iView X™ SDK

v3.0.7

June 2011

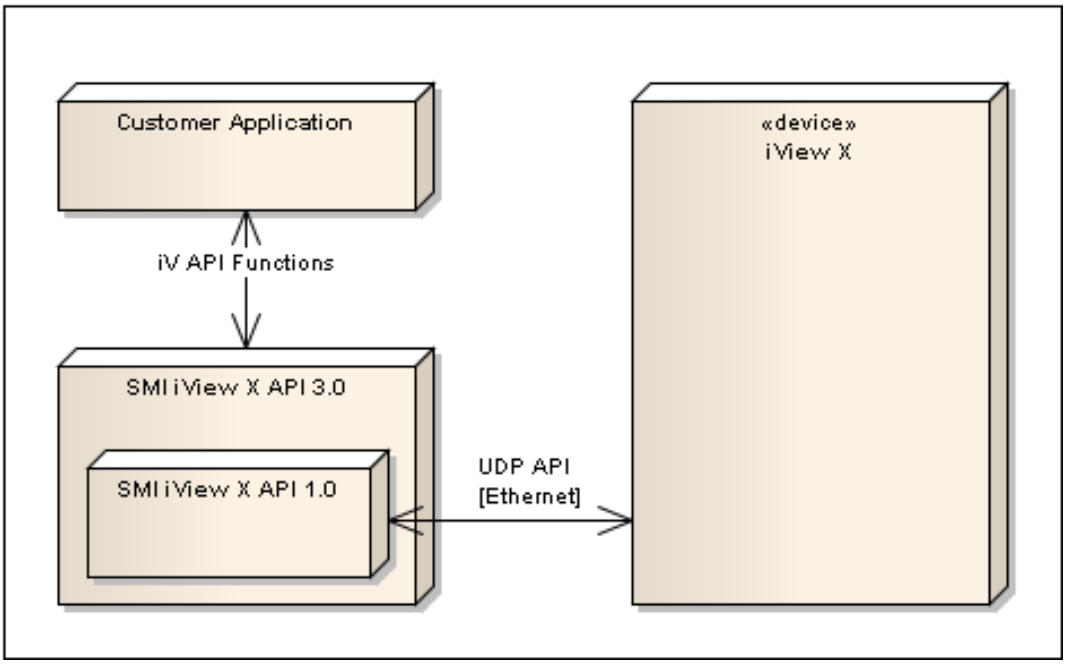
Table of Contents

Introduction

The iView X™ SDK provides an interface for communication between your application and iView X™. It uses UDP over Ethernet to provide maximum speed and minimum latency for data transfer. By using iView X™ SDK the programmer does not have to take care about Ethernet communication specific programming. The iView X™ SDK provides a large set of functions to control SMI eye tracker's and retrieve data online. It supports a growing number of programming languages and environments, e.g. C/C++, .Net, Matlab, Visual Basic, E-Prime.

Important note: To be able to exchange data between iView X™ and another application with the iView X™ SDK an Ethernet connection has to be established. Please consult according documentation on how to establish an Ethernet connection between different computers (e.g. the iView X™ user manual). Even when using iView X™ and the application on the same PC an Ethernet connection has to be established. Normally this happens via the so called localhost, 127.0.0.1. Please adjust IP address and port settings in iView X™ and your application accordingly.

API layer overview:



Supported Eye Tracking Devices

The iView X™ SDK allows to monitor and control human interfaces connected to an iView X™ system. The software version of iView X™ being used must be at least 2.0 or newer. The following list contains the supported SMI Eye Tracking devices:

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
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 (mono/bin)	500 / 1250 (mono / bin)
iView X™ MRI LR	50
iView X™ MEG	50 / 250

Included Examples

The iView X™ SDK can be used with all programming and scripting languages that are able to import C dynamic link libraries (DLLs). The following list contains all programming examples that are included in the iView X™ SDK package:

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

Supported Operating Systems

The following is a list of currently supported 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

iView X™SDK provides a vast size of functionalities which are provided for most SMI devices. The following list gives an overview about the devices and its supported functions.

	Function	RED	RED OEM	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_ClearRecordingBuffer	X	-	X	X	X
6	iV_Connect	X	X	X	X	X
7	iV_ContinueRecording	X	-	X	X	X
8	iV_DisableGazeDataFilter	X	X	X	-	X
9	iV_Disconnect	X	X	X	X	X
10	iV_EnableGazeDataFilter	X	X	X	-	X
11	iV_GetAccuracy	X	X	X	-	X
12	iV_GetActualTimestamp	X	X	X	X	X
13	iV_GetCurrentCalibrationPoint	X	X	X	-	X
14	iV_GetEvent	X	X	X	-	X
15	iV_GetEvent32	X	X	X	-	X
16	iV_GetSample	X	X	X	X	X
17	iV_GetSample32	X	X	X	X	X
18	iV_GetSystemInfo	X	X	X	X	X
19	iV_IsConnected	X	X	X	X	X
20	iV_LoadCalibration	X	X	X	-	X
21	iV_Log	X	X	X	X	X
20	iV_PauseRecording	X	-	X	X	X
21	iV_Quit	X	X	X	X	X
22	iV_ResetCalibrationPoints	X	X	X	X	X
23	iV_SaveCalibration	X	X	X	-	X
24	iV_SaveData	X	-	X	X	X
25	iV_SendCommand	X	-	X	X	X
26	iV_SendImageMessage	X	-	X	-	X
27	iV_SetCalibrationCallback	X	X	X	-	X
28	iV_SetEventCallback	X	X	X	-	X
29	iV_SetEventDetectionParameter	X	X	X	-	X
30	iV_SetSampleCallback	X	X	X	X	X
31	iV_SetLicense	-	X	-	-	-
32	iV_SetLogger	X	X	X	X	X
33	iV_SetTrackingParameter	-	-	X	X	X
34	iV_SetupCalibration	X	X	X	-	X
35	iV_SetupREDStandAloneMode	X	X	-	-	-
36	iV_ShowEyeImageMonitor	-	-	X	X	X
37	iV_ShowSceneVideoMonitor	-	-	-	X	-
38	iV_ShowTrackingMonitor	X	X	-	-	-
39	iV_Start	X	X	X	X	X

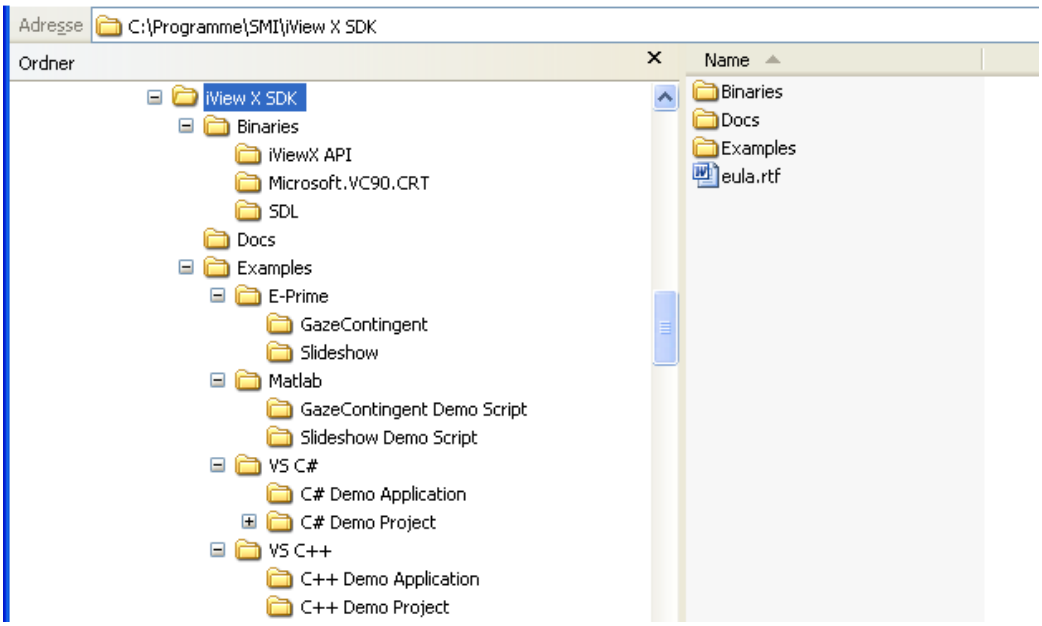
40	iV_StartRecording	X	-	X	X	X
41	iV_StopRecording	X	-	X	X	X
42	iV_Validate	X	X	X	-	X

Getting Started

The iView X™ SDK must be installed on the same PC where the application runs on that wants to communicate with iView X™. If the application runs on a dedicated PC the iView X™ SDK has to be installed on this PC. If the application runs on the same PC as iView X™ does it has to be installed on the iView X™ PC.

To install iView X™ SDK double-click the according .msi-file on the appropriate PC.

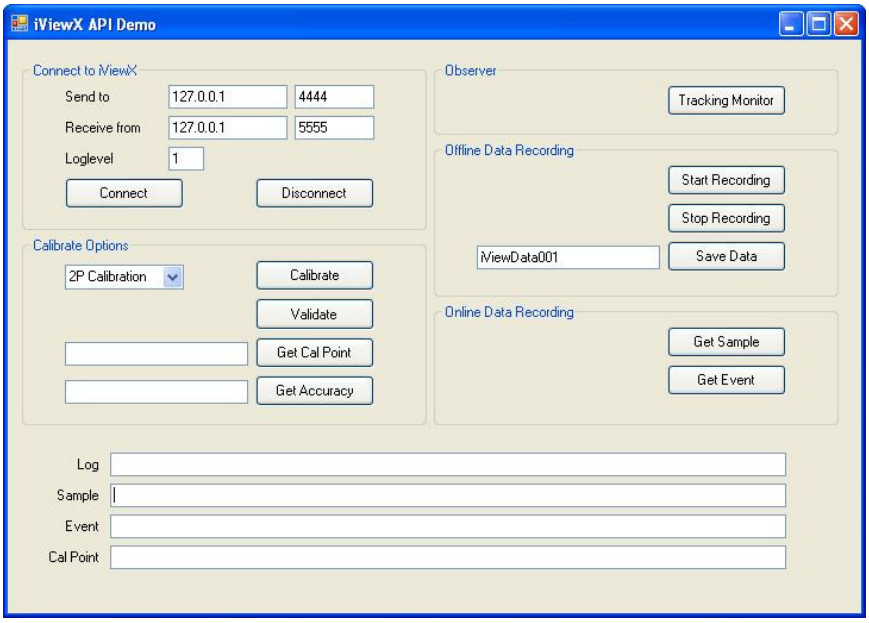
After installation of iView X™ SDK the following folder structure will be available on your system:



The folder “Binaries” contains the iView X™ SDK software itself and Microsoft binaries. The folder “Docs” contains documentation, which describes how to use iView X™ SDK. The folder “Examples” contains several sample scripts and programs for a quick and easy start into controlling iView X™. 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 provide basic functionalities and can be used as a baseline for own projects and/or experiments.

C-Sharp Demo Application

The C# demo example provides a simple application with the most common features for controlling iView X™.



To establish a connection to iView X™ set the according IP addresses in “Connect to iView X” and press “Connect”. If the 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 functions used from the iView X™ SDK DLL and how to actually use them.

Declaring external functions and data structs:

```
[DllImport("iView XAPI.dll")]
public static extern Int32 iV_Connect(char[] SendIP, int SendPort, char[] 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 iV_SampleData);

public struct SampleStruct
{
        public Int64 Timestamp;                                 // timestamp [microsecond]
        public double GazeRX, GazeRY, GazeLX, GazeLY;  // pupil gaze [pixel]
        public double DiamRX, DiamLX;                   // pupil diameter [pixel]
        public double DistanceR, DistanceL;             // distance vector element to each eye (only RED)
        public Int32 PlaneNumber;                       // using multiple planes (only HED HT)
};
SampleStruct iV_SampleData;
```

Using the functions from the DLL:

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

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

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

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

This code is taken from the provided programming example, which will give a more detailed insight into the possibilities of iView X™ SDK and its functions.

Using Matlab

Important note: To run the Matlab example script enclosed in the iView X™ SDK, it's necessary to download and install the "psychophysics toolbox" from psychotoolbox.org. The psychophysics toolbox provides Matlab specific visualizations being used in this example. The toolbox is not required for communication with iView X™ though. The Matlab examples were written with version 7.0 and version 7.11.

The Matlab example does not provide any dialog like the C# demo does. It is still possible though to use the same functionality like with the C# demo.

The following code shows how to load the required iView X™ SDK DLL. It also defines a struct which is used to receive online data from the Eye Tracker:

```
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 following code shows how to connect, get the actual data sample and disconnect from iView X™. After disconnecting, the library has to be unloaded:

```
calllib('iView XAPI', 'iV_Connect', '127.0.0.1', int32(4444), '127.0.0.1', int32(5555))

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

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

This code is taken from the provided programming example, which will give a more detailed insight into the possibilities of iView X™ SDK and its functions.

Using Python

Important note: To run the Python example script enclosed in the iView X™ SDK, it's necessary to download and install the "psychopy toolbox" from psychopy.org. The psychopy toolbox provides Python specific visualizations being used in this example. The toolbox is not required for communication with iView X™ though. These Python examples were written with Python version 2.7.

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

```
from ctypes import *

class eye(Structure):
        _fields_ = [("gazeX", c_int),
        ("gazeY", c_int),
        ("diam", c_int),
        ("eyePositionX", c_int),
        ("eyePositionY", c_int),
        ("eyeDistance", c_int)]

class sample32(Structure):
        _fields_ = [("timestamp", c_double),
        ("leftEye", eye),
        ("rightEye", eye),
        ("planeNumber", c_int)]

leftEye = eye(0, 0, 0, 0, 0, 0)
rightEye = eye(0, 0, 0, 0, 0, 0)
sampleData = sample32(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()
```

This code is taken from the provided programming example, which will give a more detailed insight into the possibilities of iView X™ SDK and its functions.

Using E-Prime

E-Prime does not allow other programs to do any visualization. Therefore no visualization may be done by iView X™ SDK when using E-Prime. Instead the according E-Prime experiment must provide these visualizations on its own. These provided E-Prime examples show how to do this.

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

```
Declare Function iV_Connect Lib "iviewxapi_ep.dll" (ByVal SendIPAddress As String, ByVal send_port As Long,
ByVal RecvIPAddress As String, ByVal read_port As Long) As Long

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

Type EyeDataStruct
        gaze_x As Double
        gaze_y As Double
        diam As Double
        eyePos_x As Double
        eyePos_y As Double
        eyePos_z As Double
End Type

Type SampleStruct32
        timestamp As Double
        eyeData_left As EyeDataStruct
        eyeData_right As EyeDataStruct
        nPlane As Long
End Type

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


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

```
Dim ret As Long

Dim Send_IP_address as String
Dim Recv_IP_address as String
Dim send_port As Long
Dim read_port As Long

send_port = 4444
read_port = 5555
Send_IP_address = "127.0.0.1"
Recv_IP_address = "127.0.0.1"

Dim sample As SampleStruct32

' connect to iView X
ret = iV_Connect (Send_IP_address, send_port, Recv_IP_address, read_port)
ret = iV_GetSample32 (sample)
ret = iV_Disconnect
```

This code is taken from the provided programming example, which will give a more detailed insight into the possibilities of iView X™ SDK and its functions.

iView X SDK Reference

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
- CalibrationPointStruct
- EventStruct
- EventStruct32
- EyeStruct
- SampleStruct
- SampleStruct32
- SystemInfoStruct
- CalibrationStruct

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_ContinueRecording (char etMessage[256])
int iV_DisableGazeDataFilter()
int iV_Disconnect ()
int iV_EnableGazeDataFilter()
int iV_GetAccuracy (struct AccuracyStruct *accuracyData, int visualization)
int iV_GetActualTimestamp (int64 *actualTimestamp)
int iV_GetCurrentCalibrationPoint (struct CalibrationStruct *actualCalibrationPoint)
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_IsConnected ()
int iV_LoadCalibration (char name[256])
int iV_Log (char logMessage[256])
int iV_PauseRecording ()
int iV_Quit()
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)
void iV_SetEventCallback (pDLLSetEvent pEvent)
int iV_SetEventDetectionParameter (int minduration, int maxDispersion)
void iV_SetSampleCallback (pDLLSetSample pSample)
int iV_SetLicense(char key[16])
int iV_SetLogger (int status, char filename[256])
int iV_SetTrackingParameter (int ET_PARAM_EYE, int ET_PARAM, int value)
int iV_SetupCalibration (struct CalibrationStruct *CalibrationData)
int iV_SetupREDStandAloneMode (struct REDStandAloneModeStruct standAloneModeGeometry)
int iV_ShowEyeImageMonitor ()
int iV_ShowSceneVideoMonitor()
int iV_ShowTrackingMonitor ()
int iV_Start()
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_RELFEX_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.

Data Fields

double deviationLX
double deviationLY
double deviationRX
double deviationRY

Detailed Description

If a validation has been performed this struct contains the following information:

- 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.

Data Fields

int number
int positionX
int positionY

Detailed Description

This struct contains the following information:

- 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

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 information in “EventStruct” use function iV_GetEvent.

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 information in “EventStruct32” use function iV_GetEvent32.

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

The struct contains the following information:

- gazeX: horizontal gaze position [pixel]
- gazeY: vertical gaze position [pixel]
- diam: pupil diameter [pixel, mm]
- 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.

SampleStruct Reference

This struct provides information about gaze data samples.

Data Fields

long long timestamp
EyeStruct leftEye
EyeStruct 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 "SampleStruct" use function iV_GetSample.

SampleStruct32 Reference

This struct provides information about gaze data samples.

Data Fields

double timestamp

EyeStruct leftEye
EyeStruct 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.

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

Detailed Description

The struct contains the following information:

- 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 SDK in use
- API_MinorVersion: minor version number of iView X SDK in use
- API_Buildnumber: build number of iView X SDK 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

The struct contains the following information:

- 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”.

REDStandAloneModeStruct Reference

Use this struct to customize RED stand alone mode.

Data Fields

int stimX
int stimY
int stimHeightOverFloor
int redHeightOverFloor
int redStimDist
int redInclAngle

Detailed Description

The struct contains the following information:

- 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 “REDStandAloneModeStruct” use function “iV_SetupREDStandAloneMode”.

Function Documentation

int iV_AbortCalibration ()

aborts the calibration (only during a calibration or validation)

Parameters:

none	

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

Parameters:

none	

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

Parameters:

none	

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_WRONG_CALIBRATION_METHOD	- eye tracking device required for this calibration method is not connected

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

Chagres the position of a calibration point

Parameters:

number	Selected calibration point
positionX	New X position on screen
positionY	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

int iV_ClearRecordingBuffer ()

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

Parameters:

none	

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_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:

SendIPAddress	IP address of iView X™ computer
SendPort	port being used by iView X SDK for sending data to iView X™
RecvIPAddress	IP address of local computer
ReceivePort	port being used by iView X SDK for receiving data from iView X™

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
ERR_WRONG_PARAMETER	- parameter out of range
ERR_COULD_NOT_CONNECT	- failed to establish connection

int iV_ContinueRecording (char etMessage[256])

pauses 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

Parameters:

etMessage	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

int iV_DisableGazeDataFilter()

disables the raw data filter

Parameters:

none	

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:

none	

Returns:

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

int iV_EnableGazeDataFilter()

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

Parameters:

none	

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:

accuracyData	see reference information for “AccuracyStruct”
visualization	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_WRONG_PARAMETER - parameter out of range

int iV_GetActualTimestamp (int64* actualTimestamp)

requests the internal eye tracker timestamp

Parameters:

actualTimestamp	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_GetCurrentCalibrationPoint (struct CalibrationPointStruct * currentCalibrationPoint)

updates “currentCalibrationPoint” with current calibration point data

Parameters:

currentCalibrationPoint	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_GetEvent (struct EventStruct * eventDataSample)

updates “eventDataSample” with current event data

Parameters:

eventDataSample	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

int iV_GetEvent32 (struct EventStruct32 * eventDataSample)

updates “eventDataSample” with current event data

Parameters:

eventDataSample	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_GetSample (struct SampleStruct * rawDataSample)

updates “rawDataSample” with current eyetracking data

Parameters:

rawDataSample	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

Parameters:

rawDataSample	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_GetSystemInfo (struct SystemInfoStruct * systemInfoData)

updates “systemInfoData” with current system information

Parameters:

systemInfoData	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_IsConnected ()

checks if connection to iView X™ is still established

Parameters:

none	

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”

Parameters:

name	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:

logMessage	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_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:

none	

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™

Parameters:

none	

Returns:

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

int iV_ResetCalibrationPoints()

resets all calibration points to default position

Parameters:

none	

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_SaveCalibration (char name [256])

stores a performed calibration

Parameters:

name	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:

filename	filename of data files being created (.idf: eyetracking data, .avi: scene video data)
description	optional experiment description
user	optional name of test person
overwrite	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_NO_RECORDED_DATA	- recording buffer is empty

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.
Important Note: This command is preliminary and will not be supported in the following versions

Parameters:

ETMessage	iView X™ remote command

Returns:

RET_SUCCESS	- intended functionality has been fulfilled
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:

ETMessage	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)

- „iV_CalibrationCallback“ function will be called if a calibration point has changed, the calibration has been finished or aborted.
- This callback allows drawing a customized calibration routine.

Parameters:

pCalibrationPoint	pointer to CalibrationCallbackFunction

Returns:

none

void iV_SetEventCallback (pDLLSetEvent pEvent)

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

Parameters:

pEvent	pointer to EventCallbackFunction

Returns:

none

int iV_SetEventDetectionParameter (int minDuration, int maxDispersion)

- defines detection parameter for online fixation detection algorithm

Parameters:

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

Returns:

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

void iV_SetSampleCallback (pDLLSetSample pSample)

- „iV_SampleCallback“ 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:

pSample	pointer to SampleCallbackFunction

Returns:

none

int iV_SetLicense(char key[16])

sets the license key (only for OEM customer)

Parameters:

key	Provided license key

Returns:

RET_SUCCESS - intended functionality has been fulfilled

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

defines the logging behavior of iView X SDK

Parameters:

status	logging status, see “Explanations for Defines” in this manual for further information
filename	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

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

sets iView X tracking parameters

Parameters:

ET_PARAM_EYE	select specific eye
ET_PARAM	select parameter that shall be set
value	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 calibration parameters

Parameters:

CalibrationData	see reference information for "CalibrationStruct"

Returns:

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

int iV_SetupREDStandAloneMode (struct REDStandAloneModeStruct standAloneModeGeometry)

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

Parameters:

standAloneModeGeometry	see reference information for "REDStandAloneModeStruct"

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 monitor (available for all devices except RED)

Parameters:

none	

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 dialog (available for HED devices only)

Parameters:

none	

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 (available for RED devices only)

Parameters:

none	

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()

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

Parameters:

none	

Returns:

RET_SUCCESS - intended functionality has been fulfilled
ERR_COULD_NOT_CONNECTED - 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:

none	

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_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:

none	

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_Validate ()

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

Parameters:

none	

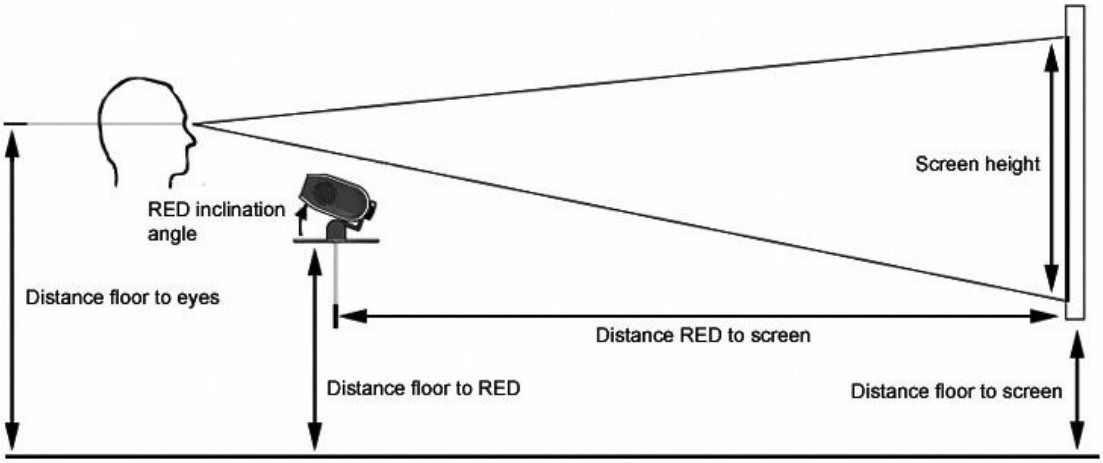
Returns:

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

iView X™ SDK can be used to configure the RED stand-alone mode. The data struct "standAloneModeGeometry" contains all geometrical parameter while the function "iV_SetupREDStandAloneMode" 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 "standAloneModeGeometry"struct
5. Call the function "iV_SetupREDStandAloneMode" including the "standAloneModeGeometry"struct as parameter to iView X™

Return Codes

Each function returns a value that provides status information. The following is a list of all error codes defined.

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.

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_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_NO_RESPONSE_FROM_IVIEW	124	no response from iView X; check iView X connection settings (IP addresses, ports) or last command
ERR_INVALID_IVIEWX_VERSION	125	iView X version could not be resolved
ERR_WRONG_IVIEWX_VERSION	126	wrong version of iView X
ERR_DELETE_SOCKET	131	failed to close sockets
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	failed to start iView X

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.

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

Last updated: January 2011