitorAttachedGeometry	-	X	-	-	-
13	iV_DeleteStandAloneGeometry	X	-	-	-	-
14	iV_DisableAOI	X	X	X	-	X
15	iV_DisableAOIGroup	X	X	X	-	X
16	iV_DisableGazeDataFilter	X	X	X	-	X
17	iV_Disconnect	X	X	X	X	X
18	iV_EnableAOI	X	X	X	-	X
19	iV_EnableAOIGroup	X	X	X	-	X
20	iV_EnableGazeDataFilter	X	X	X	-	X
21	iV_GetAccuracy	X	X	X	-	X
22	iV_GetAccuracyImage	X	X	X	-	X
23	iV_GetCurrentCalibrationPoint	X	X	X	X	X
24	iV_GetCurrentTimestamp	X	X	X	X	X
25	iV_GetEvent	X	X	X	-	X
26	iV_GetEvent32	X	X	X	-	X
27	iV_GetEyeImage	X	X	X	X	X
28	iV_GetSample	X	X	X	X	X
29	iV_GetSample32	X	X	X	X	X
30	iV_GetSceneVideo	-	-	-	X	-
31	iV_GetSystemInfo	X	X	X	X	X
32	iV_GetTrackingMonitor	X	X	-	-	-
33	iV_IsConnected	X	X	X	X	X
34	iV_LoadCalibration	X	X	X	-	X
35	iV_Log	X	X	X	X	X
36	iV_PauseEyetracking	X	X	-	-	-
37	iV_PauseRecording	X	X	X	X	X
38	iV_Quit	X	X	X	X	X
39	iV_ReleaseAOIPort	X	X	X	-	X
40	iV_RemoveAOI	X	X	X	-	X
41	iV_ResetCalibrationPoints	X	X	X	X	X
42	iV_SaveCalibration	X	X	X	-	X
43	iV_SaveData	X	X	X	X	X
44	iV_SendCommand	X	X	X	X	X
45	iV_SendImageMessage	X	X	X	-	X

Function	RED	RED-m	HiSpeed / Primate	HED	MRI / MEG
46 iV_SetCalibrationCallback	X	X	X	-	X
47 iV_SetConnectionTimeout	X	X	X	X	X
48 iV_SetEventCallback	X	X	X	-	X
49 iV_SetEventDetectionParameter	X	X	X	-	X
50 iV_SetEyeImageCallback	X	X	X	X	X
51 iV_SetResolution	X	X	X	-	X
52 iV_SetLicense	-	X	-	-	-
53 iV_SetLogger	X	X	X	X	X
54 iV_SetResolution	X	X	X	-	X
55 iV_SetSampleCallback	X	X	X	X	X
56 iV_SetSceneVideoCallback	-	-	-	X	-
57 iV_SetTrackingMonitorCallback	X	X	-	-	-
58 iV_SetTrackingParameter	-	-	X	X	X
59 iV_SetupCalibration	X	X	X	-	X
60 iV_SetupMonitorAttachedGeometry	-	X	-	-	-
61 iV_SetupStandAloneMode	X	-	-	-	-
62 iV_ShowEyeImageMonitor	X	X	X	X	X
63 iV_ShowAccuracyMonitor	X	X	X	X	X
64 iV_ShowSceneVideoMonitor	-	-	-	X	-
65 iV_ShowTrackingMonitor	X	X	-	-	-
66 iV_Start	X	X	X	X	X
67 iV_StartRecording	X	X	X	X	X
68 iV_StopRecording	X	X	X	X	X
69 iV_Validate	X	X	X	-	X

Getting Started

Quickly get started with developing your SDK application by reading the sections below. In the following sections you will learn how to set up your SDK environment, about the various functions available in the SDK, and how to create your first basic eye tracking application based on the provided examples.

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

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>.

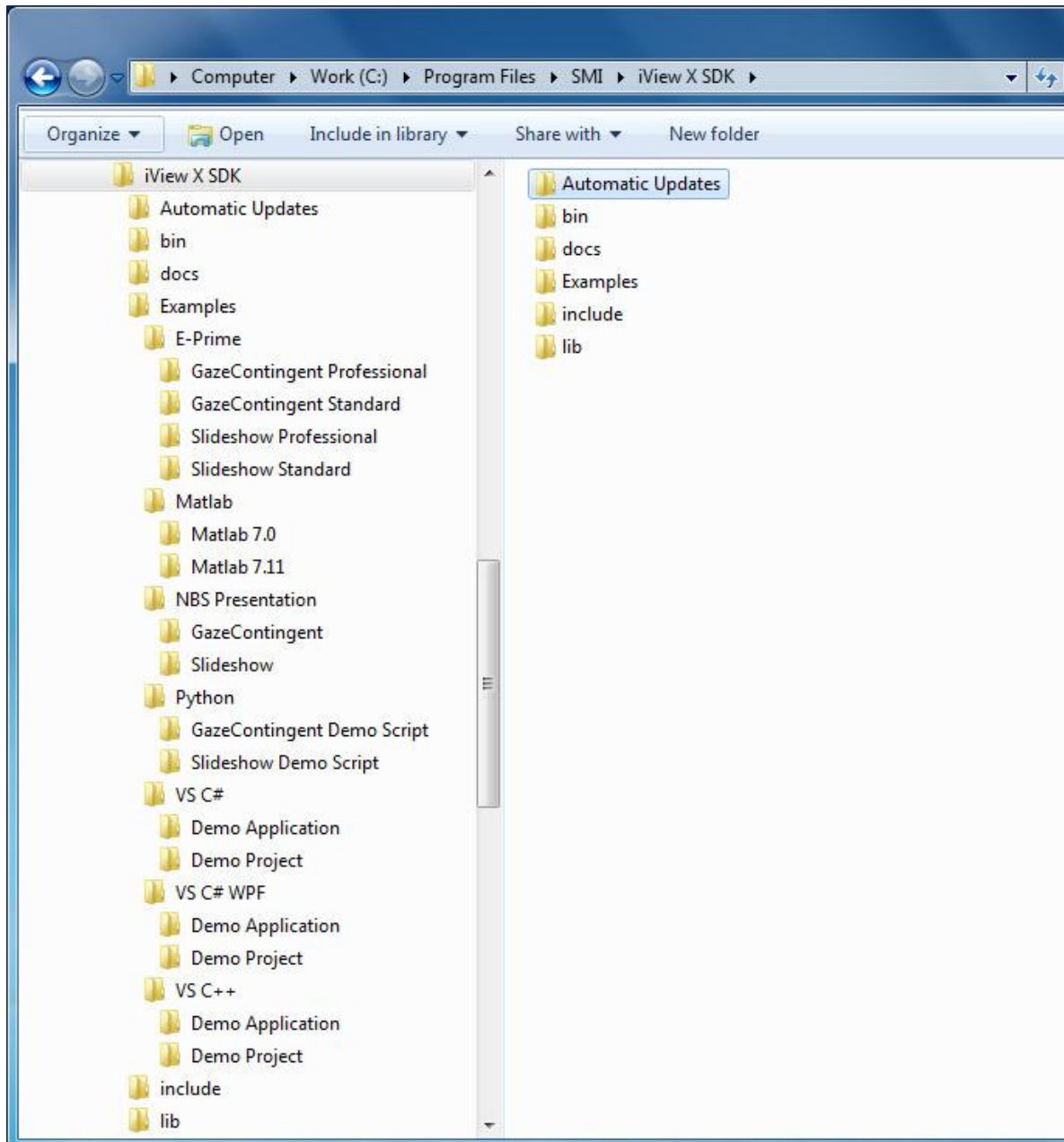
Running the Installer

After you have downloaded the SDK installer package, double-click the .msi file 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.

When the SDK installation process is completed, the following folder structure will be available on your computer:



As can be seen from the figure above, the SDK folder is divided into six sub-folders, “Automatic Updates”, “bin”, “docs”, “Examples”, “include”, and “lib”. The “bin” folder contains the Microsoft binaries. The “docs” folder contains documentation, which describes the iView X™ API itself. The “Examples” folder contains several sample scripts and programs, which provide a quick and easy start into controlling iView X™ via the SDK. For detailed syntax information the user can take a look into the functional characteristics of the ready-to-use source code of all examples. The examples illustrate the basic functionalities of the SDK and can be used as a baseline for developing your own projects and/or experiments. The “include” folder contains the header file of the iView X™ API. The “lib” folder contains the static library of the iView X™ API.

Next Steps

Once you have completed installation of the SDK, you are ready to begin developing applications. Here are a few ways you can get started:

Explore some code

The SDK includes sample code and applications for each major platform. You can browse these examples in the Examples folder.

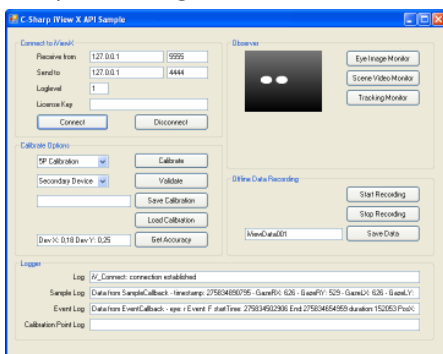
Learn about the iView X™ SDK

Take a look at the “Getting Started with the SDK Examples” and “iView X™ SDK Reference” sections below to learn more about the included example programs and available functions.

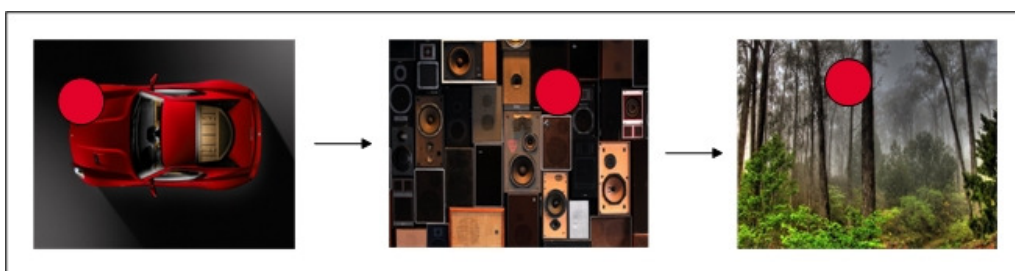
Getting started with the SDK Examples


The following sample experiments are provided with the SDK:

- 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.



 = real time gaze overlay (not drawn to scale)

- 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 example illustrates several example functions and commands and is a good starting point for writing your own eye tracking application.



The above examples demonstrate concepts that are fundamental to SDK 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. (e.g., \Examples\VS C# for the C# demo application) The source code will give a more detailed insight into the possibilities of the SDK and its functions.

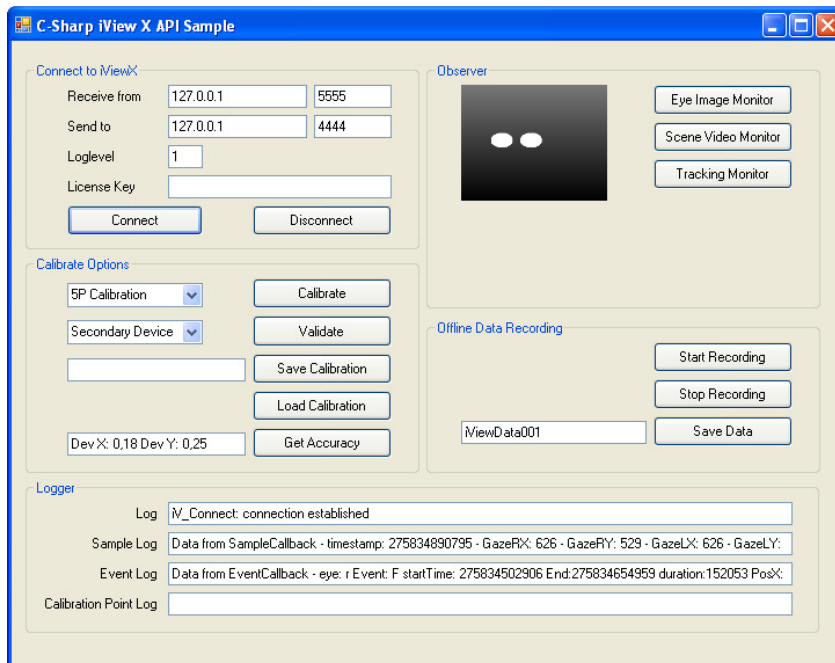
Using C#

The SDK includes a C# example program, the Remote Control Application, to help you get started with developing your own application.

Languages	Example
C#	Remote Control Application

The C# example was created using Visual Studio 2008.

You can run the C# demo application by double-clicking on the “csdemo.exe” file in the VS C#\Demo Application folder. Doing so will bring up an application that looks like the image shown below. The full source code of this sample is included in the VS C#\Demo Project folder.

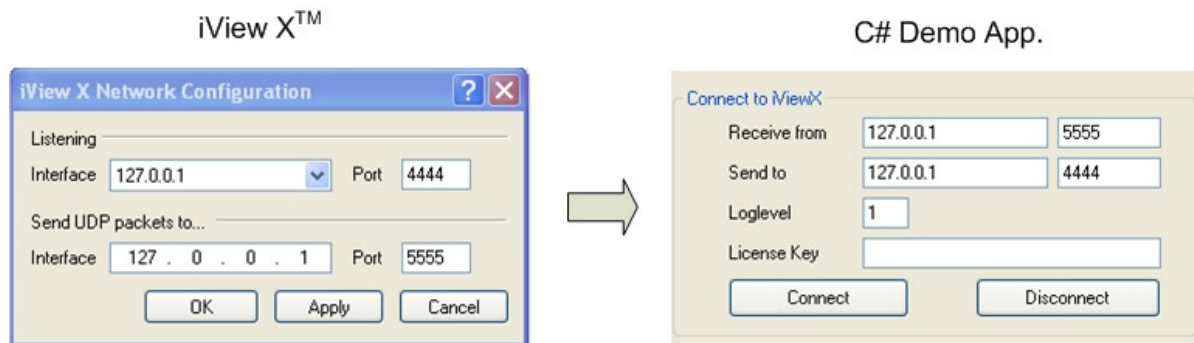


To establish a connection to iView X™ you must first set the according IP addresses in the *Connect to iView X* sections of the User Interface. For single-PC setups, the “Received from” and “Send to” IP addresses and ports will likely be (127.0.0.1; 5555) and (127.0.0.1; 4444), respectively. Please note

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. After configuring the IP addresses and ports, click the “Connect” button. If a connection has been established, gaze data will be streamed automatically and will be shown in the “Sample” text box.



The following code shows how to declare and use several functions from the SDK function set.

Declaring external functions and data structs:

```
[DllImport("iView XAPI.dll")]
public static extern Int32 iV_Connect(StringBuilder sendIP, int sendPort, StringBuilder 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 right eye
    public Int32 planeNumber;             // plane number of gaze data sample (only HED HT)
};
```

15

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();
}
```


Using MATLAB®

The SDK includes two MATLAB® example programs to help you get started with developing your own applications. They will provide you with insights on how to setup a Slideshow and Gaze Contingent experiment using the iView X™ API.

Languages	Example
Matlab	Slide show and Gaze contingent Experiment

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 just for visualization purposes and is not required for communication with iView X™. For using the iView X™ SDK without the “psychophysics toolbox” use the DataStreaming example enclosed in the iView X™ SDK. Due to changes in Matlab in handing over parameter to dynamic libraries the MATLAB® examples were written with version 7.0 and version 7.11.

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 SDK DLL. It also defines a struct which is used to receive online data from the eye tracking device:

```
loadlibrary('iView XAPI.dll', 'iView XAPI.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 = int32(0);
Sample.leftEye = EyeData;
Sample.rightEye = EyeData;
Sample.planeNumber = int32(0); pSample32 = libpointer('SampleStruct32', 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('iView XAPI', 'iV_Connect', int8('127.0.0.1'), int32(4444), int8('127.0.0.1'), int32(5555))

calllib('iView XAPI', 'iV_GetSample32', pSample32)
get(pSample32, 'Value')

calllib('iView XAPI', 'iV_Disconnect')
unloadlibrary('iView XAPI');
```

Using Python

The iView X™ SDK includes three sample experiments for use with Python that are similar to those included for MATLAB®. To run the Slideshow and Gaze Contingent experiments, 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. However, please note that the toolbox is NOT required for communication with iView X™, which is demonstrated in the SimpleExperiment. These Python examples were written with Python version 2.6.6. and the Psychopy2 toolbox version 1.73.06.

In the **iViewXAPI** file it is demonstrated, how to import the iView X™ SDK library and how to declare and initialize data structs that are needed for the use of the iView X™ SDK functions.

The file **iViewXAPIReturnCodes** handles iView X SDK™ return codes in case of a connection error.

The following code shows how to load the required SDK DLL. Connecting to, retrieving data and disconnecting from iView X™ look like this:

```

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()

```

Using E-Prime

The SDK includes several example experiments for E-Prime, two for the Standard version and two for the Professional version. Since E-Prime does not allow other programs to display visualizations, no images may be created by the SDK when used in conjunction 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 E-Prime limitation in handling callback functions its needed to poll for the required data. 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.

Languages	Example
E-Prime	Slide show and Gaze contingent Experiment

The following code shows how to declare structs and functions from SDK that are needed for connecting to, getting a sample 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

```

19

The following code shows how to connect to, get 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

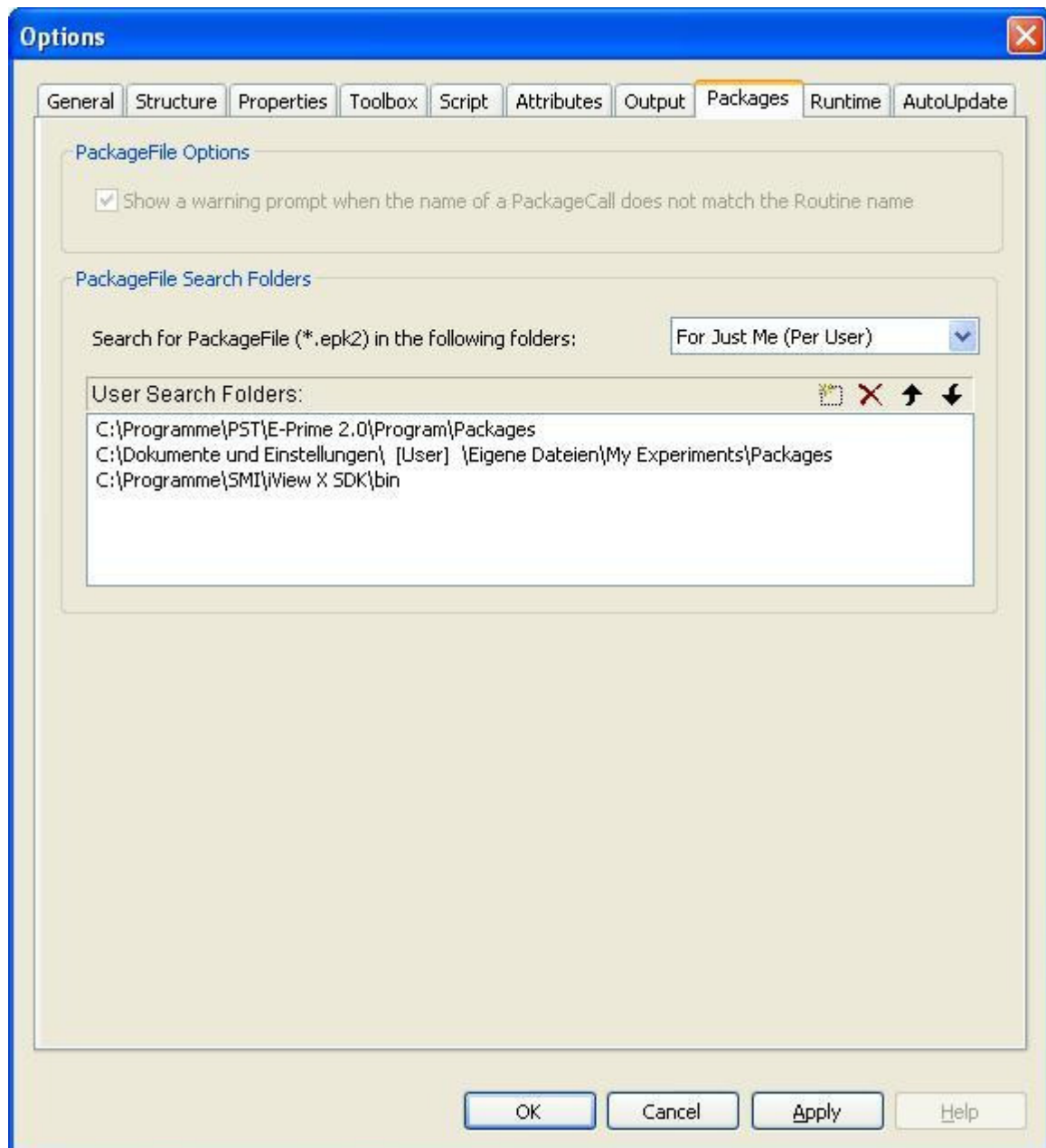
' connect to iView X
ret = iV_Connect (sendIPAddress, sendPort, recvIPAddress, readPort)

ret = iV_GetSample32 (sample)

```

NOTE: The iView X™ SDK provides a package file (.epk2) for E-Prime 2 Professional to simplify writing of own experiments. To make the package file available in E-Prime you have to set an according 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



Using NBS Presentation

The SDK includes two example experiments for use with NBS Presentation. Since the iView X™ API was implemented inside the NBS Presentation COM API, the iView X SDK .dll file associated with NBS Presentation needs to be registered with NBS. The particular .dll file, "iViewXAPI_NBS.dll", may be found in the "\iView X SDK\Binaries" folder. Registration of the .dll must be done through the NBS Presentation "Extension Manager". To begin, navigate to the "Tools" menu option in NBS Presentation, click the "Select Extension File" button, and subsequently find the "iViewXAPI_NBS.dll" file. After a successful registration of the extension, the extension can be used in the .pcl files. For more information on Presentation extensions and the Extension Manager please visit the NBS website (<http://www.neurobs.com>).

Languages	Example
NBS Presentation	Slide show and Gaze contingent Experiment

The supported iView X™ API functions are distributed in two different Presentation Extensions (Eye_Tracker and PCL_Extension). The following code shows how to create instances of both extensions and how to use them.

```
# create iViewXAPI 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 instance and start tracking
eye_tracker tracker = new eye_tracker( "{FDC35980-7480-4761-859F-4DCCFA93BA57}" );
tracker.start_tracking();
tracker.start_data(dt_position);

# start calibration and recording
tracker.calibrate( et_calibrate_default, calibration_method, calibration_auto_accept, calibration_speed);
tracker.set_recording (true);

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

# stop recording and save data
tracker.set_recording (false);
tracker2.save_data("presentation_data.idf", "description", "user", 1);

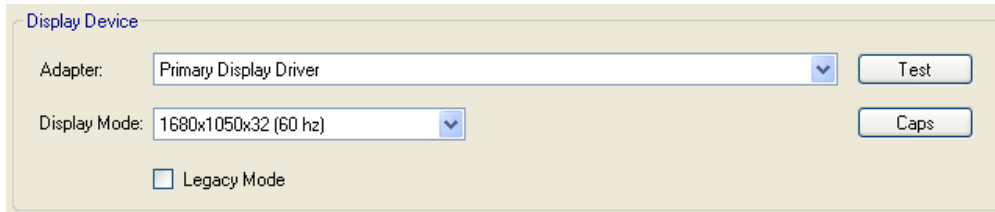
# disconnect from iView X

tracker2.disconnect
```

Before getting started with the NBS Presentation example experiments included with the SDK, you will want to 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 displaying your experiment on a secondary monitor, you will want to select the *Secondary Display Driver* option from the “Adapter” drop-down menu.



(2) Screen Resolution Settings

The Screen Resolution Settings for the NBS Presentation experiments is set in the .sce file. You will want to be 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.

```
#####
#
#  set the resolution
#
#####

$width_screen = 1680;           # width of screen
$height_screen = 1050;         # height of screen
```

(3) Network Connection Settings

The Network Connection Settings for the NBS Presentation experiments is set in the .pcl file. You will want to verify that settings here match those set forth in iViewX (“Setup->Hardware->Communication->Ethernet”). Otherwise, the NBS Presentation experiment will not be able to communicate with iViewX. 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 conjunction 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.

iView X SDK Reference

This section provides comprehensive information about all data structures and functions available through the SDK.

Header File

Defines

#define LOG_BUG	1
#define LOG_IV_FCT	2
#define LOG_ETCOM	4
#define LOG_ALL	8
#define LOG_IV_COMMAND	16
#define ET_PARAM_EYE_LEFT	0
#define ET_PARAM_EYE_RIGHT	1
#define ET_PARAM_PUPIL_THRESHOLD	0
#define ET_PARAM_REFLEX_THRESHOLD	1
#define ET_PARAM_SHOW_AOI	2
#define ET_PARAM_SHOW_CONTOUR	3
#define ET_PARAM_SHOW_PUPIL	4
#define ET_PARAM_SHOW_REFLEX	5
#define ET_PARAM_DYNAMIC_THRESHOLD	6
#define ET_PARAM_PUPIL_AREA	11
#define ET_PARAM_PUPIL_PERIMETER	12
#define ET_PARAM_PUPIL_DENSITY	13
#define ET_PARAM_REFLEX_PERIMETER	14
#define ET_PARAM_REFLEX_PUPIL_DISTANCE	15

Enumerations

```
enum ETSystem { NONE, RED, HiSpeed, MRI, HED, Custom }
```

Structs

- AccuracyStruct
- AOIRectangleStruct
- AOIStruct
- CalibrationPointStruct
- CalibrationStruct
- EventStruct
- EventStruct32
- EyeDataStruct
- ImageStruct
- MonitorAttachedGeometryStruct

- StandAloneModeGeometryStruct
- SampleStruct
- SampleStruct32
- SystemInfoStruct

Functions

- int iV_AbortCalibration()
- int iV_AcceptCalibrationPoint()
- int iV_Calibrate ()
- int iV_ChangeCalibrationPoints (int number, int positionX, int positionY)
- int iV_ClearRecordingBuffer ()
- int iV_Connect (char sendIPAddress[16], int sendPort, char recvIPAddress[16], int receivePort)
- int iV_ContinueEyetracking()
- int iV_ContinueRecording (char etMessage[256])
- int iV_DefineAOI (struct AOIStruct *aoiData)
- int iV_DefineAOIPort (int port)
- int iV_DeleteMonitorAttachedGeometry (char name[256])
- int iV_DeleteStandAloneGeometry (char name[256])
- int iV_DisableAOI (char aoiName[256])
- int iV_DisableAOIGroup (char aoiName[256])
- int iV_DisableGazeDataFilter ()
- int iV_Disconnect ()
- int iV_EnableAOI (char aoiName[256])
- int iV_EnableAOIGroup (char aoiName[256])
- int iV_EnableGazeDataFilter()
- int iV_GetAccuracy (struct AccuracyStruct *accuracyData, int visualization)
- int iV_GetAccuracyImage (struct ImageStruct *imageData)
- int iV_GetCalibrationParameter (struct CalibrationStruct *calibrationData)
- int iV_GetCurrentCalibrationPoint (struct CalibrationStruct *actualCalibrationPoint)
- int iV_GetCurrentTimestamp (int64 *currentTimestamp)
- int iV_GetEvent (struct EventStruct *EventDataSample)
- int iV_GetEvent32 (struct EventStruct32 *EventDataSample)
- int iV_GetSample (struct SampleStruct *rawDataSample)
- int iV_GetSample32 (struct SampleStruct32 *rawDataSample)
- int iV_GetSystemInfo (struct SystemInfoStruct *systemInfoData)
- int iV_GetTrackingMonitor (struct ImageStruct *image)
- int iV_IsConnected ()
- int iV_LoadCalibration (char name[256])
- int iV_Log (char logMessage[256])
- int iV_PauseEyetracking ()
- int iV_PauseRecording ()
- int iV_Quit()
- int iV_ReleaseAOIPort ()
- int iV_RemoveAOI (char aoiName[256])
- int iV_ResetCalibrationPoints()
- int iV_SaveCalibration (char name[256])
- int iV_SaveData (char filename[256], char description[64], char user[64], int overwrite)
- int iV_SendCommand (char etMessage[256])

- int iV_SendImageMessage (char etMessage[256])
- void iV_SetCalibrationCallback (pDLLSetCalibrationPoint pCalPoint)
- int iV_SetConnectionTimeout (int time)
- int iV_SetResolution (int stimulusWidth, int stimulusHeight)
- void iV_SetEventCallback (pDLLSetEvent pEvent)
- int iV_SetEventDetectionParameter (int minduration, int maxDispersion)
- void iV_SetEyeImageCallback (pDLLSetEyeImage pEyeImage)
- int iV_SetLicense (char licenseKey[16])
- int iV_SetLogger (int status, char filename[256])
- void iV_SetSampleCallback (pDLLSetSample pSample)
- void iV_SetSceneVideoCallback (pDLLSetSceneVideo pSceneVideo)
- void iV_TrackingMonitorCallback (pDLLSetTrackingMonitor pTrackingMonitor)
- int iV_SetTrackingParameter (int ET_PARAM_EYE, int ET_PARAM, int value)
- int iV_SetupCalibration (struct CalibrationStruct *CalibrationData)
- int iV_SetupMonitorAttachedGeometry (struct MonitorAttachedGeometryStruct attachedMonitorGeometry)
- int iV_SetupStandAloneMode (struct StandAloneModeGeometryStruct standAloneModeGeometry)
- 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_Validate ()

Explanations for Defines

With **LOG_** defines it is possible to setup the logging status for the function “iV_Log”. With “iV_Log” it is possible to observe the communication between a user’s application and iView X™ and/or function calls. Log levels can be combined (e.g. **LOG_BUG** | **LOG_IV_COMMAND** | **LOG_ETCOM**).

```
#define LOG_LEVEL_BUG                1
#define LOG_LEVEL_IV_FCT              2
#define LOG_LEVEL_ETCOM               4
#define LOG_LEVEL_ALL                 8
#define LOG_LEVEL_IV_COMMAND          16
```

With **ET_PARAM_** and function “iV_SetTrackingParameter” it is possible to change iView X™ 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™ system. Only experienced users should use this function.

```
#define ET_PARAM_EYE_LEFT             0
#define ET_PARAM_EYE_RIGHT            1
```

```
#define ET_PARAM_PUPIL_THRESHOLD 0
#define ET_PARAM_REFLEX_THRESHOLD 1
#define ET_PARAM_SHOW_AOI 2
#define ET_PARAM_SHOW_CONTOUR 3
#define ET_PARAM_SHOW_PUPIL 4
#define ET_PARAM_SHOW_REFLEX 5
#define ET_PARAM_DYNAMIC_THRESHOLD 6
#define ET_PARAM_PUPIL_AREA 11
#define ET_PARAM_PUPIL_PERIMETER 12
#define ET_PARAM_PUPIL_DENSITY 13
#define ET_PARAM_REFLEX_PERIMETER 14
#define ET_PARAM_RELFEFEX_PUPIL_DISTANCE 15
```

Explanations for Enumerations

The enumeration ETDevice can be used in connection with “iV_GetSystemInfo” to get information about which type of device is connected to iView X™. It is part of the “SystemInfoStruct”.

```
enum ETDevice { NONE, RED, HiSpeed, MRI, HED, Custom }
```

Explanations for Data Structures

AccuracyStruct Reference

This struct provides information about the last validation. If no validation has been done so far all data fields have the value -1.

Data Fields

double deviationLX
 double deviationLY
 double deviationRX
 double deviationRY

Detailed Description

deviationLX: horizontal deviation target - gaze position for left eye [°]
 deviationLY: vertical deviation target - gaze position for left eye [°]
 deviationRX: horizontal deviation target - gaze position for right eye [°]
 deviationRY: vertical deviation target - gaze position for right eye [°]

To update information in “AccuracyStruct” use function iV_GetAccuracy.

CalibrationPointStruct Reference

This struct provides information about the current calibration point. If no calibration or validation is in progress all data fields have the value -1.

Data Fields

int number
 int positionX
 int positionY

Detailed Description

number: number of calibration point that is currently active
 positionX: horizontal position of calibration point that is currently active
 positionY: vertical position of calibration point that is currently active

To update information in “CalibrationPointStruct” use function iV_GetCurrentCalibrationPoint during a calibration or validation procedure.

EventStruct Reference

This struct provides information about the last eye event that has been calculated.

Data Fields

char eventType
 char eye
 long long startTime
 long long endTime
 long long duration
 double positionX
 double positionY

Detailed Description

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

The data describes the last eye event that has been calculated. It will be updated when a new event has been calculated. To update the data fields in “EventStruct” use function `iV_GetEvent` or the event callback function.

EventStruct32 Reference

This struct provides information about the last eye event that has been calculated.

Data Fields

char eventType
 char eye
 double startTime
 double endTime
 double duration
 double positionX
 double positionY

Detailed Description

This struct contains the following information:

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

The data describes the last eye event that has been calculated. It will be updated when a new event has been calculated. To update the data fields in “EventStruct32” use function `iV_GetEvent32` or the event callback function.

EyeDataStruct Reference

This struct provides information about eye data.

Data Fields

double gazeX
 double gazeY
 double diam
 double eyePositionX
 double eyePositionY
 double eyePositionZ

Detailed Description

gazeX: horizontal gaze position [pixel]
 gazeY: vertical gaze position [pixel]
 diam: pupil diameter [pixel, mm] (mm for RED devices)
 eyePositionX: horizontal eye position relative to camera
 eyePositionY: vertical eye position relative to camera
 eyePositionZ: distance to camera

“EyeDataStruct” is part of “SampleStruct”. To update information in “SampleStruct” use function `iV_GetSample` or the sample callback function.

SampleStruct Reference

This struct provides information about gaze data samples.

Data Fields

long long timestamp
 EyeDataStruct leftEye
 EyeDataStruct rightEye
 int planeNumber

Detailed Description

timestamp: timestamp of the last gaze data sample [microseconds]
 leftEye: eye data left eye
 rightEye: eye data right eye
 planeNumber: plane number of gaze data sample

The data describes the last gaze data sample that has been calculated. It will be updated when a new gaze data sample has been calculated. To update information in “SampleStruct” use function `iV_GetSample` or the sample callback function.

SampleStruct32 Reference

This struct provides information about gaze data samples.

Data Fields

double timestamp
 EyeDataStruct leftEye
 EyeDataStruct rightEye
 int planeNumber

Detailed Description

The struct contains the following information:

timestamp: timestamp of the last gaze data sample [microseconds]
 leftEye: eye data left eye
 rightEye: eye data right eye
 planeNumber: plane number of gaze data sample

The data describes the last gaze data sample that has been calculated. It will be updated when a new gaze data sample has been calculated. To update information in “SampleStruct32” use function `iV_GetSample32` or the sample callback function.

SystemInfoStruct Reference

This struct provides information about the eyetracking system in use.

Data Fields

int samplerate
 int iV_MajorVersion
 int iV_MinorVersion
 int iV_Buildnumber
 int API_MajorVersion
 int API_MinorVersion
 int API_Buildnumber
 enum ETDevice iV_ETDevice

 30

Detailed Description

samplerate: sample rate of eyetracking system in use
 iV_MajorVersion: major version number of iView X™ in use
 iV_MinorVersion: minor version number of iView X™ in use
 iV_Buildnumber: build number of iView X™ in use
 API_MajorVersion: major version number of iView X API in use
 API_MinorVersion: minor version number of iView X API in use
 API_Buildnumber: build number of iView X API in use
 iV_ETDevice: type of eyetracking device

To update information in “SystemInfoStruct” use function `iV_GetSystemInfo`.

CalibrationStruct Reference

Use this struct to customize calibration behaviour.

Data Fields

int method
 int visualization
 int displayDevice
 int speed
 int autoAccept
 int foregroundBrightness
 int backgroundBrightness
 int targetShape
 int targetSize

char targetFilename[256]

Detailed Description

method: select Calibration method (default: 5)
 visualization: set Visualization status [0: visualization by external stimulus program
 1: visualization by SDK (default)]
 displayDevice: set Display Device [0: primary device (default), 1: secondary device]
 speed: set Calibration/Validation speed [0: slow (default), 1: fast]
 autoAccept: set Calibration/Validation point acceptance [1: automatic (default)
 0: manual]
 foregroundBrightness: set Calibration/Validation target brightness [0..255] (default: 20)
 backgroundBrightness: set Calibration/Validation background brightness [0..255]
 (default: 239)
 targetShape: set Calibration/Validation target shape [IMAGE = 0,
 CIRCLE1 = 1 (default), CIRCLE2 = 2, CROSS = 3]
 targetSize: set Calibration/Validation target size (default: 10 pixels)
 targetFilename: select custom Calibration/Validation target

To set calibration parameters with “CalibrationStruct” use function “iV_SetupCalibration”.

MonitorAttachedGeometryStruct Reference

Use this struct to customize RED-m position relative to display device.

Data Fields

Int setupName
 int stimX
 int stimY
 int redStimDistHeight
 int redStimDistDepth
 int redInclAngle

Detailed Description

setupName: name for the defined geometry setup
 stimX: horizontal stimulus calibration size [mm]
 stimY: vertical stimulus calibration size [mm]
 redStimDistHeight: vertical distance RED to stimulus screen [mm]
 redStimDistDepth: horizontal distance RED to stimulus screen [mm]
 redInclAngel: RED inclination angle [°]

Use “MonitorAttachedGeometryStruct” and “iV_SetupMonitorAttachedGeometry” to setup RED-m position parameters.

StandAloneModeGeometryStruct Reference

Use this struct to customize RED stand alone mode.

Data Fields

Int setupName
 int stimX
 int stimY
 int stimHeightOverFloor
 int redHeightOverFloor
 int redStimDist
 int redInclAngle

Detailed Description

setupName: name for the defined geometry setup
 stimX: horizontal stimulus calibration size [mm]
 stimY: vertical stimulus calibration size [mm]
 stimHeightoverFloor: distance floor to stimulus screen [mm]
 redHeightOverFloor: distance floor to RED [mm]
 redStimDist: distance RED to stimulus screen [mm]
 redInclAngel: RED inclination angle [°]

Setup RED stand alone mode parameters with “StandAloneModeStruct” use function “iV_SetupStandAloneMode”.

ImageStruct Reference

Use this struct to receive image buffer for receiving images.

Data Fields

int imageHeight
 int imageWidth
 int imageSize
 char* imageBuffer

Detailed Description

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

The struct will be used to transmit eye image, scene video and RED tracking monitor to allow GUI visualization. To update an image struct use iV_GetEyeImage, iV_GetSceneVideo, iV_GetTrackingMonitor, or use the callback functions.

AOIRectangleStruct Reference

Use this struct to customize AOI position on screen.

Data Fields

int x1
 int x2
 int y1

int y2

Detailed Description

x1: x-coordinate of left border of the AOI
 x2: x-coordinate of right border of the AOI
 y1: x-coordinate of upper border of the AOI
 y2: x-coordinate of lower border of the AOI

The struct defines the position of AOI on screen. Use `iV_DefineAOI` to setup an AOI.

AOIStruct Reference

Use this struct to customize trigger AOI.

Data Fields

int enabled
 char aoiName[256]
 char aoiGroup[256]
 AOIRectangleStruct position
 Int fixationHit
 char eye
 int outputValue
 char outputMessage[256]

Detailed Description

enabled: enable/disable trigger functionality [1: enabled, 0: disabled]
 aoiName: name of the AOI
 aoiGroup: group name of the AOI
 position: position of the AOI on screen
 fixationHit: uses fixations or gaze data as trigger [1: fixation hit, 0: raw data hit]
 eye: ['l': left, 'r': right]
 outputValue: TTL output value
 outputMessage: message in idf data stream

33

Defines trigger position, trigger parameter, and trigger value to synchronize raw data or fixation AOI hits with external devices. Use `iV_DefineAOIPort` and `iV_DefineAOI` functions to setup.

Function Reference

This section contains detailed information on the functions included with the SDK.

int iV_AbortCalibration ()

aborts the calibration or validation

Parameters:

<i>none</i>	
-------------	--

Returns:

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_AcceptCalibrationPoint ()

accepts a calibration or validation point (participant has to be tracked; only during a calibration or validation)

Parameters:

<i>none</i>	
-------------	--

Returns:

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_Calibrate ()

starts a calibration procedure. Change calibration and validation parameter with “iV_SetupCalibration”. If “CalibrationStruct::visualization” is set to “1” with “iV_SetupCalibration” “iV_Calibrate” will not return until the calibration has been finished or aborted.

Parameters:

<i>none</i>	
-------------	--

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
RET_CALIBRATION_ABORTED	- Calibration was aborted
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

Parameters:

<i>number</i>	Selected calibration point
<i>positionX</i>	New X position on screen
<i>positionY</i>	New Y position on screen

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
-------------	---

- ERR_NOT_CONNECTED - no connection established
- ERR_WRONG_PARAMETER - parameter out of range
- ERR_NO_RESPONSE_FROM_IVIEW - no response from iView X

int iV_ClearAOI ()

Removes all AOIs

Parameters:

<i>none</i>	
-------------	--

Returns:

- RET_SUCCESS - intended functionality has been fulfilled
- RET_NO_VALID_DATA - no data available
- ERR_AOI_ACCESS - could not access AOI data

int iV_ClearRecordingBuffer ()

clears the data buffer and scene video buffer (if connected eyetracking device is "HED").

Parameters:

<i>none</i>	
-------------	--

Returns:

- 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 data buffer is empty
- ERR_RECORDING_DATA_BUFFER - recording is activated

int iV_Connect (char sendIPAddress[16], int sendPort, char recvIPAddress[16], int receivePort)

establishes a UDP connection to iView X™.

"iV_Connect" will not return until connection has been established. If no connection can be established it will return after three seconds.

Parameters:

<i>sendIPAddress</i>	IP address of iView X™ computer
<i>sendPort</i>	port being used by iView X™ SDK for sending data to iView X™
<i>recvIPAddress</i>	IP address of local computer
<i>receivePort</i>	port being used by iView X™ SDK for receiving data from iView X™

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
ERR_IVIEWX_NOT_FOUND	- no SMI eye tracking application detected
ERR_EYETRACKING_APPLICATION_NOT_RUNNING	- no SMI eye tracking application running
ERR_WRONG_PARAMETER	- parameter out of range
ERR_COULD_NOT_CONNECT	- failed to establish connection

int iV_ContinueEyetracking ()

continues performing calculation of gaze data. Eye tracking can be paused with “iV_PauseEyetracking”

Parameters:

<i>none</i>	
-------------	--

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
ERR_NOT_CONNECTED	- no connection established

int iV_ContinueRecording (char etMessage[256])

continues gaze data recording and scene video recording (if connected eyetracking device is “HED”) “iV_ContinueRecording” does not return until gaze and scene video recording is continued

36

Parameters:

<i>etMessage</i>	text message to be written to data file
------------------	---

Returns:

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 data buffer is empty

int iV_DefineAOI(struct AOIStruct * aoiData)

defines an AOI. The API can handle up to 20 AOIs.

Parameters:

<i>aoiData</i>	See reference information for “AOIStruct”
----------------	---

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
ERR_WRONG_PARAMETER	- parameter out of range

int iV_DefineAOIPort(int portNumber)

selects a port for sending out TTL trigger

Parameters:

<i>port</i>	port address
-------------	--------------

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
ERR_WRONG_PARAMETER	- parameter out of range
ERR_COULD_NOT_OPEN_PORT	- could not open port for TTL output

int iV_DisableAOI (char aoIName[256])

disables all AOIs with the given name

Parameters:

<i>aoIName</i>	name of the AOI which will be disabled
----------------	--

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
RET_NO_VALID_DATA	- no data available
ERR_AOI_ACCESS	- could not access AOI data

int iV_DisableAOIGroup (char aoIGroup[256])

disables an AOI group

Parameters:

<i>aoIGroup</i>	name of the AOI group which will be disabled
-----------------	--

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
RET_NO_VALID_DATA	- no data available
ERR_AOI_ACCESS	- could not access AOI data

int iV_DisableGazeDataFilter()

disables the raw data filter

Parameters:

<i>none</i>	
-------------	--

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
-------------	---

int iV_Disconnect ()

disconnects from iView X™

“iV_Disconnect” will not return until the connection has been disconnected.

Parameters:

<i>none</i>	
-------------	--

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 ERR_DELETE_SOCKET - failed to delete sockets

int iV_EnableAOI (char aoIName[256])

enables all AOIs with the given name

Parameters:

<i>aoIName</i>	name of the AOI which will be enabled
----------------	---------------------------------------

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 RET_NO_VALID_DATA - no data available
 ERR_AOI_ACCESS - could not access AOI data

int iV_EnableAOIGroup (char aoIGroup[256])

enables an AOI group

Parameters:

<i>aoIGroup</i>	name of the AOI group which will be enabled
-----------------	---

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 RET_NO_VALID_DATA - no data available
 ERR_AOI_ACCESS - could not access AOI data

int iV_EnableGazeDataFilter()

enables a gaze data filter. This API bilateral filter was implemented due to special HCI application requirements

Parameters:

<i>none</i>	
-------------	--

Returns:

RET_SUCCESS - intended functionality has been fulfilled

int iV_GetAccuracy (struct AccuracyStruct * accuracyData, int visualization)

updates "accuracyData" with current accuracy data

If parameter "visualization" is set to "1" the accuracy data will be visualized in a dialog window

iV_GetAccuracy will not return until "AccuracyStruct" is updated

Parameters:

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

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 RET_NO_VALID_DATA - No new data available
 ERR_NOT_CONNECTED - no connection established
 ERR_NOT_CALIBRATED - system is not calibrated
 ERR_NOT_VALIDATED - system is not validated
 ERR_WRONG_PARAMETER - parameter out of range

int iV_GetAccuracyImage (struct ImageStruct * imageData)

39

updates "imageData" with validation visualization

Parameters:

<i>imageData</i>	see reference information for "ImageStruct"
------------------	---

Returns:

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

int iV_GetCurrentCalibrationPoint (struct CalibrationPointStruct * currentCalibrationPoint)

updates "currentCalibrationPoint" with current calibration point data

Parameters:

<i>currentCalibrationPoint</i>	see reference information for "CalibrationPointStruct"
--------------------------------	--

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 RET_NO_VALID_DATA - No new data available

ERR_NOT_CONNECTED - no connection established

int iV_GetCurrentTimestamp (int64* currentTimestamp)

requests the eye tracker timestamp

Parameters:

<i>currentTimestamp</i>	provides the internal timestamp
-------------------------	---------------------------------

Returns:

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

int iV_GetEvent (struct EventStruct * eventDataSample)

updates "eventDataSample" with current event data

Parameters:

<i>eventDataSample</i>	see reference information for "EventStruct"
------------------------	---

Returns:

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

40

int iV_GetEvent32 (struct EventStruct32 * eventDataSample)

updates "eventDataSample" with current event data

Parameters:

<i>eventDataSample</i>	see reference information for "EventStruct32"
------------------------	---

Returns:

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

int iV_GetEyeImage (struct ImageStruct* image)

updates "image" with current eye image

Parameters:

<i>image</i>	see reference information for "ImageStruct"
--------------	---

Returns:

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_GetSample (struct SampleStruct * rawDataSample)

updates "rawDataSample" with current eyetracking data.

Note: The "iV_GetSample" function should *only* be called up to as many times as the actual sampling rate of your eye tracker (e.g., 500Hz). The data in the "SampleStruct" will be set to a negative value when there is no new data available.

Parameters:

<i>rawDataSample</i>	see reference information for "SampleStruct"
----------------------	--

Returns:

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

int iV_GetSample32 (struct SampleStruct32 * rawDataSample)

updates "rawDataSample" with current eyetracking data.

Note: The "iV_GetSample32" function should *only* be called up to as many times as the actual sampling rate of your eye tracker (e.g., 500Hz). The data in the "SampleStruct" will be set to a negative value when there is no new data available.

 41

Parameters:

<i>rawDataSample</i>	see reference information for "SampleStruct32"
----------------------	--

Returns:

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

int iV_GetSceneVideo(struct ImageStruct* image)

updates "image" with current scene video image

Parameters:

<i>image</i>	see reference information for "ImageStruct"
--------------	---

Returns:

- 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_GetSystemInfo (struct SystemInfoStruct * systemInfoData)

updates "systemInfoData" with current system information

Parameters:

<i>systemInfoData</i>	see reference information for "SystemInfoStruct"
-----------------------	--

Returns:

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

int iV_GetTrackingMonitor (struct ImageStruct* image)

updates "image" with current tracking monitor image

Parameters:

<i>image</i>	see reference information for "ImageStruct"
--------------	---

Returns:

- 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_IsConnected ()

checks if connection to iView X™ is still established

Parameters:

<i>none</i>	
-------------	--

Returns:

- RET_SUCCESS - intended functionality has been fulfilled
- ERR_NOT_CONNECTED - no connection established

int iV_LoadCalibration (char name [256])

loads a saved calibration

a calibration has to be previously saved by using "iV_SaveCalibration"
can only be used with iView X version 2.3 or higher

Parameters:

<i>name</i>	calibration name / identifier
-------------	-------------------------------

Returns:

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_NO_RESPONSE_FROM_IVIEWX - no response from iView X; check calibration name / identifier

int iV_Log (char logMessage[256])

Writes "logMessage" to log file

Parameters:

<i>logMessage</i>	message that shall be written to the log file
-------------------	---

Returns:

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

int iV_PauseEyetracking ()

pauses eyetracking and calculation gaze data. Eye tracking can be continued with "iV_ContinueEyetracking"

Parameters:

<i>none</i>	
-------------	--

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 ERR_NOT_CONNECTED - no connection established

int iV_PauseRecording ()

pauses gaze data recording and scene video recording (if connected eyetracking device is "HED")
 "iV_PauseRecording" does not return until gaze and scene video recording is paused

Parameters:

<i>none</i>	
-------------	--

Returns:

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_Quit()

disconnects and closes iView X™
 can only be used with iView X™ version 2.8.7 or higher

Parameters:

<i>none</i>	
-------------	--

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 ERR_DELETE_SOCKET - failed to delete sockets

int iV_ReleaseAOIPort ()

releases the port for sending out TTL trigger

Parameters:

<i>none</i>	
-------------	--

44

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 ERR_COULD_NOT_CLOSE_PORT - failed to close TTL port

int iV_RemoveAOI (char aoIName[256])

removes all AOIs with the given name

Parameters:

<i>aoIName</i>	name of the AOI which will be removed
----------------	---------------------------------------

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 RET_NO_VALID_DATA - no new data available
 ERR_AOI_ACCESS - could not access AOI data

int iV_ResetCalibrationPoints()

resets all calibration points to default position

Parameters:

<i>none</i>	
-------------	--

Returns:

- RET_SUCCESS - intended functionality has been fulfilled
- ERR_NOT_CONNECTED - no connection established

int iV_SaveCalibration (char name [256])

stores a performed calibration
can only be used with iView X version 2.3 or higher

Parameters:

<i>name</i>	calibration name / identifier
-------------	-------------------------------

Returns:

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

int iV_SaveData (char filename [256], char description [64], char user [64], int overwrite)

writes data buffer and scene video buffer (if connected eyetracking device is “HED”) to file “filename”
“iV_SaveData” will not return until the data has been saved

Parameters:

<i>filename</i>	filename of data files being created (.idf: eyetracking data, .avi: scene video data)
<i>description</i>	optional experiment description
<i>user</i>	optional name of test person
<i>overwrite</i>	0: do not overwrite file “filename” if it already exists 1: overwrite file “filename” if it already exists

Returns:

- 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

int iV_SendCommand (char etMessage[256])

sends a remote command to iView X™. Please refer to the iView X™ help file for further information about remote commands.

Note: the “iV_SendCommand” is just temporarily implemented and will be deleted in the next release

Parameters:

<i>etMessage</i>	iView X™ remote command
------------------	-------------------------

Returns:

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[256])

sends a text message to iView X™. “etMessage” will be written to the data file. If “etMessage” ends on .jpg, .bmp, .png, or .avi BeGaze will separate the data buffer into according trials.

Parameters:

<i>etMessage</i>	text message to be written to data file
------------------	---

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
ERR_NOT_CONNECTED	- no connection established

void iV_SetCalibrationCallback (pDLLSetCalibrationPoint pCalibrationPoint)

The “iV_SetCalibrationCallback” function will be called if there are no valid parameters at the start of a calibration session, a calibration point has changed, the calibration session has finished, or the calibration session has been aborted either by the user or iViewX.

This callback provides users with the ability to draw a customized calibration routine.

Parameters:

<i>pCalibrationPoint</i>	pointer to CalibrationCallbackFunction
--------------------------	--

Returns:

none

CalibrationCallbackFunction definition:

CallbackFunction(CalibrationPointStruct calibrationPointData)

{

... visualize the point on screen ...

}

CalibrationPointStruct variables:

Number
positionX
positionY

void iV_SetEventCallback (pDLLSetEvent pEvent)

„iV_SetEventCallback“ function will be called if an real-time detected fixation has started or ended.

Parameters:

<i>pEvent</i>	pointer to EventCallbackFunction
---------------	----------------------------------

Returns:

none

int iV_SetEventDetectionParameter (int minDuration, int maxDispersion)

defines detection parameter for online fixation detection algorithm

Parameters:

<i>minDuration</i>	minimun fixation duration [ms]
<i>maxDispersion</i>	maximum dispersion [deg] for head tracking systems or [px] for non head tracking systems

Returns:

RET_SUCCESS - intended functionality has been fulfilled
ERR_WRONG_PARAMETER - parameter out of range

void iV_SetEyeImageCallback (pDLLSetEyeImage pEyeImage)

„iV_SetEyeImageCallback“ function will be called if an new eye image is available.

Parameters:

<i>pEyeImage</i>	pointer to EyeImageCallbackFunction
------------------	-------------------------------------

Returns:

none

int iV_SetLicense (char key[16])

validates the customer license (only for RED-m devices)

Parameters:

<i>key</i>	provided license key
------------	----------------------

Returns:

RET_SUCCESS - intended functionality has been fulfilled

int iV_SetLogger (int logLevel, char filename[256])

defines the logging behavior of iView X SDK

Parameters:

<i>logLevel</i>	see "Explanations for Defines" in this manual for further information
<i>filename</i>	filename of log file

Returns:

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

void 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.

48

Parameters:

<i>stimulusWidth</i>	horizontal resolution of stimulus screen [pixel]
<i>stimulusHeight</i>	vertical resolution of stimulus screen [pixel]

Returns:

RET_SUCCESS - intended functionality has been fulfilled
 ERR_WRONG_PARAMETER - parameter out of range

void iV_SetSampleCallback (pDLLSetSample pSample)

„iV_SetSampleCallback“ function will be called if iView X™ has generated a new raw data sample.

Important note: Dependent on the sample rate critical algorithms with high processor usage shouldn't be running within this callback

Parameters:

<i>pSample</i>	pointer to SampleCallbackFunction
----------------	-----------------------------------

Returns:

none

void iV_SetSceneVideoCallback (pDLLSetSceneVideo pSceneVideo)

„iV_SetSceneVideoCallback“ function will be called if a new scene video image is available.

Parameters:

<i>pSceneVideo</i>	pointer to SceneVideoCallbackFunction
--------------------	---------------------------------------

Returns:

none

void iV_SetTrackingMonitorCallback (pDLLSetTrackingMonitor pTrackingMonitor)

„iV_SetTrackingMonitorCallback“ function will be called if a new RED tracking monitor image is available.

Parameters:

<i>pTrackingMonitor</i>	pointer to TrackingMonitorCallbackFunction
-------------------------	--

Returns:

none

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

49

sets iView X tracking parameters

Important note: This function can strongly affect tracking stability of your iView X™ system. Only experienced users should use this function.

Parameters:

<i>ET_PARAM_EYE</i>	select specific eye
<i>ET_PARAM</i>	select parameter that shall be set
<i>value</i>	new value for selected parameter

Returns:

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

int iV_SetupCalibration(struct CalibrationStruct *calibrationData)

sets the calibration parameters

Parameters:

<i>calibrationData</i>	see reference information for “CalibrationStruct”
------------------------	---

Returns:

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

int iV_SetupMonitorAttachedGeometry (struct MonitorAttachedGeometryStruct *attachedModeGeometry)

defines the RED-m display device geometry

Parameters:

<i>attachedModeGeometry</i>	see reference information for "MonitorAttachedGeometryStruct"
-----------------------------	---

Returns:

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_SetupStandAloneMode (struct StandAloneModeGeometryStruct *standAloneModeGeometry)

defines remotely the RED stand-alone mode. See chapter RED stand alone Mode for further information

Parameters:

<i>standAloneModeGeometry</i>	see reference information for "StandAloneModeStruct"
-------------------------------	--

Returns:

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_ShowEyeImageMonitor ()

visualizes eye image in separate window

Parameters:

<i>none</i>	
-------------	--

Returns:

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_ShowSceneVideoMonitor()

visualizes scene video in separate window (available for HED devices only)

Parameters:

<i>none</i>	
-------------	--

Returns:

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_ShowTrackingMonitor ()

visualizes RED tracking monitor in separate window (available for RED devices only)

Parameters:

<i>none</i>	
-------------	--

Returns:

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_Start(int etApplication)

starts and connects automatically to iView X™ (only if iView X™ is running on the same PC)

Parameters:

<i>etApplication</i>	0: iView X 1: iView X OEM
----------------------	------------------------------

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
ERR_IVIEWX_NOT_FOUND	- no SMI eye tracking application detected
ERR_EYETRACKING_APPLICATION_NOT_RUNNING	- no SMI eye tracking application running
ERR_COULD_NOT_CONNECT	- failed to establish connection
ERR_IVIEWX_NOT_FOUND	- failed to start iView X™

int iV_StartRecording ()

starts gaze data recording and scene video recording (if connected eyetracking device is "HED")
"iV_StartRecording" does not return until gaze and scene video recording is started.

Parameters:

<i>none</i>	
-------------	--

Returns:

- 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 eyetracking device is “HED”)
 “iV_StopRecording” does not return until gaze and scene video recording is stopped

Parameters:

<i>none</i>	
-------------	--

Returns:

- 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_Validate ()

starts a validation procedure.

If “CalibrationStruct::visualization” is set to “1” with “iV_SetupCalibration” “iV_Validate” will not return until the validation has been finished or aborted.

Parameters:

<i>none</i>	
-------------	--

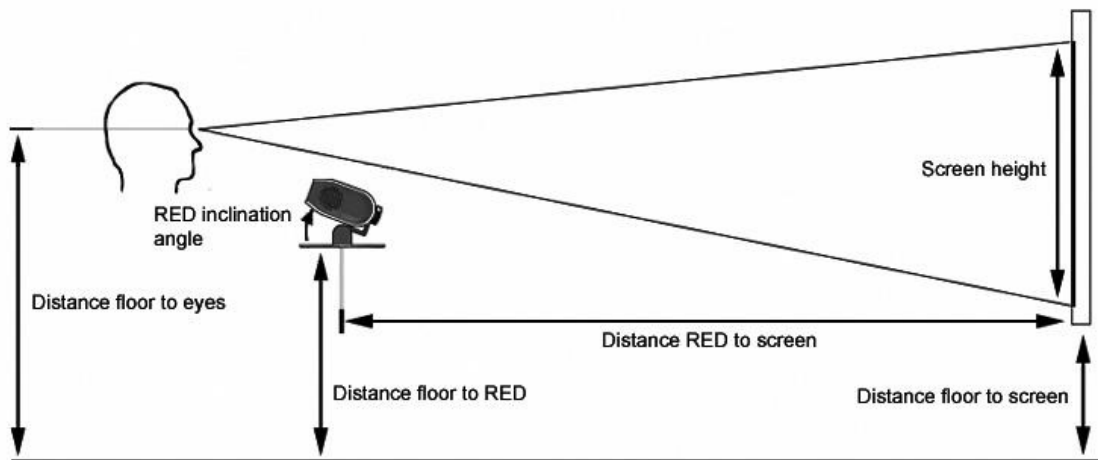
Returns:

- RET_SUCCESS - intended functionality has been fulfilled
- 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

RED Stand Alone Mode

The SDK can be used to configure the RED stand-alone mode. The data struct “standAloneModeGeometryStruct” contains all geometrical parameter while the function “iV_SetupStandAloneMode” configures remotely the settings due to the given stand-alone data. To change the mode the SDK needs an established connection to iView X™.

The corresponding profiles are stored and handled from iView X™ and are therefore system dependent.

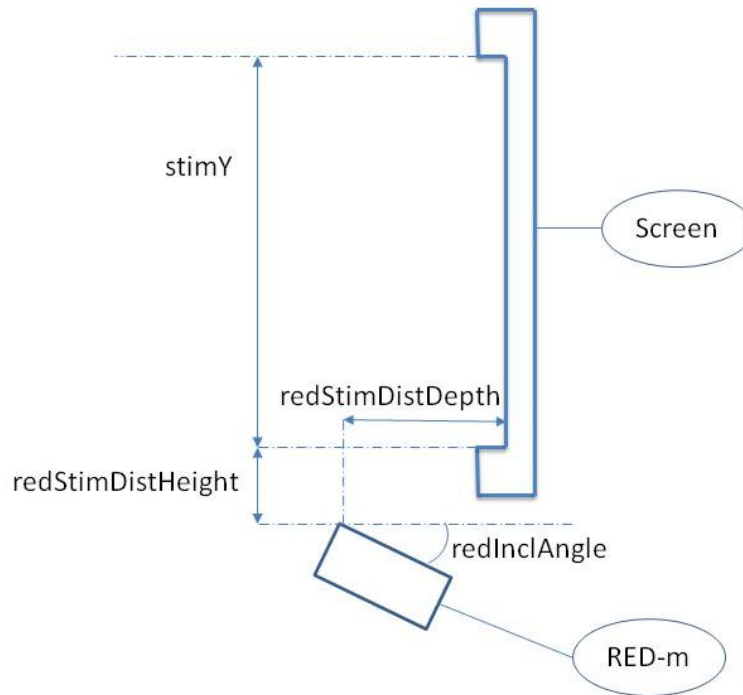


The following steps are necessary to setup the RED in stand-alone mode:

1. Remove the RED from the monitor and mount it at 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
4. Enter the geometrical dimensions of your setup into “standAloneModeGeometryStruct”
5. Call the function “iV_SetupStandAloneMode” including the “standAloneModeGeometryStruct” as parameter to iView X™

RED Monitor Attached Mode

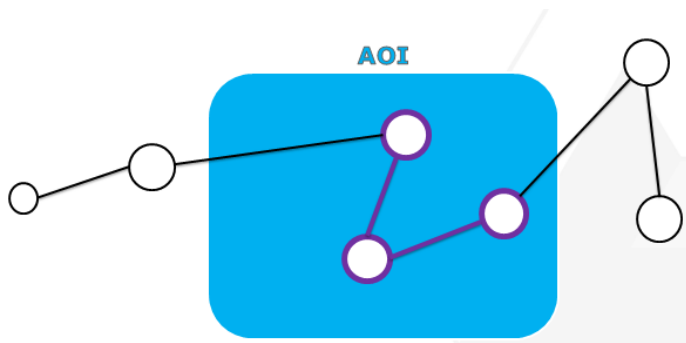
iView X™ SDK can be used to configure the RED-m in a mode attached to display device. The data "MonitorAttachedGeometryStruct" contains all geometrical parameter while the function "iV_SetupMonitorAttachedGeometry" configures remotely the settings related to the display device. To change the mode the API needs an established connection to iView X™ running a RED-m device.



Areas of Interest (AOI)

The Area of Interest (AOI) feature allows you to create objects within the scene view for real-time I/O signal generation. The iView X™ API performs an online analysis and detects, whether the raw gaze data enters or leaves an AOI, or an online detected fixation event was calculated within an AOI. If the recording was started a message will be send to the idf data stream. This is useful if you wish to trigger and synchronize other research devices by the test person's gaze position.

To define an output port, use the function “iV_DefineAOIPort”. After a port has been opened it is possible to generate Areas of Interest (AOI) and send out TTL values. See reference information for “iV_DefineAOI” and “AOIStruct” how to define AOIs.



Return Codes

Each iViewX SDK function defined as having a return type other than `void` should return a value. This value corresponds to a particular set of decimal and return codes, providing the developer with pertinent status information. The following is a list of all return codes defined:

Return Code	Decimal Codes	Notes
RET_SUCCESS	1	intended functionality has been fulfilled
RET_NO_VALID_DATA	2	No new data available
RET_CALIBRATION_ABORTED	3	Calibration was aborted
ERR_COULD_NOT_CONNECT	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_APPLICATION_NOT_RUNNING	104	no SMI eye tracking application running
ERR_WRONG_COMMUNICATION_PARAMETER	105	wrong port settings
ERR_WRONG_DEVICE	111	eye tracking device required for this function is not connected
ERR_WRONG_PARAMETER	112	parameter out of range
ERR_WRONG_CALIBRATION_METHOD	113	eye tracking device required for this calibration method is not connected
ERR_CREATE_SOCKET	121	failed to create sockets
ERR_CONNECT_SOCKET	122	failed to connect sockets
ERR_BIND_SOCKET	123	failed to bind sockets
ERR_DELETE_SOCKET	124	failed to delete sockets
ERR_NO_RESPONSE_FROM_IVIEW	131	no response from iView X; check iView X connection settings (IP addresses, ports) or last command
ERR_INVALID_IVIEWX_VERSION	132	iView X version could not be resolved
ERR_WRONG_IVIEWX_VERSION	133	wrong version of iView X
ERR_ACCESS_TO_FILE	171	failed to access log file
ERR_SOCKET_CONNECTION	181	socket error during data transfer
ERR_EMPTY_DATA_BUFFER	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 is not ready
ERR_IVIEWX_NOT_FOUND	201	no installed SMI eye tracking application detected
ERR_COULD_NOT_OPEN_PORT	220	Could not open port for TTL output
ERR_COULD_NOT_CLOSE_PORT	221	Could not close port for TTL output
ERR_AOI_ACCESS	222	Could not access AOI data

ERR_AOI_NOT_DEFINED

223

No defined AOI found

Important note: Certain functions write data to a struct that is provided to the function as parameter. If the function is called and new data is available this data will be written to the struct. If no new data is available all data in the struct will be set to -1.

Technical Support

Due to the complex nature of SDK's in general and the wide variety of applications that may be created using the iViewX SDK, it is not always possible to provide in-depth support. However, if you feel there is an error or omission in the iViewX 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 iViewX SDK to write your own eye tracking 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.

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, 2004-2012

SensoMotoric Instruments GmbH

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>
Copyright © 2012 SensoMotoric Instruments GmbH
Last updated: April 2012