Mirametrix Research

Application Programming Interface

Version 1.0

Revised January 13, 2010

Mirametrix Research Inc.

309-2233 3rd Ave West • Vancouver, B.C. • V6K 1L5 • (604) 346-0335 • www.mirametrix.com

Table of Contents

1 Introduction

The Mirametrix eye-trackers conform to the Open Eye-gaze Interface, providing developers with the ability to add eye-gaze to custom applications. The Open Eye-gaze Interface provides a standardized method for performing calibration and acquiring eye-gaze information.

2 Operation

The Open Eye-gaze Interface is based on a web services model for data transmission. Communication is performed using a client / server architecture, passing XML strings over a TCP/IP connection.

2.1 Client / Server

The eye-gaze tracker operates as the server, responding to queries from a connected client. The server responds with an acknowledge once for each client command with the exception of calibration and free-running data transmission. When calibrating the server will transmit the results of the calibration procedure at each point as it takes place, while a continuous stream of eye-gaze information is transmit when free-running data transmission is enabled.

2.2 TCP/IP

Communication takes place over a TCP/IP connection. Using TCP/IP allows the eye-tracker to easily operate on a second computer with commands and data transmission taking place over a network connection.

2.3 XML

All data structures take the form of XML string fragments. Using strings prevents the need for proprietary data structures, while the XML format provides an open and easily extensible method for managing and parsing the formatted data.

Client commands are either GET or SET operations where the XML tag is GET or SET.

2.4 Coordinates

All reported coordinates are in percentages of the tracking window (unless otherwise specified), which itself is in percentages of the screen size. By default the tracking window is 100% of the screen-size and therefore only the screen size is required to convert from the reported X,Y coordinates to pixels. For X coordinates, 0 is the right most position and 1 is the left most position and for the Y coordinate 0 is the top most position and 1 is the bottom most position.

For the pupil image coordinates the X,Y values are percentages of the camera image size.

3 Commands

The eye-tracker is controlled using GET and SET commands in the form of XML fragments. A GET or SET command is always replied to with a successful (ACK) or unsuccessful response (NACK). The command may have any number of additional parameters and takes the form:

```
SEND: <CMD ID="DATA_ID" PARAM1="VALUE" ... />
REPLY: <ACK ID="DATA_ID" REPLYPARAM1="VALUE" ... />
```

3.1 SET

Description

The SET command is used to set variables within the eye-tracker and to initiate or cancel operations.

Example Usage

```
SEND: <SET ID="ENABLE_SEND_COUNTER" STATE="0" />
REPLY: <ACK ID="ENABLE_SEND_COUNTER" STATE="0" />
```

3.2 GET

Description

The GET command is used to get variables from the eye-tracker and to check on the state of an operation.

Example Usage

```
SEND: <GET ID="ENABLE_SEND_COUNTER" />
REPLY: <ACK ID="ENABLE_SEND_COUNTER" STATE="0" />
```

4 Calibration

4.1 CALIBRATE_START

Description

Start or stop the calibration procedure.

DATA_ID	Read/Write	Parameters	Type	Description
CALIBRATE_START	R/W	STATE	Boolean	Start or stop calibration procedure

Example Usage

```
SEND: <GET ID="CALIBRATE_START" />
REPLY: <ACK ID="CALIBRATE_START" STATE="0" />
SEND: <SET ID="CALIBRATE_START" STATE="1" />
REPLY: <ACK ID="CALIBRATE_START" STATE="1" />
```

4.2 CALIBRATE_SHOW

Description

Show or hide the calibration window.

DATA_ID	Read/Write	Parameters	Type	Description
CALIBRATE_SHOW	R/W	STATE	Boolean	Show or hide the default calibration window

Example Usage

```
SEND: <GET ID="CALIBRATE_SHOW" />
REPLY: <ACK ID="CALIBRATE_SHOW" STATE="0" />
SEND: <SET ID="CALIBRATE_SHOW" STATE="1" />
REPLY: <ACK ID="CALIBRATE_SHOW" STATE="1" />
```

4.3 Calibration Point Completed

While calibrating, after the completion of each calibration point a response is sent indicating the point that just finished.

Data ID	Param	Type	Description
CALIB_RESULT_PT	PT	int	Calibration point completed

Example Result

```
<CAL ID="CALIB_RESULT_PT" PT="1" />
```

4.4 Calibration Procedure Completed

When the calibration procedure completes, the results are summarized in an XML string and transmit. An example of the calibration return data is shown below for a single point.

Data ID	Param	Type	Description
CALIB_RESULT	CALX?	float	Calibration X coordinate for point ?
CALIB_RESULT	CALY?	float	Calibration X coordinate for point ?
CALIB_RESULT	LX?	float	Left eye point-of-gaze X for point ?
CALIB_RESULT	LY?	float	Left eye point-of-gaze Y for point ?
CALIB_RESULT	LV?	int	Left eye valid flag for point ?
CALIB_RESULT	RX?	float	Right eye point-of-gaze X for point ?
CALIB_RESULT	RY?	float	Right eye point-of-gaze Y for point ?
CALIB_RESULT	RV?	int	Right eye valid flag for point ?

Example Result

```
<CAL ID="CALIB_RESULT" CALX1="0.10000" CALY1="0.08000" LX1="0.09200"
LY1="0.07420" LV1="1" RX1="0.12679" RY1="0.13619" RV1="1" ... />
```

5 Remote Data

The GET and SET commands operate on the remote data specified by the DATA_ID identifier, along with optional parameters.

5.1 ENABLE_SEND_DATA

Description

Enable or disable the sending of the eye-gaze data records. Enabling this variable begins the continuous sending of the eye-gaze data records.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_DATA	R/W	STATE	Boolean	Start or stop data streaming

Example Usage

```
SEND: <GET ID="ENABLE_SEND_DATA" />
REPLY: <ACK ID="ENABLE_SEND_DATA" STATE="0" />
SEND: <SET ID="ENABLE_SEND_DATA" STATE="1" />
REPLY: <ACK ID="ENABLE_SEND_DATA" STATE="1" />
```

Data Record

If enabled the data record will be continuously transmit. If no other parameters are enabled the result will look like:

```
<REC />
<REC />
...
```

5.2 ENABLE_SEND_COUNTER

Description

Enable or disable the sending of the packet counter. The packet counter is incremented each time a packet is sent from the server. The packet counter provides a means for determining if packets are being lost.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_COUNTER	R/W	STATE	Boolean	Enable counter data

Data Record

The counter is identified in the data record as follows:

Param	Type	Description
CNT	integer	Sequence counter for data packets

Example Data Record

```
<REC CNT="1"/>
<REC CNT="2"/>
...
```

5.3 ENABLE_SEND_TIME

Description

Enable or disable the elapsed time variable. The elapsed time variable identifies the elapsed time since the last calibration.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_TIME	R/W	STATE	Boolean	Enable time stamp data

Data Record

The elapsed time is identified in the data record as follows:

Param	Type	Description
TIME	float	Elapsed time in seconds since last system initialization or calibration

Example Data Record

```
<REC TIME="1141.437"/>
<REC TIME="1141.453"/>
...
```

5.4 ENABLE_SEND_TIME_TICK

Description

Enable or disable the high resolution timer tick. The high resolution timer tick returns the output of the QueryPerformanceCounter (Win32 API) function and can be used to synchronize eye-gaze with other data recorded on the same computer. To convert to seconds divide this value by the output of the TIME_TICK_FREQUENCY variable.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_TIME_TICK	R/W	STATE	Boolean	Enable high resolution timer tick

Data Record

The elapsed time is identified in the data record as follows:

Param	Type	Description
TIME_TICK	LONGLONG	Tick count (signed 64-bit integer)

Example Data Record

```
<REC TIME_TICK="5712427212840"/>
<REC TIME_TICK="5712464779821"/>
...
```

5.5 TIME_TICK_FREQUENCY

Description

Get the timer tick frequency for high resolution timing. This returns the output of QueryPerformanceFrequency in the Win32 API.

DATA_ID	Read/Write	Parameters	Type	Description
TIME_TICK_FREQUENCY	R	FREQ	LONGLONG	Tick frequency (signed 64-bit integer)

Example Usage

```
SEND: <GET ID="TIME_TICK_FREQUENCY" />
REPLY: <ACK ID="TIME_TICK_FREQUENCY" FREQ="2405480000" />
```

5.6 ENABLE_SEND_POG_LEFT

Description

Enable or disable the sending of the point-of-gaze data from the left eye.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_POG_LEFT	R/W	STATE	Boolean	Enable left eye point-of-gaze data

Data Record

The left eye point of gaze is identified in the data record as follows:

Param	Type	Description
LPOGX	float	Left point-of-gaze X
LPOGY	float	Left point-of-gaze Y
LPOGV	int	Left point-of-gaze valid flag

Example Data Record

```
<REC LPOGX="0.21726" LPOGY="0.35524" LPOGV="1"/>
<REC LPOGX="0.15774" LPOGY="0.37048" LPOGV="1"/>
...
```

5.7 ENABLE_SEND_POG_RIGHT

Description

Enable or disable the sending of the point-of-gaze data from the right eye.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_POG_RIGHT	R/W	STATE	Boolean	Enable right eye point-of-gaze data

Data Record

The right eye point of gaze is identified in the data record as follows:

Param	Type	Description
RPOGX	float	Right point-of-gaze X
RPOGY	float	Right point-of-gaze Y
RPOGV	int	Right point-of-gaze valid flag

Example Data Record

```
<REC RPOGX="0.11667" RPOGY="0.39333" RPOGV="1"/>
<REC RPOGX="0.11131" RPOGY="0.48857" RPOGV="1"/>
...
```

5.8 ENABLE_SEND_POG_FIX

Description

Enable or disable the sending of the fixation point-of-gaze data.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_POG_FIX	R/W	STATE	Boolean	Enable fixation point-of-gaze data

Data Record

The fixation point of gaze is identified in the data record as follows:

Param	Type	Description
FPOGX	float	Fixation point-of-gaze X
FPOGY	float	Fixation point-of-gaze Y
FPOGS	float	Fixation start (seconds)
FPOGD	float	Fixation duration (elapsed time since fixation start (seconds))
FPOGID	int	Fixation number ID
FPOGV	int	Fixation point-of-gaze valid flag

Example Data Record

```
<REC FPOGX="0.40595" FPOGY="0.78762" FPOGS="0.512" FPOGD="0.320" FPOGID="2" FPOGV="1"/>
<REC FPOGX="0.40274" FPOGY="0.80048" FPOGS="0.512" FPOGD="0.336" FPOGID="2" FPOGV="1"/>
...
```

5.9 ENABLE_SEND_PUPIL_LEFT

Description

Enable or disable the sending of the left pupil image data.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_PUPIL_LEFT	R/W	STATE	Boolean	Enable left eye image data

Data Record

The pupil image data is identified in the data record as follows:

Param	Type	Description
LPCX	float	Left eye pupil center X
LPCY	float	Left eye pupil center Y
LPD	float	Left eye pupil diameter
LPS	float	Left eye pupil distance (unit less, from calibration position)
LPV	int	Left eye pupil image valid

Example Data Record

```
<REC LPCX="0.09973" LPCY="0.20000" LPD="16.30" LPS="0.77" LPV="1"/>
<REC LPCX="0.09973" LPCY="0.19375" LPD="16.20" LPS="0.77" LPV="1"/>
...
```

5.10 ENABLE_SEND_PUPIL_RIGHT

Description

Enable or disable the sending of the right pupil image data.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_PUPIL_RIGHT	R/W	STATE	Boolean	Enable right eye image data

Data Record

The pupil image data is identified in the data record as follows:

Param	Type	Description
RPCX	float	Right eye pupil center X
RPCY	float	Right eye pupil center Y
RPD	float	Right eye pupil diameter
RPS	float	Right eye pupil distance (unit less, from calibration position)
RPV	int	Right eye pupil image valid

Example Data Record

```
<REC RPCX="0.47473" RPCY="0.17917" RPD="14.90" RPS="1.55" RPV="1"/>
<REC RPCX="0.47074" RPCY="0.17917" RPD="14.82" RPS="1.55" RPV="1"/>
...
```

5.11 ENABLE_SEND_CURSOR

Description

Enable or disable the sending of the right mouse cursor position.

DATA_ID	Read/Write	Parameters	Type	Description
ENABLE_SEND_CURSOR	R/W	STATE	Boolean	Enable mouse cursor data

Data Record

The mouse cursor data is identified in the data record as follows:

Param	Type	Description
CX	float	Cursor X position
CY	float	Cursor Y position
CS	int	Cursor button state 0 - No press 1 - Left button down 2 - Left button up 3 - Left double click 4 - Right button down 5 - Right button up 6 - Right double click

Example Data Record

```
<REC CX="0.12321" CY="0.32571" CS="0"/>
<REC CX="0.12619" CY="0.36286" CS="0"/>
...
```

5.12 SCREEN_SIZE

Description

Get the size of the screen in pixels.

DATA_ID	Read/Write	Parameters	Type	Description
SCREEN_SIZE	R	WIDTH	int	Screen width (pixels)
	R	HEIGHT	int	Screen height (pixels)

Example Usage

```
SEND: <GET ID="SCREEN_SIZE" />
REPLY: <ACK ID="SCREEN_SIZE" WIDTH="1680" HEIGHT="1050" />
```

5.13 CAMERA_SIZE

Description

Get the size of the camera image in pixels.

DATA_ID	Read/Write	Parameters	Type	Description
CAMERA_SIZE	R	WIDTH	int	Camera image width (pixels)
	R	HEIGHT	int	Camera image height (pixels)

Example Usage

```
SEND: <GET ID="CAMERA_SIZE" />
REPLY: <ACK ID="CAMERA_SIZE" WIDTH="752" HEIGHT="480" />
```

5.14 TRACK_RECT

Description

Get the size of the position and size of the tracking rectangle.

DATA_ID	Read/Write	Parameters	Type	Description
TRACK_RECT	R	X	float	X coordinate of tracking rectangle
	R	Y	float	Y coordinate of tracking rectangle
	R	WIDTH	float	Width of tracking rectangle
	R	HEIGHT	float	Height of tracking rectangle

Example Usage

```
SEND: <GET ID="TRACK_RECT" />
REPLY: <ACK ID="TRACK_RECT" X="0.0000" Y="0.0000" WIDTH="1.0000" HEIGHT="1.0000" />
```

5.15 PRODUCT_ID

Description

Product identifier.

DATA_ID	Read/Write	Parameters	Type	Description
PRODUCT_ID	R	VALUE	string	Product identifier

Example Usage

```
SEND: <GET ID="PRODUCT_ID" />
REPLY: <ACK ID="PRODUCT_ID" VALUE="S1" />
```

5.16 SERIAL_ID

Description

Product serial number.

DATA_ID	Read/Write	Parameters	Type	Description
SERIAL_ID	R	VALUE	string	Device serial number

Example Usage

```
SEND: <GET ID="SERIAL_ID" />
REPLY: <ACK ID="SERIAL_ID" VALUE="570631454" />
```

5.17 COMPANY_ID

Description

Manufacturer identifier.

DATA_ID	Read/Write	Parameters	Type	Description
MFG_ID	R	VALUE	string	Manufacturer identity

Example Usage

```
SEND: <GET ID="COMPANY_ID" />
REPLY: <ACK ID="COMPANY_ID" VALUE="Mirametrix Research Inc." />
```

5.18 API_ID

Description

API identity.

DATA_ID	Read/Write	Parameters	Type	Description
API_ID	R	MFG_ID	string	API vendor identity
		VER_ID	string	API version number

Example Usage

```
SEND: <GET ID="API_ID" />
REPLY: <ACK ID="API_ID" MFG_ID="Mirametrix" VER_ID="1.0" />
```

5.19 API_SELECT

Description

Select the API to use for further communication.

DATA_ID	Read/Write	Parameters	Type	Description
API_SELECT	R	MFG_ID?	string	List of vendors supported
	R	VER_ID?	string	List of versions supported
	W	STATE	int	API selected

Example Usage

```
SEND: <GET ID="API_SELECT" />
REPLY: <ACK ID="API_SELECT" MFG_ID0="generic" VER_ID0="1.0"
MFG_ID1="Mirametrix" VER_ID1="1.0" />
SEND: <SET ID="API_SELECT" STATE="1" />
REPLY: <ACK ID="API_SELECT" STATE="1" />
```