

Product Description

Tobii Software Development Kit

Version 2.0.2

Contents

CONTENTS	II
1 INTRODUCTION	3
1.1 Contents of the Tobii Software Development Kit	3
1.2 Skill Requirements	3
1.3 Licensing of the SDK	3
2 SOFTWARE OVERVIEW	4
2.1 Interface levels	4
2.2 Schematic Overview	4
3 EYE TRACKER COMPONENTS API	5
4 EYE TRACKER LOW LEVEL API	7
4.1 Introduction	7
4.2 Functions	8
5 CLEARVIEW TRIGGER APIS	9
5.1 Introduction	9
5.2 ClearView Trigger Component API	9
5.3 ClearView Trigger Low Level API	10
5.4 ClearView Trigger Protocol API	11
A CONTENTS DEVELOPER'S GUIDE	12
B DEVELOPER'S GUIDE EXAMPLES	17
Example 1: Tobii Eye Tracker Components API - TetClient Object	17
Example 2: Tobii Eye Tracker Low level API – Function Tet_Connect	20

1

Introduction

This Product Description document aims to give an overview of the contents of the Tobii Software Development Kit (SDK). The Tobii SDK enables development of application software for controlling and retrieving data from the Tobii Eye Trackers. This is useful for highly customized experimental routines as well as many varieties of interaction applications based on eye tracking.

1.1 Contents of the Tobii Software Development Kit

The Tobii SDK contains the following elements:

Application programming interfaces (APIs)

The SDK provides interfaces on different levels, which are suitable for different kinds of applications – ranging from low level interfaces with high level of customization, to high level interfaces which require a minimum of programming effort. The three API levels are:

- Tobii Eye Tracker Components APIs (see Chapter 3)
- Tobii Eye Tracker Low Level APIs (see Chapter 4)
- ClearView Trigger APIs (see Chapter 5)

Code samples

Well documented code samples provide a straight-forward introduction to the functionality of the SDK. Each of the examples illustrates most of the available functionality of the respective APIs. The following samples are included:

- Eye Tracker Components, C# example
- Eye Tracker Components, VB6 example
- Eye Tracker Components, C++ example
- ClearView Trigger Component, VB6 example
- ClearView Trigger Component, C# example
- ClearView Trigger Low-level, C example

Developer's Guide

A comprehensive Developer's Guide provides documentation, explanations and code snippets for all functions of the Tobii SDK. The notation in the Developer's focuses on the Microsoft Visual Basic 6.0 (VB 6) and/or Microsoft Visual C/C++ (VC++) syntax.

Please refer to Appendix A for a table of contents of the Developer's Guide.

1.2 Skill Requirements

The SDK itself aims to target developers that not necessarily need a lot of programming experience. Just the basics. Especially, if the Microsoft Visual Basic 6.0 programming language is used and the included samples are walked through, it should be fairly easy to develop an application retrieving gaze data.

1.3 Licensing of the SDK

A special license which is charged separately is required to gain access to the Tobii SDK.

Distribution of software for internal use and for research purposes, which incorporate Tobii APIs and components from the Tobii SDK, is free of charge. For commercial distribution of software that utilizes APIs and components from the Tobii SDK, a separate licensing arrangement with Tobii Technology is required.

2 Software Overview

2.1 Interface levels

The Tobii SDK provides interfaces on three different levels:

High Level Interface The high level interface (TET Comp API) is suitable for designing customized applications in a fast and easy way. It offers a set of ready-made COM and ActiveX objects for fully automatic collection of gaze data and ready-to-use calibration procedure, track status and calibration plotting tools.

Low Level Interface The low level interfaces (TET and Ttime APIs) provide full control of the actual eye tracking and the calibration procedure. The application communicates directly with the Tobii Eye Tracker API and if needed the Tobii Time API.

Trigger Interface The trigger interface (TCVTrigger API) enables control of and the sending of signals to the ClearView analysis software.

2.2 Schematic Overview

Figure 2.1 gives a schematic overview of all possible combinations of ways to access Tobii software programmatically. At the bottom, there is the eye tracker hardware and the TETServer controlling it.

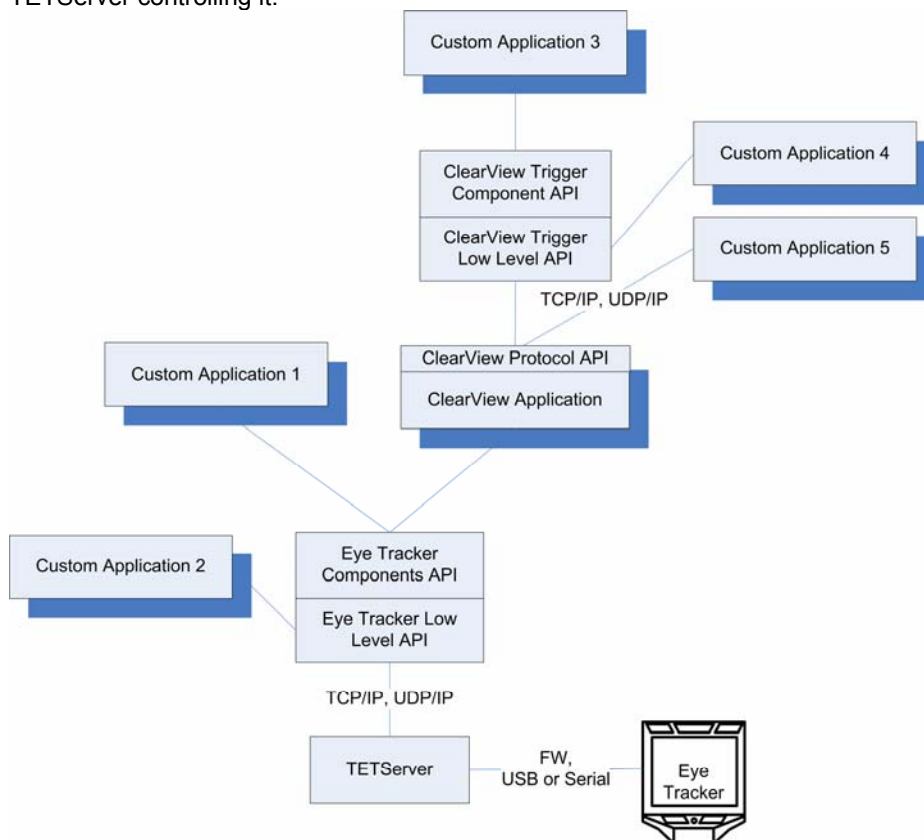


Figure 2.1. All APIs, the applications that can be built upon them and how they relate.

3 Eye Tracker Components API

The *Tobii Eye Tracker Components API (TetComp)* is an interface containing everything you need to programmatically get real-time high-precision gaze data from Tobii eye tracker hardware. It is a type library implemented as a set of COM objects, making it accessible to many modern high-level program languages for Microsoft 32-bit platforms.

The library provides the highest software abstraction layer provided by Tobii and gains access to all functionality of the Tobii eye tracker hardware. Internally it is using the *Tobii Eye Tracker Low Level API*, described in the next chapter.

The *Tobii Eye Tracker Components API* is the natural first choice of the two eye tracker APIs for a developer, since it is hiding much of the programming complexity of the *Tobii Eye Tracker Low Level API*. It also provides ready made GUI tools to calibrate and to display tracking ability. Such tools can of course be implemented from scratch, but why reinvent the wheel.

The eye tracker may reside on any host as long as there is IP connectivity to the host running the application that uses *TetComp*.

The *TetComp* objects are divided into following categories:

- Tracking component. Use the *TetClient* object to get gaze data.
- Real time display of the tracking ability for the current subject. This is helpful to make sure the subject is being tracked properly and is positioned well relative the eye tracker sensor. The *TetTrackStatus* object provides this tool.
- Calibration tools: The eye tracker must be calibrated for each subject either by creating a new calibration or using a saved calibration. The *TetCalibProc*, *TetCalibPlot* and *TetCalibManager* objects provides tools for this process.
- Utilities and help objects: Some basic Tobii time stamp calculations are provided with the *TetTimeUtilities* object. The *TetCalibAnalyzeDataArray*, *TetPointDArray* and *TetGazeDataHolder* are help object used by other objects to represent data.

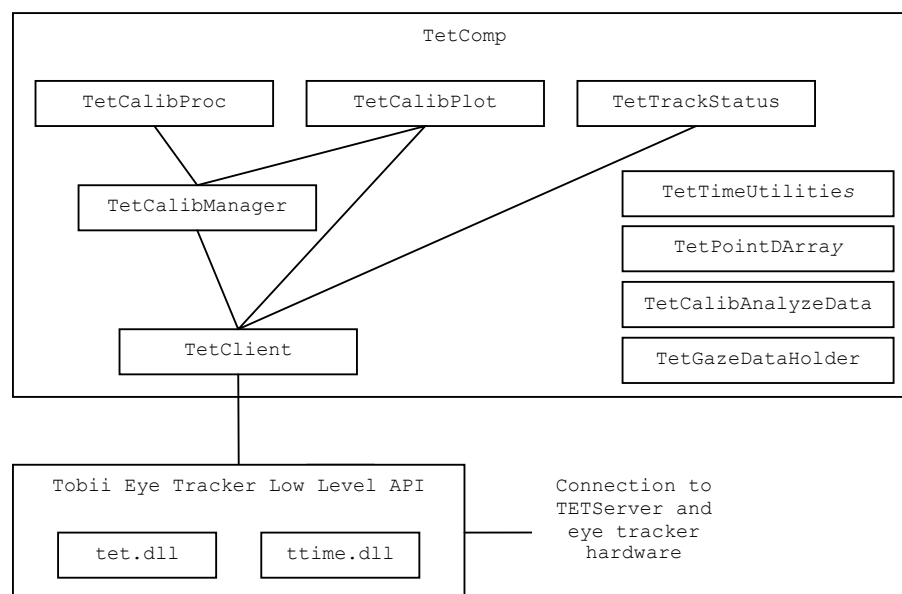


Figure 3.1. The *TetComp* objects internal dependencies, and how they communicate with

the TETServer Indirectly via the low level API.

Below follows a brief description for each of the *TetComp* objects:

TetClient - Handles the calls to the lower software abstraction layer. Thus, it exposes the full functionality of the TETServer and it is possible to build a complete eye tracking application by using this object only.

TetTrackStatus - It shows the tracking ability of the subject being tracked. It is a good tool for confirming that the subject is sitting in an advantageous position. It is implemented as an ActiveX control.

TetCalibProc – This object is used to calibrate the subject. To do that it opens its own window to displays an appropriate calibration stimulus.

TetCalibPlot - It displays the result of a calibration, and can be used to provide information to decide if the calibration should be accepted, rejected or improved. It is implemented as an ActiveX control.

TetCalibManager - It can be used when developing a new calibration tool. It exposes the *TetClient* functionality needed to perform a calibration, and also has some own functionality for deciding the optimal positions of calibration points and how to improve a calibration.

TetTimeUtilities – Tobii provides a time stamp that can be shared among different threads, processes and hosts. This object handles some basic time stamp calculations.

TetPointDArray – a data storage object for calibration points.

TetCalibAnalyzeDataArray - a data storage object for calibration information.

TetGazeDataHolder – a help object passed by the gaze data events when languages like VB 6 are used.

Please refer to Appendix B for an example illustrating how the TetClient Object is documented in the Developer's Guide.

4 Eye Tracker Low Level API

4.1 Introduction

This API is a set of function calls that controls Tobii Eye Tracker hardware, retrieves real time gaze data and provides a high-resolution time format accessible from any application.

It is the Eye Tracker API of choice when COM objects are not an option. It is implemented as two traditional Windows DLLs (tet.dll and ttime.dll) which makes it available to most programming languages running on the Windows 32-bit systems.

This API is a set of function calls that handles the connection to and the communication with the Tobii Eye Tracker Server (TETServer) application, which in turn is connected to the eye tracker hardware. Internally, the implementation uses two Tobii proprietary TCP/IP and UDP/IP communications protocols.

The API also provides a high-resolution time stamp that is accessible from any number of applications and threads, even running on different hosts. This time is compatible with the time stamped gaze data.

One of the issues to consider before choosing this API is the effort required to make GUI applications work smoothly. Eye tracking at this API level is blocking and passes data to a callback function, which means that GUI updates and user input must be performed either in the callback or in a different thread. Since many eye tracking applications need near real time performance and/or a high degree of synchronization between user input, gaze data arrival and GUI updates, these two alternatives will add a significant source code complexity, compared to the use of the *Tobii Eye Tracker Components API*.

Warning! Always consider the higher level Tobii Eye Tracker Components API as the first choice to gain access the Tobii eye tracker hardware. It hides programming complexity and adds GUI tools that must be implemented when using this Tobii Low Level Eye Tracker API.

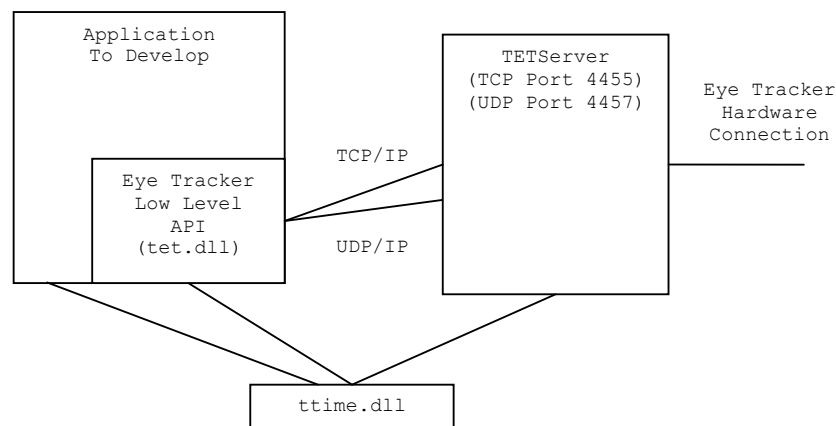


Figure 4.1. Application using the Eye Tracker Low Level API. If the Application To Develop and TETServer is running on the same host, they share the same ttime.dll as shown in the figure.

4.2 Functions

Program code that is used in this chapter is ANSI C code. It is assumed that the reader has a basic knowledge of the C programming language.

This is a description of the available function calls.

Function Call	DLL	Description
Tet_Connect	tet	Connect to an eye tracker and perform internal initializations.
Tet_Disconnect	tet	Disconnect from an eye tracker and perform internal cleanups.
Tet_Start	tet	Start the eye tracker to receive gaze data and to receive optional timer events in a callback. Blocking.
Tet_Stop	tet	Stop the eye tracker.
Tet_CallbackFunction	tet	User defined callback function that is called whenever new gaze data is available or whenever there is a timer callback event.
Tet_CalibLoadFromFile	tet	Load the eye tracker with a calibration stored in a file.
Tet_CalibSaveToFile	tet	Save the calibration in use to file.
Tet_CalibClear	tet	Remove all the new calibration points.
Tet_CalibAddPoint	tet	Start the sampling of gaze data for a new calibration. Receive optionally gaze data and timer events in a callback. Blocking.
Tet_CalibCalculateAndSet	tet	End the calibration process and set the calibration in use to the new calibration points.
Tet_CalibGetResult	tet	Get information about a calibration. The source may be either the one in use or one stored in a file.
Tet_CalibRemovePoints	tet	Remove points within a given area from the set of new calibration points.
Tet_PerformSystemCheck	tet	Verify the eye tracker system is ok.
Tet_GetSerialNumber	tet	Get the serial numbers of eye tracker hardware components.
Tet_GetLastError	tet	Get the Tobii error code for latest error.
Tet_GetLastErrorAsText	tet	Get the Tobii error description for latest error.
Tet_UseDebugLog	tet	Turn on tet.dll logging.
TT_IsSupported	ttime	Check if the hardware supports the functionalities required by ttime.
TT_GetLocalTimeStamp	ttime	Get the local time stamp. Compatible time stamp when called from any thread or process using the same ttime DLL on a host.
TT_GetServerTimeStamp	ttime	Get the server time stamp. Compatible time stamp when called from any thread or process on any host that has made a successful call to Tet_Connect to the same server using the TET_SYNC_SERVER or TET_SYNC_LOCAL parameters.
TT_ElapsedTime	ttime	Calculate the difference between two time stamps.
TT_AddTime	ttime	Calculate the sum of two time stamps.
TT_AverageTime	ttime	Calculate the mean value of two time stamps.
TT_Compare	ttime	Compare two time stamps.
TT_GetCurrentTime	ttime	Get the current system time. Not compatible with ttime time stamps.

Table 4.1. Tobii Eye Tracker Low Level API Function Calls Quick Reference.

Struct	Functions using the struct	Description
STet_GazeData	Tet_CalibAddPoint , Tet_Start	All data for one gaze point sample.
STet_CalibAnalyzeData	Tet_CalibGetResult	Information about all points for a certain calibration.

Table 4.2. Tobii Eye Tracker Low Level API Structs Quick Reference.

5

ClearView Trigger APIs

5.1 Introduction

The historical background of *the ClearView Trigger APIs* is that it was implemented when some ClearView operators found that they interfered with the recording subject when they needed access to the same computer as the subject were using. Others were requesting a more controlled way of starting a recording.

Therefore, a minor subset of the ClearView tasks is implemented and the intention is to be able to start, stop recording, log information during recording and send an event that may, for example, trig a slide change in a slide show stimuli.

See the Tobii ClearView manual for ClearView details.

The application developed using the APIs can be run on the same host as the ClearView application or on any another host that has a TCP/IP stack implemented and IP network access to the ClearView host.

There are three *ClearView Trigger API* implementations providing equivalent functionality at different software levels:

- The highest level is the *ClearView Component API*. This API is implemented as a Microsoft COM API and can be used by a number of common environments supporting COM objects on the Windows platform.
- Next level is the *ClearView Trigger Low Level API*. An API accessible from Windows 32-bit environments. It is implemented as a traditional Windows DLL and can be used by any language that supports DLL calls.
- The lowest level is the *ClearView Trigger Protocol API*, which is the protocol API at the communication level. The advantage of this API is that it is not bound to any specific operating system.

For highest level of abstraction and for fewer implementation details, choose the highest level that suits the needs of your development project.

5.2 ClearView Trigger Component API

This is the trigger API *CVTrigComp* implemented as a Microsoft COM component, making it accessible to many modern high-level program languages for the Microsoft 32-bit platforms.

The *CVTrigComp* exposes the *CVTrig* COM class (with interface *ICVTrig*) and is part of the *Tobii CVTrigComp 1.0 Type Library*.

The API is a set of methods that handles the connection to and the communication with the ClearView application. Internally, it is basically a wrapper of the *ClearView Trigger Low Level API* with a COM interface.

Physically it is provided as the file *CVTrigComp.dll*. It requires a standard COM component installation procedure before it can be used. During installation of the SDK, this procedure is performed. For deployment without installing the SDK, see the installation section later in this chapter.

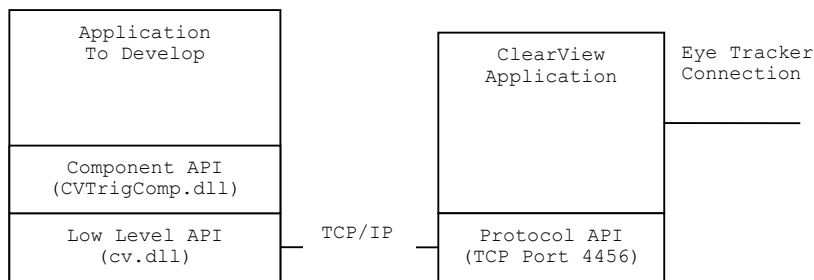


Figure 5.1. Application using the ClearView Trigger Component API

5.2.1 The ICVTrig Interface Methods

Program codes that are used in this chapter are Microsoft Visual Basic 6.0 and Microsoft C++ .NET compatible code. It is assumed that the reader has a basic understanding of one of the languages. It should be quite intuitive though, using other languages.

This is a description of the API methods.

Method	Description
Connect	Connect to ClearView and perform internal initializations.
Disconnect	Disconnect and perform internal cleanups.
Start	Start a recording.
StartWithName	Start a named recording.
Stop	Stop an ongoing recording.
LogEvent	Add a text data record to an ongoing recording.
SendGenericEvent	Send a generic event, which will trig some ClearView stimulus to certain actions during a recording. See ClearView manual for details.

Table 5.1. ICVTrig Methods Quick Reference

5.3 ClearView Trigger Low Level API

This is the ClearView Trigger API of choice when COM objects are not an option. It is implemented as a traditional Windows DLL (cv.dll) which makes it available to most programming languages running on the Windows 32-bit systems.

This API is a set of function calls that handles the connection to and the communication with the ClearView application. Internally, it is a C/C++ implementation using the *ClearView Trigger Protocol API*.

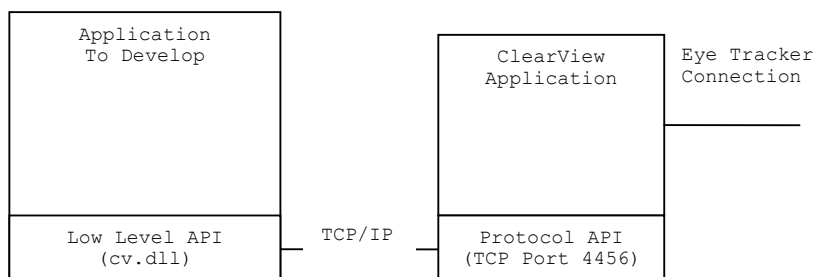


Figure 5.2. Application using the ClearView Trigger Low Level API.

5.3.1 Functions

Program code that is used in this chapter is ANSI C code. It is assumed that the reader has a basic knowledge of the C programming language.

This is a description of the API as declared in the header file *cv.h*.

Function Call	Description
CV_Init	Connect to ClearView and performs internal initializations.
CV_Close	Disconnect from ClearView and perform internal cleanups.
CV_Start	Start a recording.
CV_StartWithName	Start a named recording.
CV_Stop	Stop an ongoing recording.
CV_LogEvent	Add a text data record to an ongoing recording.
CV_SendGenericEvent	Send a generic event, which will trig some ClearView stimulus to certain actions during a recording. See ClearView manual for details.
CV_GetLastError	Get error details if any function call returned error.

Table 5.2. ClearView Trigger Low Level API Function Calls Quick Reference

5.4 ClearView Trigger Protocol API

This is the lowest software level possible that can be used to gain access to the ClearView Trigger functionality.

Normally there is no benefit of using the protocol if the target platforms are any of the supported Windows client platforms. Preferably use the *ClearView Trigger Component API* or *ClearView Trigger Low Level API* instead. However, if the client application must run on an unsupported operating system or if the programming language to use cannot call the other APIs for some reason, this interface is an option.

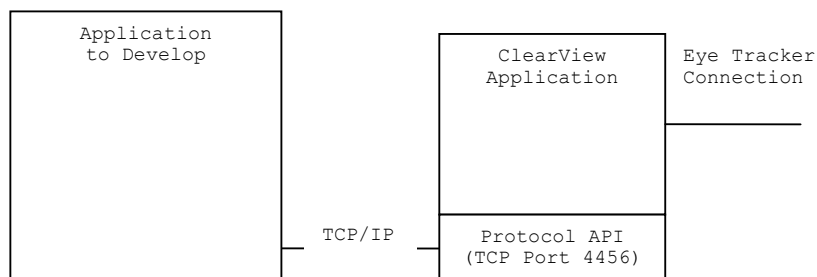


Figure 5.3. Application using the ClearView Trigger Protocol API.

Any programming language keywords that are used in this chapter are ANSI C keywords.



Contents Developer's Guide

1	Introduction	6
1.1	Document History	6
1.2	Skill Requirements	6
1.3	Programming Languages in Documentation	6
1.4	Compatibility	7
1.5	Help Improving the Document and SDK	7
1.6	System Requirements and Compatibility	7
1.6.1	Eye Tracker APIs	7
1.6.2	ClearView APIs	8
1.7	Abbreviations, Acronyms and Definitions	8
2	Software Overview	10
2.1	System Overview	10
2.2	What Interface to Select	11
3	Eye Tracker Components API	13
3.1	Introduction	13
3.1.1	Which Components to Use	15
3.2	Requirements	15
3.3	Installation and Deployment	15
3.4	Revision History	16
3.5	TetClient Object	16
3.5.1	COM Characteristics and Interfaces	16
3.5.2	Instantiating the Object in VB 6	17
3.5.3	The ITetClient Interface Overview	17
3.5.4	Method TetClient.Connect	18
3.5.5	Method TetClient.Disconnect	19
3.5.6	Method TetClient.StartTracking	19
3.5.7	Method TetClient.StopTracking	21
3.5.8	Method TetClient.GetNumPendingPostGazeData	22
3.5.9	Method TetClient.GetTimeStamp	22
3.5.10	The Calibration Process	23
3.5.11	Method TetClient.LoadCalibrationFromFile	23
3.5.12	Method TetClient.SaveCalibrationToFile	23
3.5.13	Method TetClient.ClearCalibration	24
3.5.14	Method TetClient.AddCalibrationPoint	24
3.5.15	Method TetClient.InterruptAddCalibrationPoint	26
3.5.16	Method TetClient.CalculateAndSetCalibration	27
3.5.17	Method TetClient.GetCalibrationResult	27
3.5.18	Method TetClient.RemoveCalibrationPoints	28
3.5.19	Method TetClient.PerformSystemCheck	29
3.5.20	Method TetClient.GetSerialNumber	29
3.5.21	Property TetClient.IsConnected	30
3.5.22	Property TetClient.IsTracking	30
3.5.23	Property TetClient.ServerAddress	30
3.5.24	Property TetClient.PortNumber	30
3.5.25	Property TetClient.SynchronizationMode	31
3.5.26	Property TetClient.IsAddingCalibrationPoint	31
3.5.27	Property TetClient.GazeDataDelivery	31
3.5.28	TetClient Event OnTrackingStarted	32
3.5.29	TetClient Event OnTrackingStopped	32
3.5.30	TetClient Event TetClient.OnGazeData	33
3.5.31	TetClient Event OnPostGazeData	33
3.5.32	TetClient Event OnAddCalibrationPointStarted	33

3.5.33	TetClient Event OnAddCalibrationPointEnded	34
3.5.34	TetClient Event OnCalibrationGazeData	34
3.5.35	TetClient Event OnPostCalibrationGazeData	35
3.6	TetTrackStatus Object	35
3.6.1	COM Characteristics and Interfaces	36
3.6.2	Instantiating the Object in VB 6	36
3.6.3	The ITetTrackStatus and _ITetTrackStatusEvents Interfaces Overview	37
3.6.4	Method TetTrackStatus.Connect37	
3.6.5	Method TetTrackStatus.Disconnect	38
3.6.6	Method TetTrackStatus.Start	38
3.6.7	Method TetTrackStatus.Stop	38
3.6.8	Property TetTrackStatus.IsConnected	39
3.6.9	Property TetTrackStatus.IsTracking	39
3.6.10	Property TetTrackStatus.ServerAddress	39
3.6.11	Property TetTrackStatus.PortNumber	39
3.6.12	Property TetTrackStatus.TextColor	40
3.6.13	Property TetTrackStatus.BackgroundColor	40
3.6.14	Property TetTrackStatus.EyeColor	40
3.6.15	Property TetTrackStatus.TextFont	41
3.6.16	TetTrackStatus Event OnStopped	41
3.7	TetCalibProc Object	41
3.7.1	COM Characteristics and Interfaces	42
3.7.2	Instantiating the Object in VB 6	42
3.7.3	The ITetCalibProc Interface Overview	42
3.7.4	Method TetCalibProc.Connect	43
3.7.5	Method TetCalibProc.Disconnect	44
3.7.6	Method TetCalibProc.StartCalibration	44
3.7.7	Method TetCalibProc.ContinueCalibration	45
3.7.8	Method TetCalibProc.InterruptCalibration	45
3.7.9	Property TetCalibProc.IsConnected	46
3.7.10	Property TetCalibProc.IsCalibrating	46
3.7.11	Property TetCalibProc.ServerAddress	46
3.7.12	Property TetCalibProc.PortNumber	47
3.7.13	Property TetCalibProc.NumPoints	47
3.7.14	Property TetCalibProc.CalibArea	47
3.7.15	Property TetCalibProc.CalibManager	48
3.7.16	Property TetCalibProc.ContinueCalibrationAutomatically	48
3.7.17	Property TetCalibProc.DisplayMonitor	49
3.7.18	Property TetCalibProc.WindowTopmost	49
3.7.19	Property TetCalibProc.WindowVisible	49
3.7.20	Property TetCalibProc.PointSpeed	50
3.7.21	Property TetCalibProc.PointSize50	
3.7.22	Property TetCalibProc.PointColor	51
3.7.23	Property TetCalibProc.BackgroundColor	51
3.7.24	TetCalibProc Event OnCalibrationEnd	51
3.7.25	TetCalibProc Event OnKeyDown	52
3.8	TetCalibPlot Object	52
3.8.1	COM Characteristics and Interfaces	53
3.8.2	Instantiating the Object in VB 6	54
3.8.3	The ITetCalibPlot and _ITetCalibPlotEvents Interfaces Overview	54
3.8.4	Method TetCalibPlot.Connect	54
3.8.5	Method TetCalibPlot.Disconnect	55
3.8.6	Method TetCalibPlot.SetData	55
3.8.7	Method TetCalibPlot.UpdateData	56
3.8.8	Method TetCalibPlot.ClearData	56
3.8.9	Property TetCalibPlot.IsConnected	57
3.8.10	Property TetCalibPlot.ServerAddress	57
3.8.11	Property TetCalibPlot.PortNumber	57
3.8.12	Property TetCalibPlot.Eye	57
3.8.13	Property TetCalibPlot.CalibManager	58
3.8.14	Property TetCalibPlot.SelectedPoints	58
3.8.15	Property TetCalibPlot.AllowMouseInteraction	59

3.8.16	Property TetCalibPlot.PointColor	59
3.8.17	Property TetCalibPlot.SelectedColor	59
3.8.18	Property TetCalibPlot.EyeLeftColor	60
3.8.19	Property TetCalibPlot.EyeRightColor	60
3.8.20	Property TetCalibPlot.BackgroundColor	60
3.8.21	TetCalibPlot Event OnSelectedPointsChanged	61
3.9	TetCalibManager Object	61
3.9.1	COM Characteristics and Interfaces	63
3.9.2	Instantiating the Object in VB 6	63
3.9.3	The ITetCalibManager Interface Overview	63
3.9.4	Method TetCalibManager.Connect	64
3.9.5	Method TetCalibManager.Disconnect	65
3.9.6	Method TetCalibManager.LoadCalibrationFromFile	65
3.9.7	Method TetCalibManager.SaveCalibrationToFile	66
3.9.8	Method TetCalibManager.GetManagerFilePath	66
3.9.9	Method TetCalibManager.BeginCalibrationProcess	67
3.9.10	Method TetCalibManager.GetCalibrationProcessPoint	68
3.9.11	Method TetCalibManager.ProcessPoint	68
3.9.12	Method TetCalibManager.InterruptCalibrationProcess	69
3.9.13	Method TetCalibManager.GetCalibPoints	69
3.9.14	Method TetCalibManager.GetIdealCalibPoints	70
3.9.15	Method TetCalibManager.GetRecalibPoints	70
3.9.16	Method TetCalibManager.SetRecalibPoints	71
3.9.17	Method TetCalibManager.SetDefaultRecalibPoints	71
3.9.18	Method TetCalibManager.SetNumRecalibPoints	72
3.9.19	Method TetCalibManager.GetRemovePoints	72
3.9.20	Method TetCalibManager.SetRemovePoints	72
3.9.21	Method TetCalibManager.SetDefaultRemovePoints	73
3.9.22	Method TetCalibManager.RemoveCalibrationPoints	73
3.9.23	Method TetCalibManager.GetAnimationPoints	74
3.9.24	Property TetCalibManager.NumPoints	74
3.9.25	Property TetCalibManager.CalibArea	75
3.9.26	Property TetCalibManager.UseIdealCalibGrid	75
3.9.27	Property TetCalibManager.IsConnected	76
3.9.28	Property TetCalibManager.IsProcessing	76
3.9.29	Property TetCalibManager.ServerAddress	76
3.9.30	Property TetCalibManager.PortNumber	77
3.9.31	TetCalibManager Event OnPointProcessed	77
3.9.32	TetCalibManager Event OnCalibrationProcessEnded	77
3.9.33	TetCalibManager Event OnCalibrationSet	78
3.9.34	TetCalibManager Event OnCalibPointsChanged	78
3.9.35	TetCalibManager Event OnRecalibPointsChanged	78
3.9.36	TetCalibManager Event OnRemovePointsChanged	79
3.10	TetGazeDataHolder Object	79
3.10.1	COM Characteristics and Interfaces	79
3.10.2	The ITetGazeDataHolder Interface Overview	79
3.10.3	Method TetGazeDataHolder.GetGazeData	80
3.10.4	Method TetGazeDataHolder.SetGazeData	80
3.11	TetCalibAnalyzeDataArray Object	81
3.11.1	COM Characteristics and Interfaces	81
3.11.2	The ITetCalibAnalyzeDataArray Interface Overview	81
3.11.3	Method TetCalibAnalyzeDataArray.GetAt	81
3.11.4	Method TetCalibAnalyzeDataArray.Set	82
3.11.5	Property TetCalibAnalyzeDataArray.Size	82
3.12	TetPointDArray Object	82
3.12.1	COM Characteristics and Interfaces	83
3.12.2	The ITetPointDArray Interface Overview	83
3.12.3	Method TetPointDArray.Add	83
3.12.4	Method TetPointDArray.Remove	83
3.12.5	Method TetPointDArray.Clear	84
3.12.6	Method TetPointDArray.GetAt	84
3.12.7	Method TetPointDArray.AddArray	84

3.12.8	Property TetPointDArray.Size	85
3.13	TetTimeUtilities Object	85
3.13.1	COM Characteristics and Interfaces	85
3.13.2	The ITetTimeUtilities Interface Overview	85
3.13.3	Method TetTimeUtilities.ElapsedTime	86
3.13.4	Method TetTimeUtilities.AddTime	86
3.13.5	Method TetTimeUtilities.AverageTime	87
3.13.6	Method TetTimeUtilities.Compare	88
3.13.7	Method TetTimeUtilities.GetCurrentTime	88
3.14	TetComp Type Definitions	89
3.14.1	Struct TetComp.TetGazeData	89
3.14.2	Struct TetComp.TetCalibAnalyzeData	90
3.14.3	Struct TetComp.TetTimeStamp	91
3.14.4	Struct TetComp.TetTime	91
3.14.5	Struct TetComp.TetPointD	92
3.14.6	Struct TetComp.TetRectF	92
3.15	TetComp Error Codes	93
4	Eye Tracker Low Level API	95
4.1	Introduction	95
4.2	Requirements	96
4.3	Installation and Deployment	96
4.4	Important Programming Hints	96
4.5	Functions	97
4.5.1	Function Tet_Connect	98
4.5.2	Function Tet_Disconnect	99
4.5.3	Function Tet_Start	99
4.5.4	Function Tet_Stop	100
4.5.5	Callback Function Tet_CallbackFunction	101
4.5.6	The Calibration Process	102
4.5.7	Function Tet_CalibLoadFromFile	103
4.5.8	Function Tet_CalibSaveToFile	104
4.5.9	Function Tet_CalibClear	104
4.5.10	Function Tet_CalibAddPoint	104
4.5.11	Function Tet_CalibCalculateAndSet	106
4.5.12	Function Tet_CalibGetResult	107
4.5.13	Function Tet_CalibRemovePoints	108
4.5.14	Function Tet_PerformSystemCheck	109
4.5.15	Function Tet_GetSerialNumber	109
4.5.16	Function Tet_GetLastError	110
4.5.17	Function Tet_GetLastErrorAsText	110
4.5.18	Function Tet_UseDebugLog	111
4.5.19	Function TT_IsSupported	111
4.5.20	Function TT_GetLocalTimeStamp	112
4.5.21	Function TT_GetServerTimeStamp	112
4.5.22	Function TT_ElapsedTime	113
4.5.23	Function TT_AddTime	114
4.5.24	Function TT_AverageTime	114
4.5.25	Function TT_Compare	115
4.5.26	Function TT_GetCurrentTime	115
4.5.27	Struct STet_GazeData	116
4.5.28	Struct STet_CalibAnalyzeData	116
4.5.29	Struct STimeStamp	117
4.5.30	Struct STime	118
5	ClearView Trigger APIs	119
5.1	Introduction	119
5.2	ClearView Trigger APIs Will Change	119
5.3	Change History List	120
5.4	ClearView Trigger Component API	120
5.4.1	Requirements	120
5.4.2	Installation and Deployment	121

5.4.3	Notification and Programming Language Considerations	121
5.4.4	The ICVTrig Interface Methods	122
5.4.5	Method Connect	122
5.4.6	Method Disconnect	122
5.4.7	Method Start	123
5.4.8	Method StartWithName	123
5.4.9	Method Stop	123
5.4.10	Method LogEvent	124
5.4.11	Method SendGenericEvent	124
5.4.12	Return Codes and Exceptions	125
5.5	ClearView Trigger Low Level API	125
5.5.1	Requirements	126
5.5.2	Installation and Deployment	126
5.5.3	Important Programming Hints	126
5.5.4	Functions	126
5.5.5	Function CV_Init	127
5.5.6	Function CV_Close	127
5.5.7	Function CV_Start	128
5.5.8	Function CV_StartWithName	128
5.5.9	Function CV_Stop	129
5.5.10	Function CV_LogEvent	130
5.5.11	Function CV_SendGenericEvent	130
5.5.12	Function CV_GetLastError	130
5.6	ClearView Trigger Protocol API	131
5.6.1	Requirements	131
5.6.2	Installation and Deployment	132
5.6.3	Programming Hints	132
5.6.4	Message Flow	132
5.6.5	Message Format	132
5.6.6	The type Field	133
5.6.7	The command Field	133
5.6.8	The errorcode Field	133
5.6.9	The data Field	134
5.6.10	Examples	134
5.6.11	C/C++ Sample Header File	135
Appendix A. Gaze Data		136
Appendix B. Time Synchronization		139

B Developer's Guide examples

The following examples illustrate the content and structure of the Developer's Guide document:

- Example 1: Tobii Eye Tracker Components API - TetClient Object
- Example 2: Tobii Eye Tracker Low level API - Tet_Connect function

Example 1: Tobii Eye Tracker Components API - TetClient Object

The *TetClient* is a COM object that enables an application to communicate with the *TETServer* and thereby the Tobii eye tracker hardware. The *TetClient* is basically a *Tobii Eye Tracker Low Level API* wrapper. A difference is that the concept of event is added. The *TetClient* enables developers to do things such as receive gaze data and perform a custom calibration in a more convenient way than using the low level functions. The information about the user's gaze is exposed as regular COM events which are normally fired with the maximal frequency that the eye tracker hardware supports. For more information about this event, see the *GazeDataDelivery* property.

5.4.1 COM Characteristics and Interfaces

Here is some interface details for the developers using languages like VC++. For others (VB 6 developers), most of it is handled by the language environment.

- For VB 6, the exposed interface is the *TetClient* interface.
- COM Characteristics - Apartment threaded.
- The functionality of the *TetClient* relies on the fact that there exists a message loop in the *TetClient* thread, which is a requirement for any COM enabled thread.
- Exposed interface – *ITetClient*: Main custom interface. Supports *IErrorInfo*.
- Exposed outgoing interfaces - *_ITetClientEvents*: Main custom outgoing interface. Use this if your programming language supports it. *DTetClientEvents*: Default outgoing dispinterface. Having the same members as *_ITetClientEvents*. Less efficient though, but supports the VB 6 environment.

All events described further in this chapter refer to the members in both *_ITetClientEvents* and *DTetClientEvents*. When an event is fired it will first be fired from *_ITetClientEvents* and then the corresponding event in *DTetClientEvents* will be fired.

5.4.2 Instantiating the Object in VB 6

To instantiate the COM object in your Visual Basic 6.0 environment, read the Microsoft manual or follow the hints below:

Make sure the component is installed on the computer. There is a separate section in this chapter of how to do that.

Add a library reference to the Tobii Eye Tracker Components 1.X Type Library.

Declare and instantiate with:

```
Dim WithEvents m_tetClient As TetClient
Set m_tetClient = New TetClient
```

5.4.3 The ITetClient Interface Overview

Below is an overview of the *ITetClient* interface. All methods, properties and events are listed. Error codes are common for all *TetComp* objects and are listed in the Developer's Guide.

Method	Description
Connect	Connect to an eye tracker.
Disconnect	Disconnect from an eye tracker.
StartTracking	Start gaze tracking.
StopTracking	Stop gaze tracking.
GetNumPendingPostGazeData	Tell how many gaze data samples there are left in the gaze data queue.
GetTimeStamp	Get current Tobii time stamp.
LoadCalibrationFromFile	Load the eye tracker with a calibration stored in a file.
SaveCalibrationToFile	Save the calibration in use to file.
ClearCalibration	Clear the calibration under construction.
AddCalibrationPoint	Start the sampling of gaze data to the calibration under construction.
InterruptAddCalibrationPoint	Interrupt the adding of a calibration point.
CalculateAndSetCalibration	Set a new calibration in use based on the calibration under construction.
GetCalibrationResult	Get information about a calibration. The source may be either the one in use or one stored in a file.
RemoveCalibrationPoints	Remove points from the calibration under construction.
PerformSystemCheck	Check if there are any system errors present.
GetSerialNumber	Get the hardware serial number.

Table 5.3. *ITetClient* Methods Quick Reference

Property	Access	VC++/VB 6 Type	Description
IsConnected	Read	VARIANT_BOOL/Boolean	Whether connected to a TETServer or not.
IsTracking	Read	VARIANT_BOOL/Boolean	Determines if the TetClient is receiving gaze data from the TETServer.
ServerAddress	Read	BSTR/String	If connected, gives the TETServer IP host address.
PortNumber	Read	LONG/Long	If connected, gives the TETServer IP port number.
SynchronizationMode	Read	TetSynchronizationMode	Determines which time base the received gaze data has and which time base the method GetTimeStamp returns. See Connect.
IsAddingCalibrationPoint	Read	VARIANT_BOOL/Boolean	Determines whether the TetClient is currently in the process of adding a calibration point.
GazeDataDelivery	Read/Write	TetGazeDataDelivery	Determines how the TetClient handles firing of gaze data events.

Table 5.4. *ITetClient* Properties Quick Reference

Event	Description
OnTrackingStarted	Tracking was successfully started.
OnTrackingStopped	Tracking was stopped programmatically or by an error.
OnGazeData	There is new gaze data available during tracking.
OnPostGazeData	There is new gaze data available during tracking.
OnAddCalibrationPointStarted	Calibration was successfully started.
OnAddCalibrationPointEnded	Calibration was stopped programmatically or by an error.
OnCalibrationGazeData	There is new gaze data available during calibration.
OnPostCalibrationGazeData	There is new gaze data available during calibration.

Table 5.5. *ITetClient* Events Quick Reference

5.4.4 Method `TetClient.Connect`

VC++ Prototype:

```
HRESULT Connect(
    BSTR bstrServerAddress,
    LONG portNumber,
    TetSynchronizationMode syncMode);
```

VB 6 Prototype:

```
Sub Connect(
    bstrServerAddress As String,
    portNumber As Long,
    syncMode As TetSynchronizationMode)
```

Description:

This function will create a connection to the *Tobii Eye Tracker Server* (TETServer) and must be called successfully prior to any other call that involves the eye tracker. Call *Disconnect* when done.

Several threads and processes may be connected, at the same time, to a TETServer. Bear in mind though that the tasks being performed should be coordinated. For example, two threads or processes are allowed to set or clear a calibration at the same time, which will possible cause lost calibration data.

If the TETServer is running on another host, background time synchronization may be turned on implicitly. This makes the time stamp in gaze data and time stamps given by *GetTimeStamp* compatible. The parameter *syncMode* is controlling the mode of this synchronization. A detailed description of when and of how to use this is found in the appendix section of this document. The appendix is written from a *Tobii Eye Tracker Low Level API* perspective.

Arguments:

`bstrServerAddress` [IN]

The IP address or name of the host where the TETServer runs.

Ex: "192.168.1.73" or "MyHost".

Using *NULL* (VB 6: *vbNullString*) is the same as using the default local host; "127.0.0.1" or "localhost".

`portNumber` [IN]

TETServer TCP/IP listening port number. The default port number is *TetConstants.TetConstants_DefaultServerPort* = 4455.

`syncMode` [IN]

Determines which time base the received gaze data has and which time base the method *GetTimeStamp* returns. The time base may be either server host time base or local host time base. If the connecting application runs on the same host as TETServer, no synchronization is needed.

Member name	Value	Description
<code>TetSynchronizationMode_None</code>	1	No background synchronization.
<code>TetSynchronizationMode_Server</code>	2	Synchronization on. Compatible time stamps in gaze data time stamps, in <i>GetTimeStamp</i> time stamps within the same thread as <i>Connect</i> was called in and any process calling <i>GetTimeStamp</i> on the server host.
<code>TetSynchronizationMode_Local</code>	3	Synchronization on. Compatible time stamps in gaze data time stamps, in <i>GetTimeStamp</i> time stamps within the same thread as <i>Connect</i> was called in and any process calling <i>GetTimeStamp</i> on the client host.

Table 5.6. The *TetComp.TetSynchronizationMode* enumeration.

Example 2: Tobii Eye Tracker Low level API – Function `Tet_Connect`

5.4.5 Function `Tet_Connect`

```
long __stdcall Tet_Connect(
char *pServerAddress,
long portnumber,
ETet_SynchronizationMode syncmode);
```

Description:

This function will create a connection to the Tobii Eye Tracker Server (TETServer) and setup necessary internal states. No other functionality but logging (see `Tet_UseDebugLog`) is available prior to this call. A call to `Tet_Disconnect` is mandatory when done.

All `Tet_` functions can be called from any thread. However, each thread will have its own connection to the server and does not share anything with other threads within the same process. This means that `Tet_Connect` (and `Tet_Disconnect`) must be called in each thread that use `Tet_` functions. Bear in mind that if the threads or processes are connected to the same TETServer, the tasks should be coordinated. For example, two threads or processes are allowed to set or clear a calibration at the same time, which will possible cause lost calibration data.

If the TETServer is running on another host, background time synchronization may be turned on implicitly. The parameter `syncmode` is controlling the mode of this synchronization. A detailed description of when and of how to use this is found in the appendix section of this document.

Arguments:

`pServerAddress` [IN]

Pointer to a null terminated array of chars containing the IP address or name of the host where the Tobii Eye Tracker Server (TETServer) runs.

Ex: "192.168.1.73" or "MyHost".

Using NULL is the same as using the default local host; "127.0.0.1" or "localhost".

`portnumber` [IN]

TETServer TCP/IP listening port number. Using 0 is the same as using the default port number 4455.

`syncmode` [IN]

Controlling how the time stamps will be synchronized if calling process is running on another host that the TETServer.

`TET_SYNC_NONE` – No background synchronization.

`TET_SYNC_LOCAL` – Synchronization on. Gaze data time stamp will be compatible with time stamps given by the `TT_GetLocalTimeStamp`.

`TET_SYNC_SERVER` – Synchronization on. Gaze data time stamp will be compatible time stamps given by `TT_GetServerTimeStamp`

Return Code:

`TET_NO_ERROR` on success, else `TET_ERROR`.

Call `Tet_GetLastError` or `Tet_GetLastErrorAsText` for error details.

Syntax Snippet:

```
res = Tet_Connect("192.168.1.73", 4455, TET_SYNC_LOCAL);
if ( res != TET_NO_ERROR ) {
printf("Tet_Connect failed, reason %d - %s\n",
Tet_GetLastError(),
Tet_GetLastErrorAsText(pErrBuf));
}
```