E-Prime Extensions for tobii 2.0

User Manual

For Research and Education Only

Psychology Software Tools, Inc.
311 23rd Street Extension, Suite 200
Sharpsburg, PA 15215-2821
Phone: 412.449.0078
Fax: 412.449.0079
E-mail: info@pstnet.com

PST-100956

E-Prime Extensions for Tobii 2.0 User Manual
PST-100796
Rev 24

Psychology Software Tools, Inc.
311 23rd Street Extension, Suite 200
Sharpsburg, PA 15215-2821
Phone: 412-449-0078
Fax: 412-449-0079
E-mail: info@pstnet.com
Web: www.pstnet.com

For questions or comments regarding this manual or installation assistance:
Please e-mail us at support@pstnet.com or visit us at https://support.pstnet.com.

This manual describes the installation procedure for the E-Prime Extensions for Tobii 2.0. Please review the manual completely and thoroughly before beginning the system installation.

The E-Prime Extensions for Tobii 2.0 Software (PST-100956) is for research and educational purposes only.

Table of Contents

Chapter 1: **Getting Started Guide**

1.1 Installation Instructions

E-Prime Extensions for Tobii 2.0 is compatible with E-Prime 2.0 Professional Service Pack 1 (SP1) only. E-Prime 1.x is not supported with the current software release or the calibration routines included with the current software release. The E-Prime 2.0 Professional SP1 file extension is .es2. This Getting Started Guide uses the .es2 extension.

For a summary of E-Prime's new features, review the *E-Prime 2.0 New Features-Reference Guide* that can be accessed through the Start menu: Start > All Programs > E-Prime 2.0 > Documentation, or through the Help menu in E-Studio: Help > Documentation > New Features-Reference Guide.

Prior to the E-Prime Extensions for Tobii 2.0 installation, you will need to determine which version of E-Prime you currently have on your machine.

1.2 E-Prime Extensions for Tobii 2.0 Overview

E-Prime Extensions for Tobii 2.0 is a set of software routines that allow communication between the TET Server (Tobii Eye Tracker Server) and E-Prime during the run of an experiment. It will allow you to create E-Prime experiments or update existing E-Prime experiments to function with the Tobii Eye Tracker. The extensions also include a set of sample experiments that can be run directly or used as a basis from which to create new experiments.

1.3 Software Installation

If you have a compatible version of E-Prime 2.0 already installed on your system you can skip to the EET installation instructions. Before continuing, be sure that you have administrative rights to install this software on the computer. If you do not have administrative rights, you will be unable to install E-Prime 2.0. If you are unsure of your administrative privileges, contact your System Administrator.

Install E-Prime 2.0

1) ***Uninstall*** any **previous version** of **E-Prime**.
   The **E-Prime Extensions for Tobii 2.0** ***requires*** a copy of **E-Prime 2.0 Professional SP1** to be installed on the machine.
2) ***Insert*** the **E-Prime 2.0 CD** into your **CD-ROM** drive.
3) The **installation** should ***automatically launch***.
   ⚠ **NOTE:** *If it does not, you may use Windows Explorer to browse the CD and launch the Setup.exe file in the main folder.*
4) ***Follow*** the **prompts** in the **installation program** to ***provide*** any **required information** (e.g. User Name, Institution, Serial Number, etc).
   ⚠ **NOTE:** *Installation of E-Prime requires a valid E-Prime License (verified through the E-Prime HASP USB hardware key and Serial Number).*

Installing E-Prime Extensions for Tobii 2.0

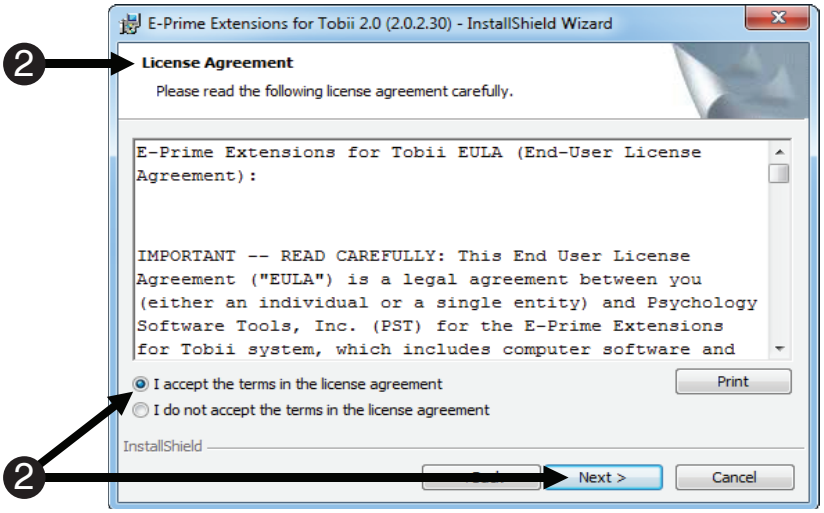
Before continuing, be sure that you have administrative rights to install this software on the computer. If you do not have administrative rights, you will be unable to install E-Prime Extensions for Tobii 2.0. If you are unsure of your administrative privileges, contact your System Administrator.

⚠ **NOTE:** *E-Prime Extensions for Tobii 2.0 is compatible with E-Prime 2.0 Professional only.*

⚠ **NOTE:** *The version number on the following images may not correspond to the version number on your software.*

1) ***Insert*** the **Tobii installation CD** into your **CD-ROM drive**. ***Click*** the **Next** button to continue installation.

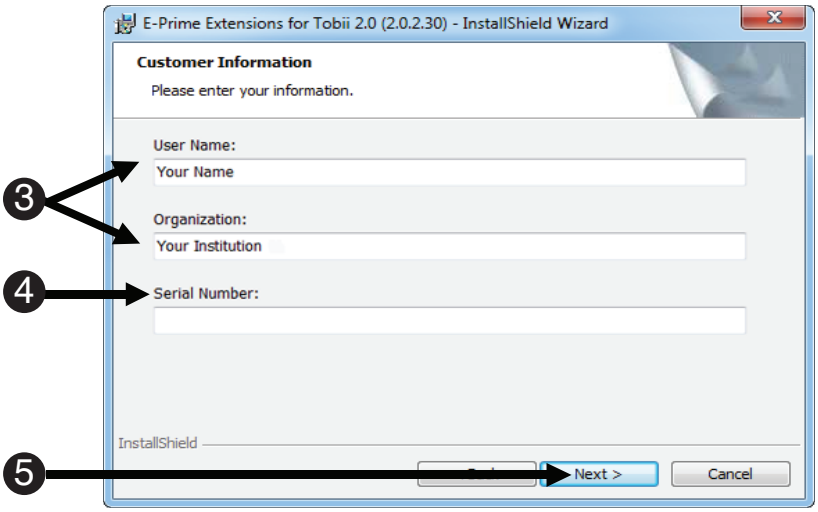
2) ***Please read*** the **License Agreement** and make sure that you ***agree completely*** with the terms and conditions described in the **agreement** before proceeding. Once you have read the agreement, ***click*** **Next** to proceed with the installation.



3) ***Specify*** **Your Name** and **Your Institution** or check with your system administrator for appropriate information.

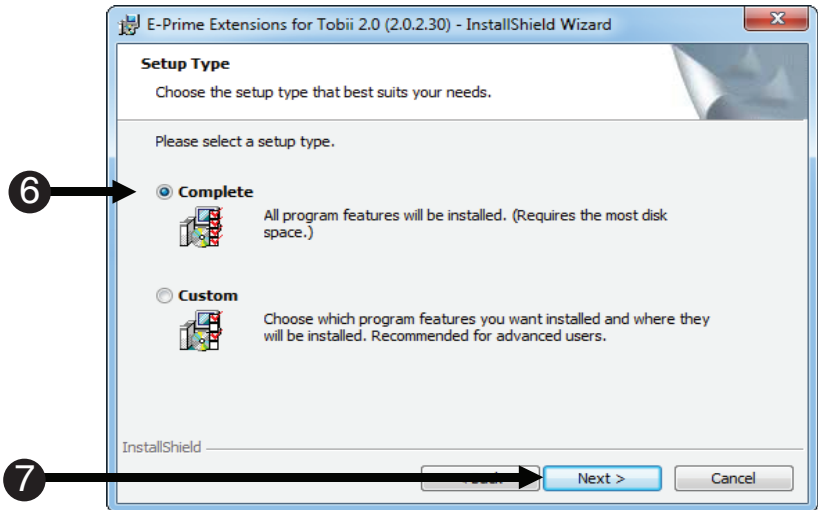
4) The **Serial Number field** must be ***complete*** to obtain access to online **Product Service and Support**.

5) ***Click*** **Next** to begin transferring files to your computer.

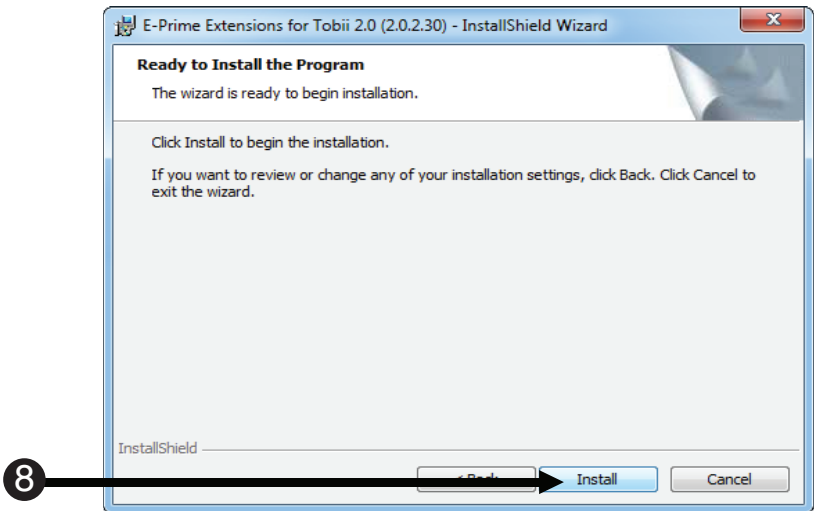


6) ***Select*** **Complete**.

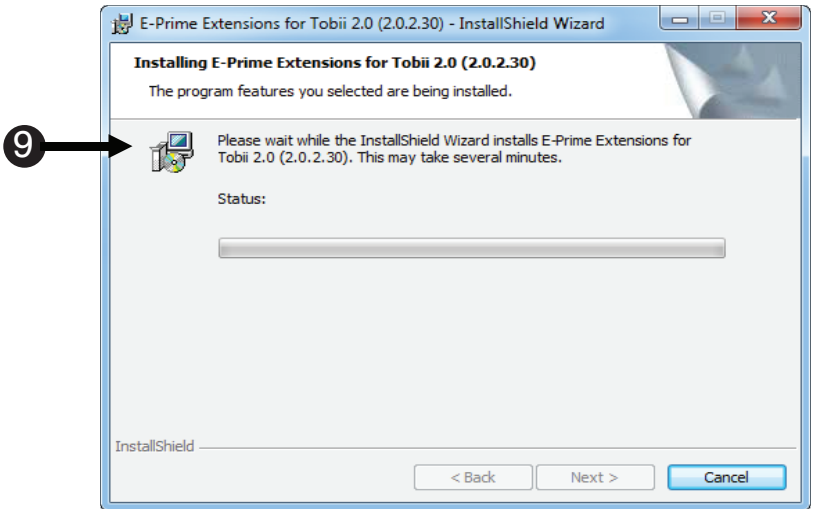
7) ***Click*** **Next**.



8) ***Click*** **Install** to continue the installation.

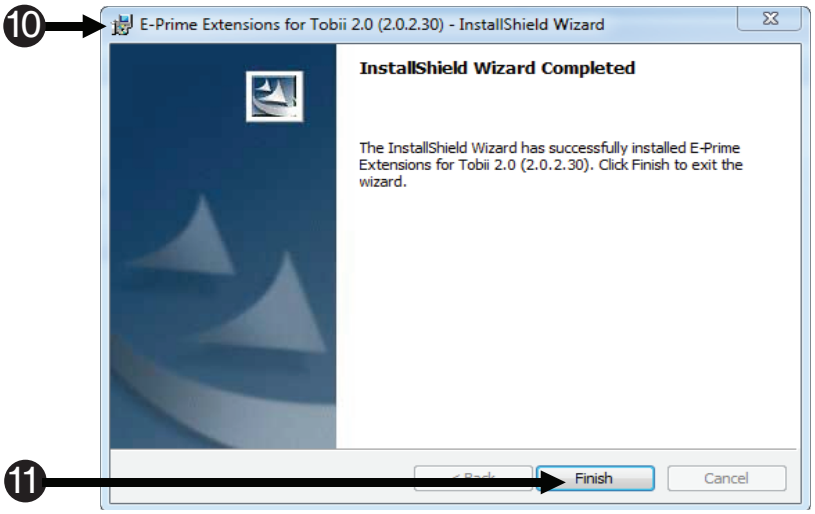


9) ***Wait*** while the installer configures the software.



10) If **E-Prime Extensions for Tobii** is installed properly, you will see the following window.

11) ***Click*** **Finish** to complete installation.



1.3 Software Installation (continued)

Finding the "My Experiments" Folder

E-Prime 2.0 is compatible with Microsoft Windows XP, Vista, 7 and 8. You will frequently be working within the "My Experiments" folder, because this folder is the default location to store new experiments created with E-Studio. E-Prime creates the "My Experiments" folder in your personal documents folder on your PC. This folder also contains the "Samples" folder, which stores the sample experiments that are documented in **Appendix B** of the *E-Prime User's Guide*, and the "Tutorials" folder, which stores the E-Studio files that are documented in the *E-Prime Getting Started Guide*.

The table below shows the default paths to your personal documents folder. Note that the path on your particular machine may have been modified by your administrator:

Operating System	Path to your personal documents folder ("My Documents" or "Documents")
XP	<drive>\Documents and Settings\<user name>\My Documents\
Vista	<drive>\Users\<user name>\Documents\
*Windows 7 and 8	<drive>\Users\<user name>\My Documents

**Windows 7 and 8 actually go to "Documents" but it is visible as "My Documents".*

When the E-Prime documentation directs you to the "My Experiments" folder, it does not include the full path to the folder. Instead, the documentation refers to "…My Experiments", where the "…" indicates the full path up to your personal documents folder. When you see this notation in the documentation, replace the "…" with the path to your personal documents folder.

Running a Test Experiment

1) Test Tobii Studio and TET installations. The following test of the EET installation assumes the proper functioning of both Tobii Studio and Tobii eye tracking monitor. If you are new to Tobii, confirm proper functioning by running and analyzing data from at least one Tobii sample paradigm.
2) Launch E-Studio.
3) When E-Studio launches, it will detect the new Tobii related sample files. You should allow E-Prime to copy the new samples into your My Experiments folder.
4) After E-Prime has copied the samples, navigate to My Experiments\Tobii\Samples\TET\TETVaryingPositionAOITracking.
5) Locate and open the TETVaryingPositionAOITracking.es2 file.
6) In the E-Studio Structure window, double-click the Experiment Object located at the top of the tree.
7) Click on the Devices tab of the Experiment Object Property Pages.
8) Double-click on the TobiiEyeTracker device to edit its Properties.
9) Specify the monitor serial number located on the back of your Tobii eye tracker for the Tobii eye tracking monitor, e.g., "TT120-204-87700423.local.".
10) Click the OK button to dismiss the Properties dialog and save the changes.
11) Click the OK button to dismiss the Experiment Object Property Pages.
12) Click the Run button in E-Studio or press F7 to run the TETVaryingPositionAOITracking sample.
13) TETVaryingPositionAOITracking:
    - The track status window will be displayed for you to find your eyes. Once you can see both eyes in the box, press any key to move on to calibration. Follow the dots to complete calibration and when you have a suitable calibration accept the calibration to proceed with the experiment.
    - Follow the instructions on screen to perform the task.
    - When the fixation + is shown, you must focus on the fixation continuously for 2 seconds to begin each trial. Alternately, you may press any key to continue.

1.4 Product Service and Support

Psychology Software Tools, Inc. provides technical support for E-Prime via the PST web site. In order to receive technical support, you must register online at

https://support.pstnet.com

Registration requires a valid E-Prime serial number, EET serial number, and e-mail address. At the support site, you will also find a Knowledge Base including release notes and a compilation of frequently asked questions. The support site also includes E-Prime sample paradigms that are available for you to download. There is an additional support forum you can use to post general questions about E-Prime 1.x, E-Prime 2.0 and E-Prime Extensions for Tobii 2.0. For additional details: https://support.pstnet.com.

1.5 Resources

1) Tobii support can be contacted at support@tobii.com.
2) Tobii manuals are included in the Tobii product shipment.
3) Check PST's web site for additional resources: http://www.pstnet.com/eet.
4) Eye Tracking Related Resources: http://www.pstnet.com/resources.cfm?ID=100.
5) EET users are entitled to one year of Silver Support E-Prime technical support via PST's Product Service and Support web site.
   For additional details: http://www.pstnet.com/eprime.cfm?tabID=Support

Chapter 2: **How to Create Your Own Tobii Experiment**

2.1 Overview

This chapter will illustrate the creation of an E-Prime/Tobii experiment using one of the sample paradigms provided with the E-Prime Extensions for Tobii 2.0 installation (EET), the TETFixedPositionAOI.es2 paradigm. It will also introduce and explain some programming methods that E-Prime uses.

In order to complete the tutorials, you will need a computer with E-Prime and the EET software already installed. If you have not installed E-Prime, please complete the installation before continuing. If you have not installed EET, please refer to the installation section in **Chapter 1: Getting Started Guide, (Page 6)** of this manual.

While this document includes some basic introduction to programming concepts specific to using E-Prime with Tobii, the tutorials assume you are familiar with using E-Prime to build behavioral experiments. If you are new to using E-Prime, it is suggested that you work through all of the tutorials included in the *E-Prime 2.0 Getting Started Guide* and *E-Prime 2.0 User's Guide* prior to beginning this tutorial.

2.2 Programming Methods

Programming Methods Overview

The E-Prime Extensions for Tobii 2.0 (EET) combines two technologies to create an integrated programming environment: PackageCalls and InLine Script. The typical EET experiment will include use of both technologies. A brief overview of each is provided below.

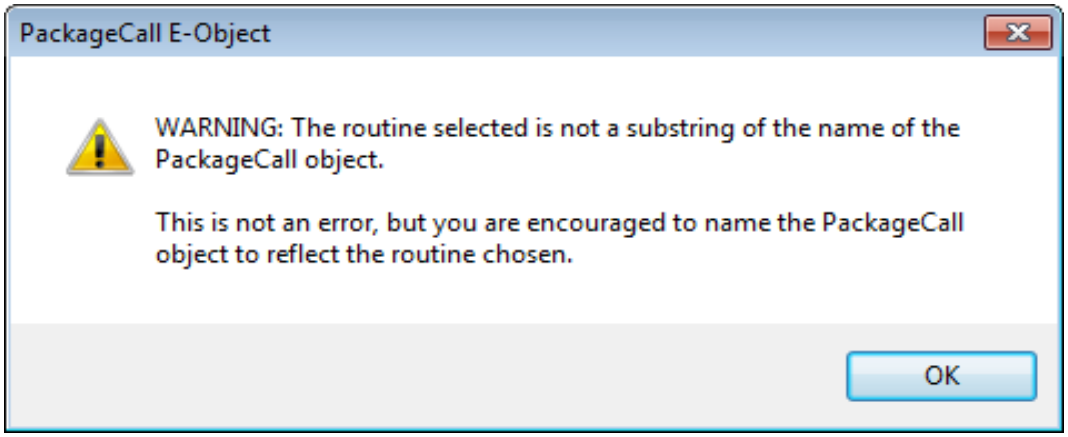
Use of PackageCalls in the EET

PackageCalls are pre-written, cohesive sets of E-Basic routines grouped together in a single file that can be maintained externally. These libraries are included in the EET installation. The EET PackageCalls will allow you to automatically perform various functions e.g., hardware initialization and data output formatting via graphical representations in the Structure window in E-Studio. Some of the EET PackageCalls will require you to set or modify parameters to enable them to work with your experiment. The **Chapter 3: Tobii PackageCall Reference, (Page 147)** found in this manual provides a listing of available routines along with descriptions of how to use them.

PackageCall Method

Routines contained within a package file are typically called by dragging a PackageCall from the E-Studio Toolbox and dropping it into a Procedure Object at the location within the experiment where the call is to be invoked.

When using the PackageCall, it is strongly recommended that you rename the object to reflect the specific Package File and Routines which are being referenced within the object. If you do not follow this recommendation, E-Prime will issue the following warning:



It is also a common convention, when calling routines within the Tobii Package File, to create a name for each object by abbreviating “Tobii” as “TET” and then concatenating the name of the routine being called. For example, a PackageCall that is configured to call the “Open” routine would be named “TETOpen”.

2.2 Programming Methods

InLine Script Method

An InLine is an object used to insert user-defined script into an experiment at a specific point. The object is placed at the desired location in the structure. The InLine method can be used to call functions or routines in the Package File as well. However, unless there is a need to make a series of PackageCalls in sequence or mingle PackageCalls with additional E-Basic script, this calling method is not typically recommended.

If you are making calls to functions or routines via the E-Basic script, please refer to **Chapter 3: Tobii PackageCall Reference, (Page 147)** and **Tutorial 2: Writing to the Tab Delimited .gazedata File, (Page 45)**. These sections of the manual provide PackageCall parameters and functions as examples of the script commonly required to successfully invoke each routine.

⚠ **NOTE:** *Users must have the EET fully and correctly installed prior to using the Tobii PackageCalls by either method.*

Tutorial 1: **Adding Tobii Support to an E-Prime Experiment**

Summary:

Incorporating Tobii support into an existing E-Prime experiment primarily involves adding both the TET PackageFile and the TET PackageCalls to the experimental structure at the appropriate locations. PackageCalls are added to the experiment, then renamed. The required parameters are then edited to allow the user to adapt the TET Package to meet their individual needs.

During this tutorial, you will add E-Prime Extensions for Tobii 2.0 support to the FixedPositionAOI.es2 sample experiment installed by default with Tobii. The FixedPositionAOI.es2 sample experiment consists of one block. At the beginning of the experiment you are prompted to calibrate the Tobii Eye Tracker, just like in Tobii Studio. First you will see a black box. This is the TrackStatus window. The TrackStatus window shows you when the Tobii Eye Tracker can see your eyes by displaying two white circles in the box, and by showing a green box along the bottom of the box that has the word both written in it. If the Tobii Eye Tracker is only detecting one eye, then the TrackStatus box will only have one white circle in it, the bar will be red, and the text will indicate which eye is the Tobii Eye Tracker is detecting. The participant views a fixation cross, and then listens to an animal noise. Next, a picture is shown, and the participant is asked to make a decision about the relationship of the sound to the picture. Response feedback is provided at the end of each trial. We recommend that you run the experiment first, so you understand what you are creating. This will allow you to learn more from this tutorial.

Goal:

This tutorial illustrates how to add the TET PackageCalls into the FixedPositionAOI.es2 sample experiment included with E-Prime Extensions for Tobii 2.0. When you have completed this tutorial, you will have created a basic “TET-enabled” paradigm.

Overview of Tasks:

- Load FixedPositionAOI and resave it as TETFixedPosition.es2.
- Add the TET PackageFile to the Experiment Object.
- Determine which timing mode is appropriate for your experiment.
- Add the Tobii EyeTracker to Devices Tab and edit the properties to enable communication between the Tobii EyeTracker and TET Server.
- Add the TET PackageCall to open/close the TET package.
- Add the TET PackageCall to display the Track Status Window and perform calibration.
- Add the TET PackageCall to open/close the .gazedata file.
- Add the TET PackageCall to start/stop tracking.
- Add the TET PackageCall to replay the .gazedata.
- Verify the overall experiment structure and run the experiment.

Estimated Time: 20-30 minutes

Task 1: Open the FixedPositionAOI.es2 Experiment in E-Studio

Locate the E-Studio icon in the Start > All Programs > E-Prime 2.0 menu and launch the application by selecting it. Load the FixedPositionAOI.es2 sample experiment.

The E-Studio application is installed as part of the typical E-Prime installation. This application is used to create, modify, and test experiments within E-Prime. Open the E-Studio application, navigate to My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOI, and load the TETFixedPositionAOI\FixedPositionAOI.es2 sample experiment.

1) ***Click*** on the Windows **Start** menu, ***select*** **All Programs**, and then ***select*** **E-Prime**. From the menu, ***click*** on **E-Studio** to launch the application.

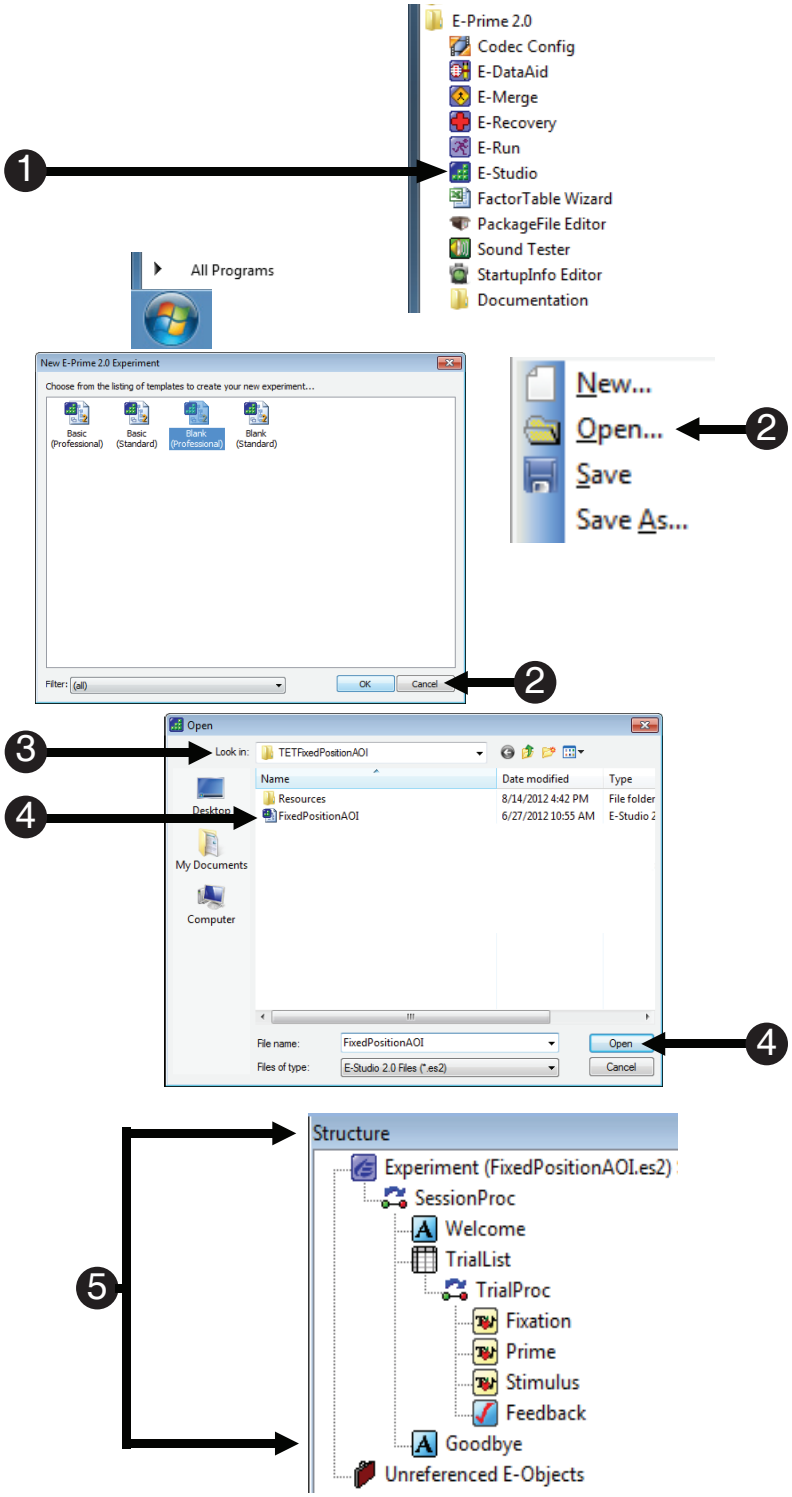
2) ***Click*** the **Cancel** button. ***Select*** **File** > **Open**.

3) ***Navigate*** to the "**…\My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOI**" folder to load the paradigm.

4) ***Select*** the **FixedPositionAOI.es2** file and then ***click*** the **Open** button to load the paradigm into E-Studio.

⚠ **NOTE:** *If you cannot find the FixedPositionAOI.es2 file, you may need to refresh your E-Prime Samples and Tutorials folders. Select Tools\Options… from the E-Studio menu bar then click "Copy Samples and Tutorials to My Experiments folder…"*

5) ***Compare*** the structure of the experiment you have opened to the one shown on the right.



Task 2: Save the Experiment Under a New Name

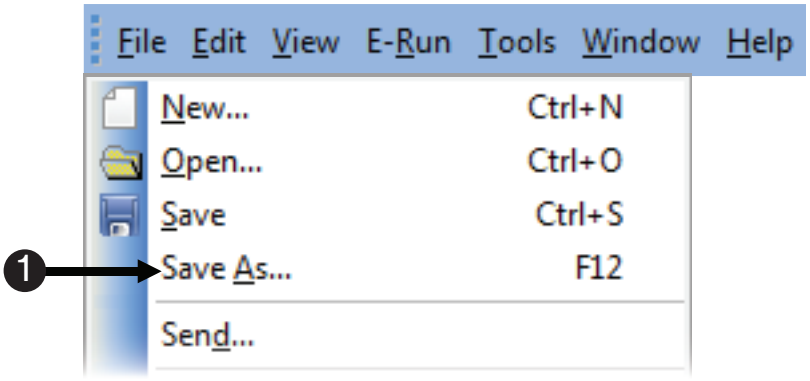
Save the FixedPositionAOI.es2 experiment in the same folder under the new name "TETFixedPositionAOI.es2."

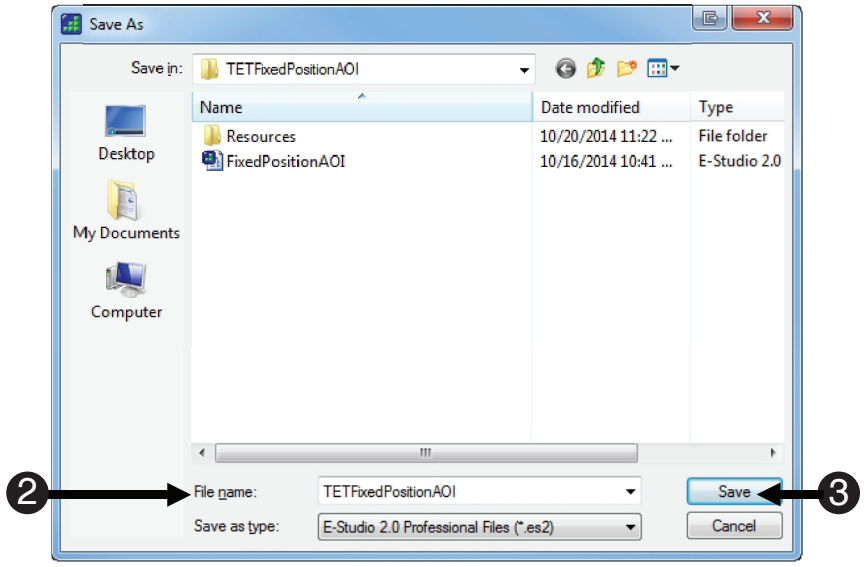
Rename the experiment, but be sure to save it in the same folder ("…\My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOI") so that any resources within the experiment will remain valid and can be reused.

1) ***Select*** **File** > **Save As…** from the application menu bar.

2) ***Type*** "**TETFixedPositionAOI.es2**" as the new name in the **File name** field.

3) ***Click*** the **Save** button.





Task 3: Add the TET Package to the Experiment Property Pages

Open the Property Pages for the Experiment Object and use the Packages tab to add the TET PackageFile to the experiment.

Package Files in E-Prime are cohesive sets of E-Basic routines that are grouped together into a single file that can be maintained externally. In order to gain access to the routines within a package file, you must first add the package file to the experiment. Package files can be added to an experiment using the Packages tab of the Experiment Object Property Pages. The routines that are used to communicate with the TET software at runtime are contained within the TET PackageFile.

1) ***Double click*** the **Experiment** Object at the top of the tree in the **Structure** window.

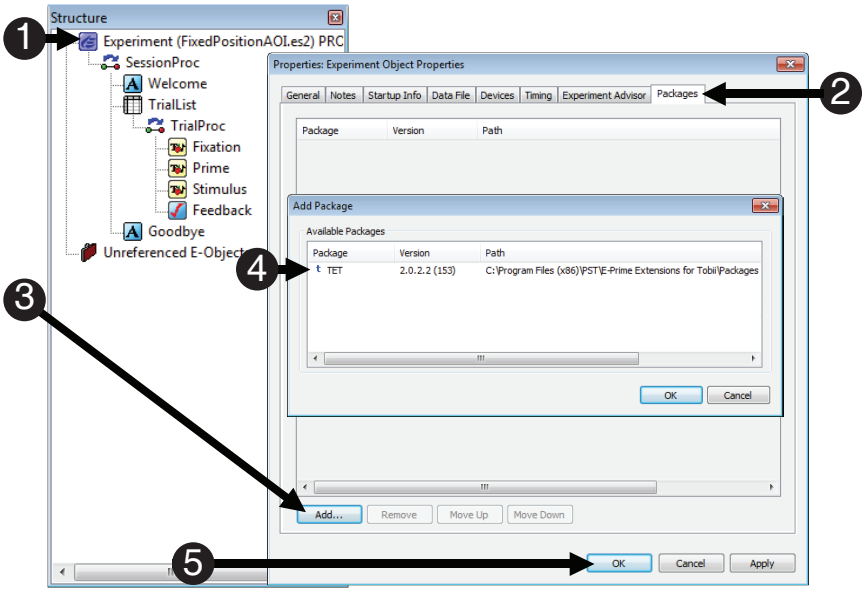
2) ***Click*** on the **Packages** tab of the **Experiment Object Property Pages**.

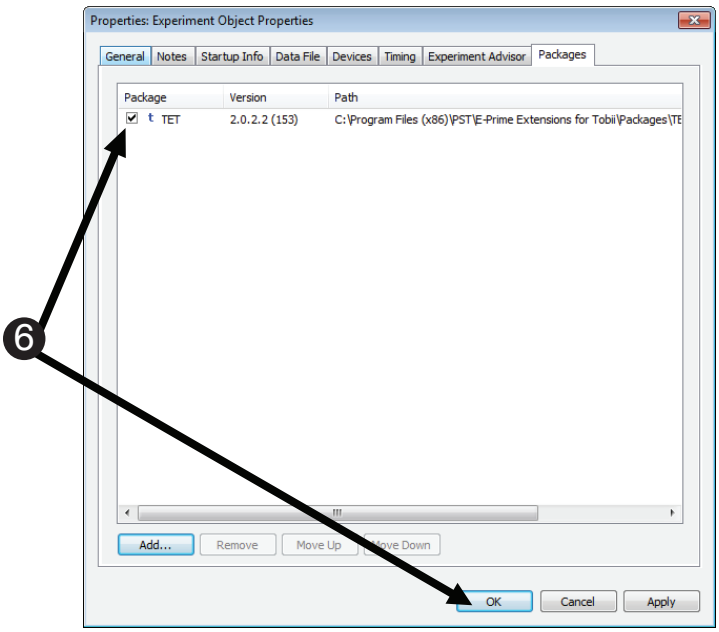
3) ***Click*** the **Add…** button.

4) ***Select*** the **TET** package file in the **Add Package Property Pages.**
   The version numbers may differ.

5) ***Click*** the **OK** button to dismiss the **Add Package Property Pages** dialog.

6) ***Verify*** the **TET PackageFile** is listed under the **Package** column and is checked. Then ***click*** the **OK** button to dismiss the **Property Pages.**
   The Package File version number displayed by E-Studio reflects the version of the TET PackageFile that is currently installed on your machine and may not match the picture.





Task 4: Add the TobiiEyeTracker to the Experiment Properties

Open the Property Pages for the Experiment Object and select the Devices tab to add the TobiiEyeTracker Device to the experiment.

Select the Devices Tab of the Experiment Object Property Pages to add the TobiiEyeTracker Device to the list of devices, and verify that it is the last device shown.

1) ***Double click*** the **Experiment** Object at the top of the tree in the **Structure** window.

2) ***Click*** on the **Devices** tab of the **Experiment Object Property Pages.**

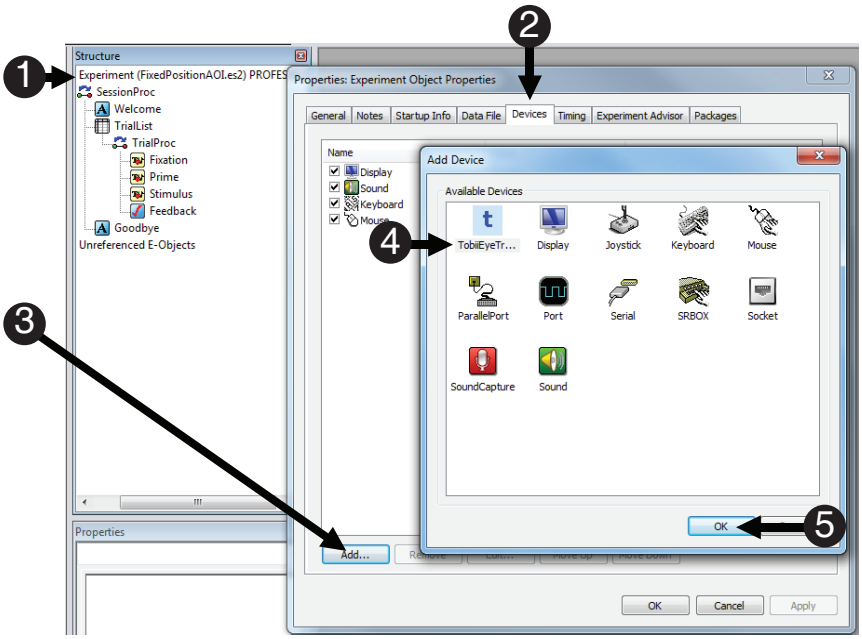
3) ***Click*** the **Add...** button.

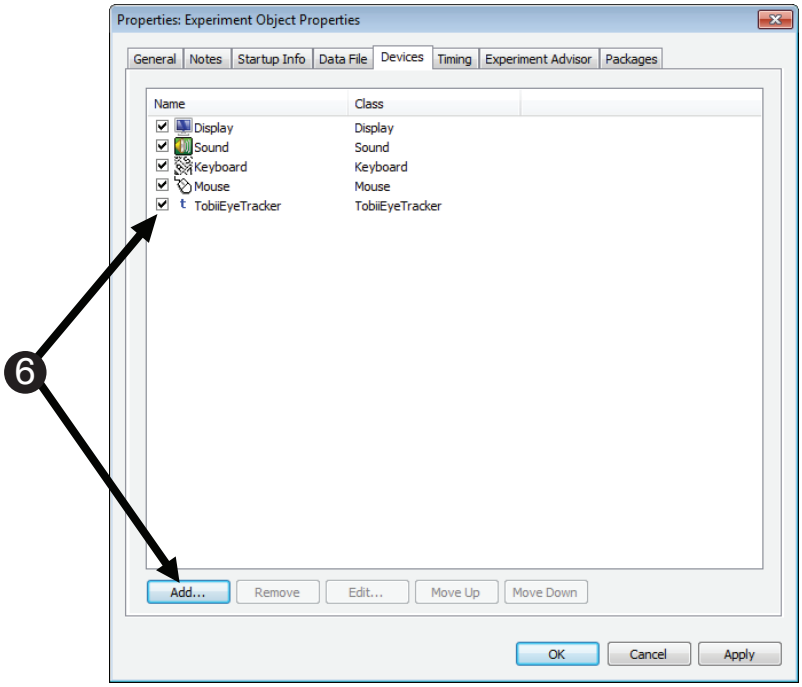
4) ***Select*** the **TobiiEyeTracker Device** in the **Add Device dialog**.

5) ***Click*** the **OK** button to dismiss the **Add Device dialog**.

6) ***Verify*** the **TobiiEyeTracker Device** is listed last under the **Name** column and is checked. Then ***click*** the **OK** button to dismiss the **Add Device dialog**.

   The order of devices from top to bottom is the order in which the devices become enabled via E-Studio at the beginning of an experimental run. Do not move the Tobii Eye Tracker Device to the top as the Display Device must open prior to the TobiiEyeTracker Device. If this order is not followed, the experiment will abort.





Task 5: Edit the TobiiEyeTracker Device Property Pages

Open the TobiiEyeTracker Device Property Pages, specify the TET Server IP Address, and confirm the TCP/IP Port number.

Now that you have added the device, you will need to add the IP address of the TET Server so E-Prime can communicate with it.

1) ***Click*** the **TobiiEyeTracker Device** to highlight it.

2) ***Click*** **Edit.**

3) ***Type*** the **TobiiEyeTracker Device 'monitor serial number.local.'** The monitor serial number is (located on the back of the Eye Tracker).

4) ***Review*** the **TCP/IP Port Number.** *A default value for Port Number is provided automatically. The default value (4455) is correct in most cases and should not be changed.*

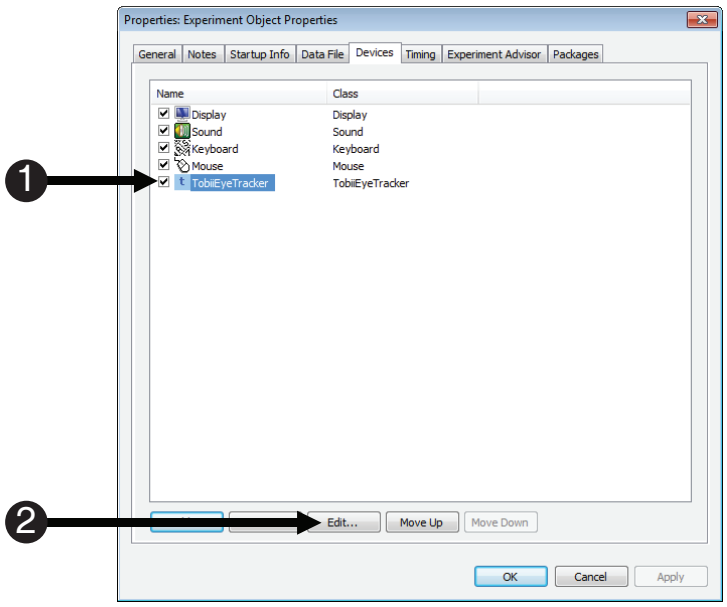
5) Optional name for log file generated for troubleshooting purposes.

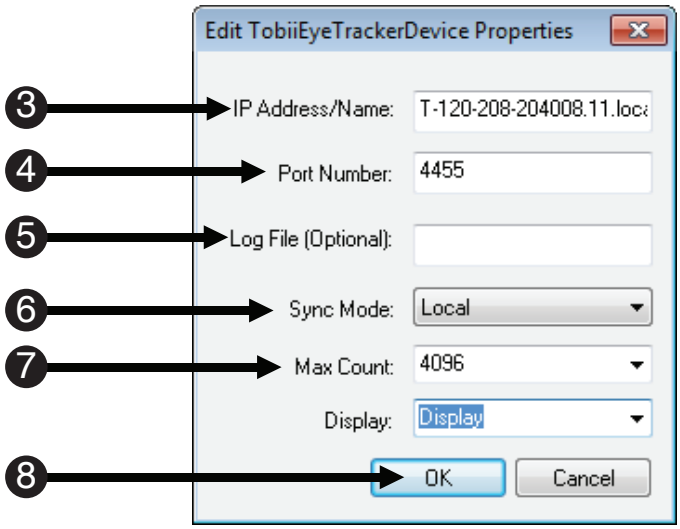
6) ***Confirm*** **Sync Mode** is set to Local. See **Appendix C: Timing and Synchronization,(Page 207)**.

7) **Max Count *sets*** .gazedata buffer size.

⚠ **NOTE:** *This setting controls the amount of time that the eye tracker can save eye gaze data before the oldest data in the buffer is overwritten. The default is 4096 samples. When using a 120Hz eye tracker the buffer will hold ~34 seconds of eye gaze data. You will need to verify that the size of this buffer is sufficiently larger to meet the needs of your experiment. See **Appendix C: Timing and Synchronization, (Page 207)**.*

8) ***Click*** **OK** to accept changes.





Task 6: Add the TETOpen PackageCall to Begin Collecting Data

Add a PackageCall at the beginning of the SessionProc and name the PackageCall TETOpen.

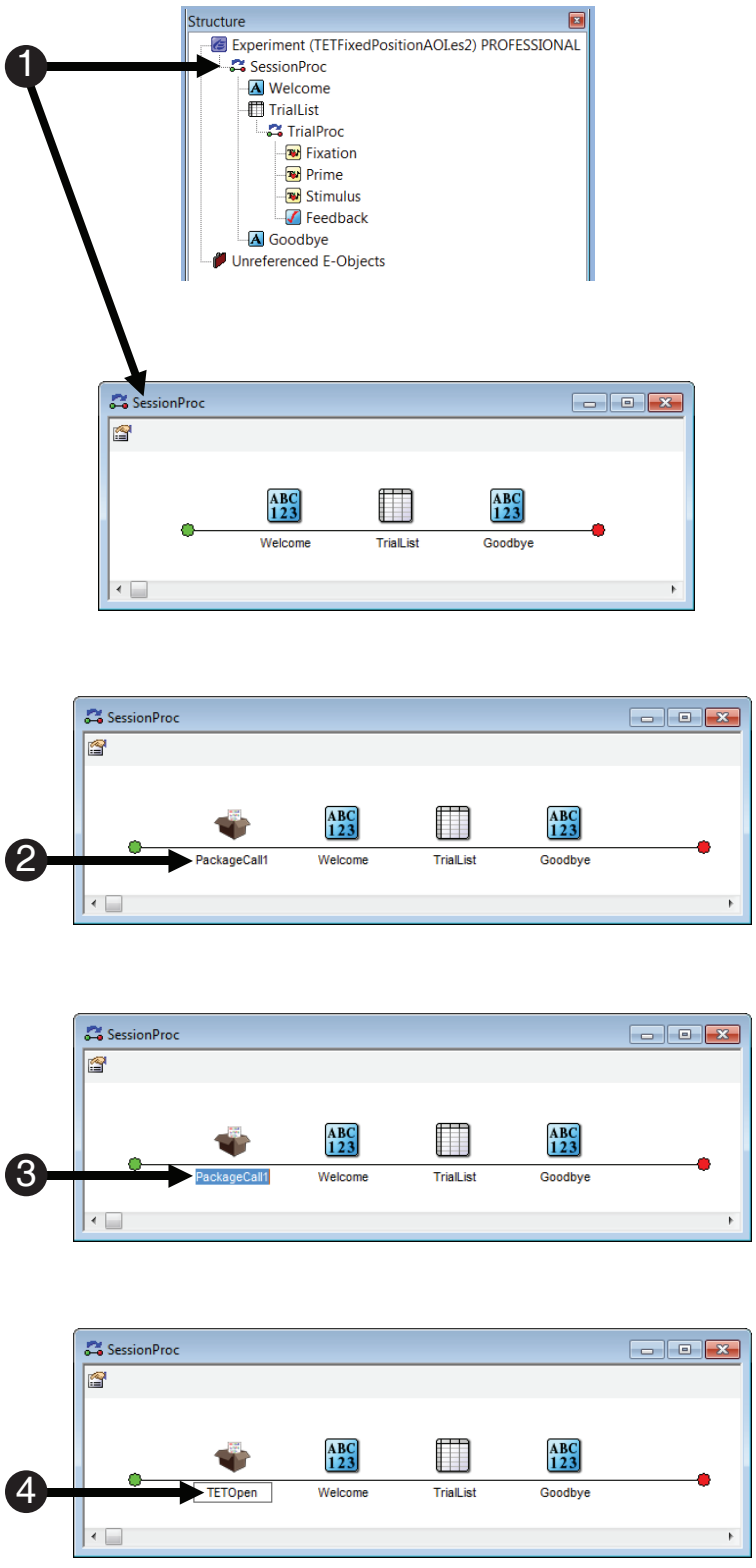
The TETOpen PackageCall opens the TobiiEyeTracker Device in preparation for eye movement data collection. The TETOpen PackageCall will be added at the beginning of the SessionProc for this purpose.

1) ***Double click*** the SessionProc Object to open it in the workspace.

2) ***Drag*** a new **PackageCall** from the E-Studio **Toolbox** and ***drop*** it as the first object in the **SessionProc** procedure. The object will be given a default name of **PackageCall1**.

3) ***Click*** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.

4) ***Type*** "**TETOpen**" as the new object name and then ***press*** **Enter** to accept the change.
   When the Package file defines an icon for the routine, it replaces the default PackageCall icon. In this case, the icon associated with the package file itself will be used.



Task 6 (continued): Add the TETOpen PackageCall to Begin Collecting Data

Configure the TETOpen PackageCall to call the Open routine to the TET package and accept the default properties.

The Property Pages of the PackageCall are used to specify which Package File and Routine are to be called in each instance. After a Package is selected within the interface, the Routine dropdown list will be populated with all of the routines contained within the package. After you select a Routine, the Parameters field will be set to the default parameters and the Description field will be filled with the text that the package file author included for the selected routine. You can refer to the Description field for information about each parameter in the list (any parameter in double quotes indicates string data). The TETOpen PackageCall includes parameters that allow you to turn on/off the communication between the E-Prime experiment and the Tobii Eye Tracker.

5) ***Double click*** the **TETOpen PackageCall** on the **SessionProc** to display its **Property Pages.**

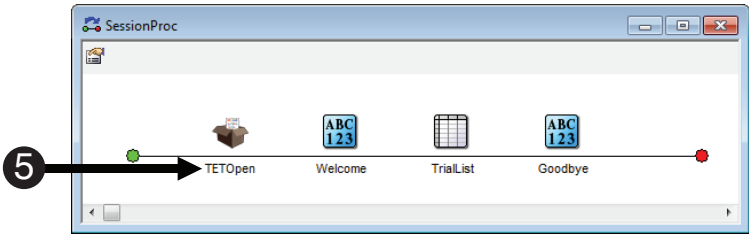
6) ***Select*** **TET** from the **Package** dropdown list.

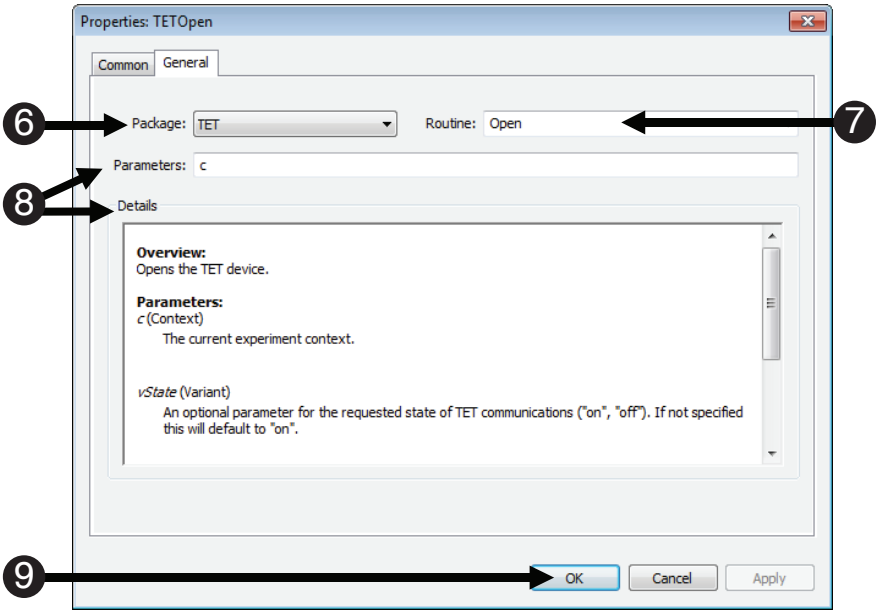
7) ***Select*** **Open** from the **Routine** dropdown list.

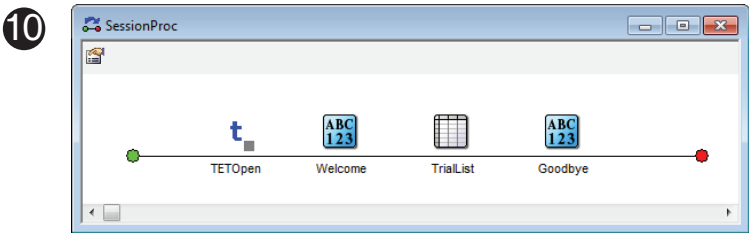
8) ***Review*** the **TETOpen** parameters listed in the **Parameters** and **Description** fields.

9) ***Click*** the **OK** button to accept the changes and dismiss the **TETOpen Property Pages.**

10) ***Confirm*** your **SessionProc** is identical to the example shown in Step 10.







Task 7: Add the TETCalibRegular PackageCall to Enable the Calibration Routines

Add a PackageCall to the SessionProc and name the object TETCalibRegular.

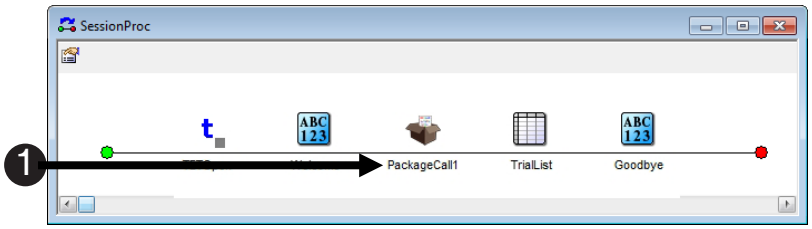
The TETCalibRegular PackageCall is used to display the Track Status window and calibrate the participant. This PackageCall will display the same window and dots that you see when running a calibration or study in Tobii Studio. A black box will appear with two white circles representing your eyes. Calibration can be started with any key press. At the end of the calibration you will be shown the results of the calibration and prompted to accept or reject the results. The default number of calibration points is nine. This can be customized with the information contained in **TET_CalibSetDefaultNumPoints, (Page 187)**, and via the InLine script. At the end of the calibration you will be shown the results of the calibration and prompted to accept or reject the results.

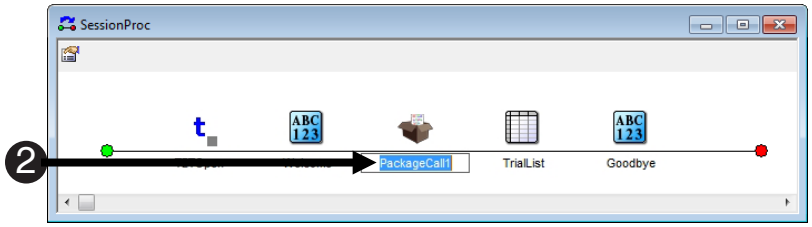
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the Welcome Object in the **SessionProc** procedure. The object will be given a default name of **PackageCall1**.

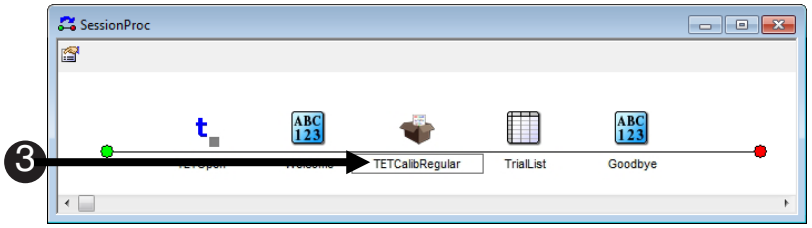
2) ***Click*** on the **PackageCall1** to select it, then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.

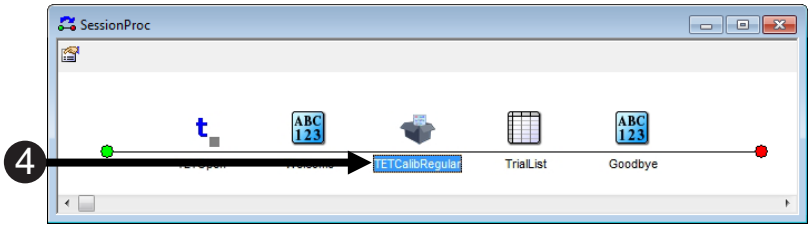
3) ***Type*** "**TETCalibRegular**" as the new object name and ***press*** **Enter** to accept the change.

4) ***Double click*** the **TETCalibRegular** PackageCall on the **SessionProc** to display its **Property Pages.**









Task 7 (continued): Add the TETCalibRegular PackageCall to Enable the Track Status Window

Add a PackageCall to the SessionProc and name the object TETCalibRegular.

After you select the TET Package and CalibRegular as the Routine, the Parameters field will be set to the default parameter, "c," and the Description field will be filled with the text that the Package File author included for the selected routine. This PackageCall automatically clears any previous calibration that has currently been run on the hardware because it assumes that you want to use the calibration you are about to do for the experiment you are running.

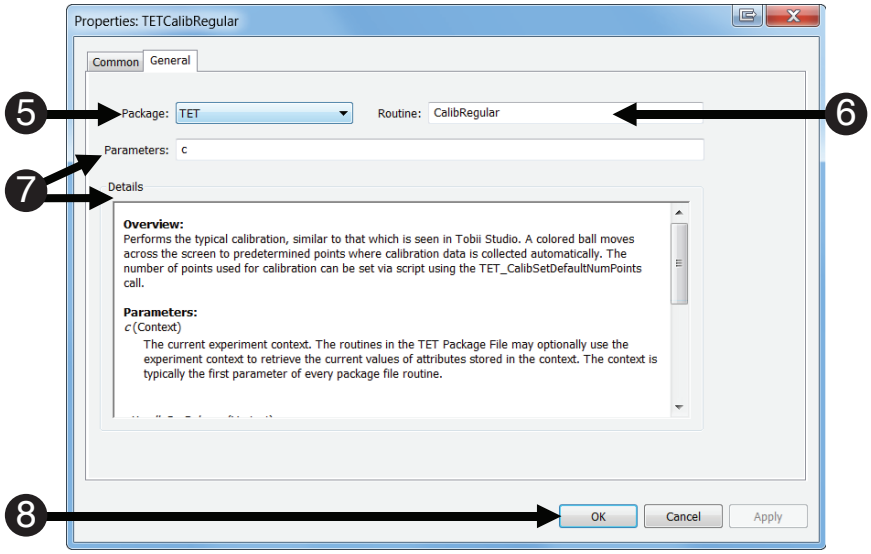
5) ***Select*** **TET** from the **Package** dropdown list.

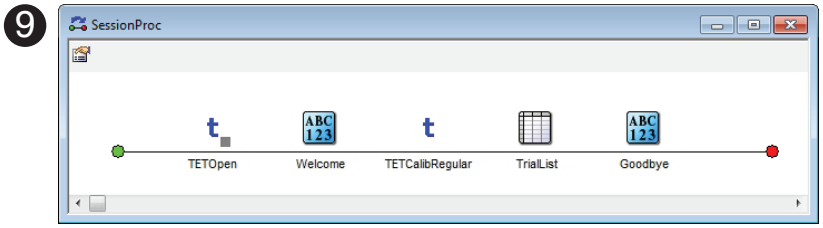
6) ***Select*** **CalibRegular** from the **Routine** dropdown list.

7) ***Review*** the **TETCalibRegular** parameters listed in the **Parameters** and **Description** fields.

8) ***Click*** the **OK** button to accept the changes and dismiss the **Property Pages**.

9) ***Confirm*** your **SessionProc** is identical to the example shown.



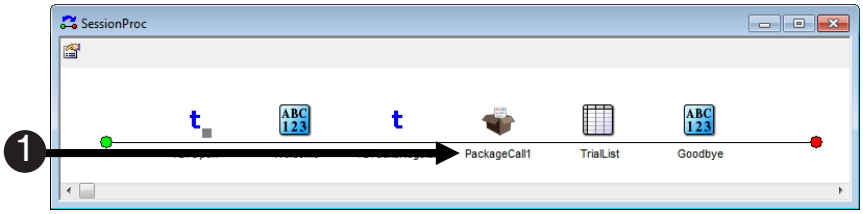


Task 8: Add the TETOpenGazeDataFile PackageCall to Create and Open the Tab Delimited .gazedata File

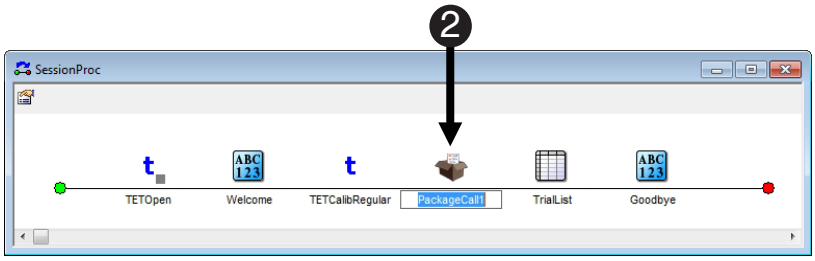
Add a PackageCall to the SessionProc and name the PackageCall TETOpenGazeDataFile.

The next step is to add the TETOpenGazeDataFile PackageCall after the TETOpen PackageCall. It is recommended that the TETOpenGazeDataFile PackageCall follow the TETCalibRegular PackageCall. The purpose of this call is to open the tab delimited .gazedata file that will be created (in addition to the .edat2 file) once an experiment has completed. (Operations associated with data collection and output will be discussed in length in the following tutorial).

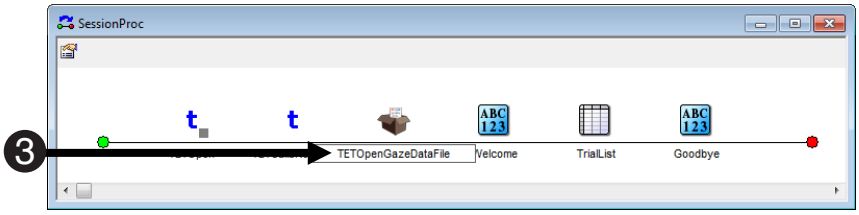
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **TETCalibRegular** PackageCall in the **SessionProc** procedure. The object will be given the default name of **PackageCall1**.



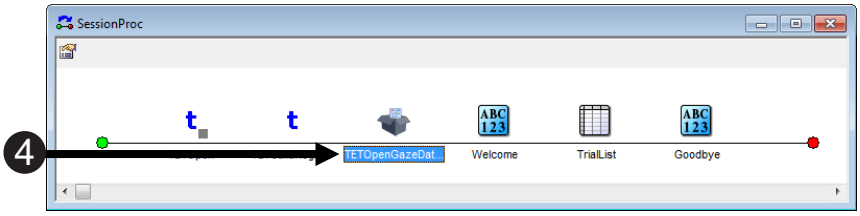
2) ***Click*** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.



3) ***Type*** "**TETOpenGazeDataFile**" as the new object name and then ***press*** **Enter** to accept the change.



4) ***Double click*** the **TETOpenGazeDataFile** PackageCall on the **SessionProc** to display its **Property Pages.**



Task 8 (continued): Add the TETOpenGazeDataFile PackageCall to Create and Open the Tab Delimited .gazedata File

Call the OpenGazeDataFile routine of the TET package and accept the default properties.

The default parameters of this PackageCall specify the name of the output file. If the parameter is not specified a filename will be created in the form of DataFile.BaseName. DataFile.BaseName defaults to [ExperimentName]-[Subject#]-[Session#]. If you wish to change the name of the output file, use quotation marks to enclose the string. The proper syntax to do this is **c, "your desired filename"**.

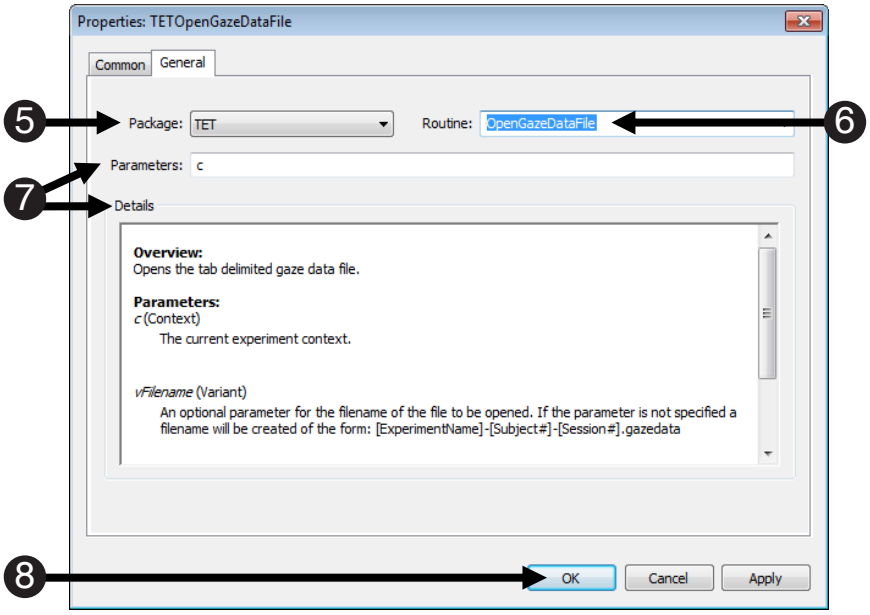
5) ***Select*** **TET** from the **Package** dropdown list.

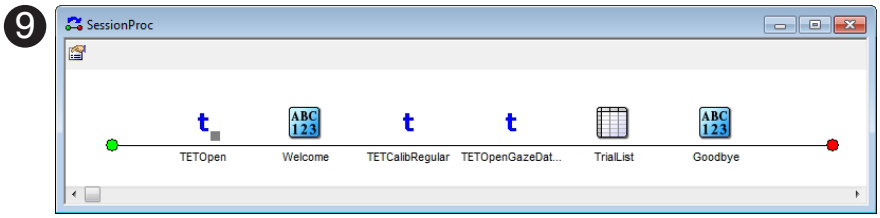
6) ***Select*** **OpenGazeDataFile** from the **Routine** dropdown list.

7) ***Review*** the **TETOpenGazeDataFile** parameters listed in the **Parameters** and **Description** fields.

8) ***Click*** the **OK** button to accept the changes and dismiss the **TETOpenGazeDataFile Property Pages**.

9) ***Confirm*** your **SessionProc** is identical to the example shown.





Task 9: Add the TETStartTracking PackageCall to Begin Tracking Eye Movements

Add a PackageCall to the TrialProc and name the PackageCall TETStartTracking.

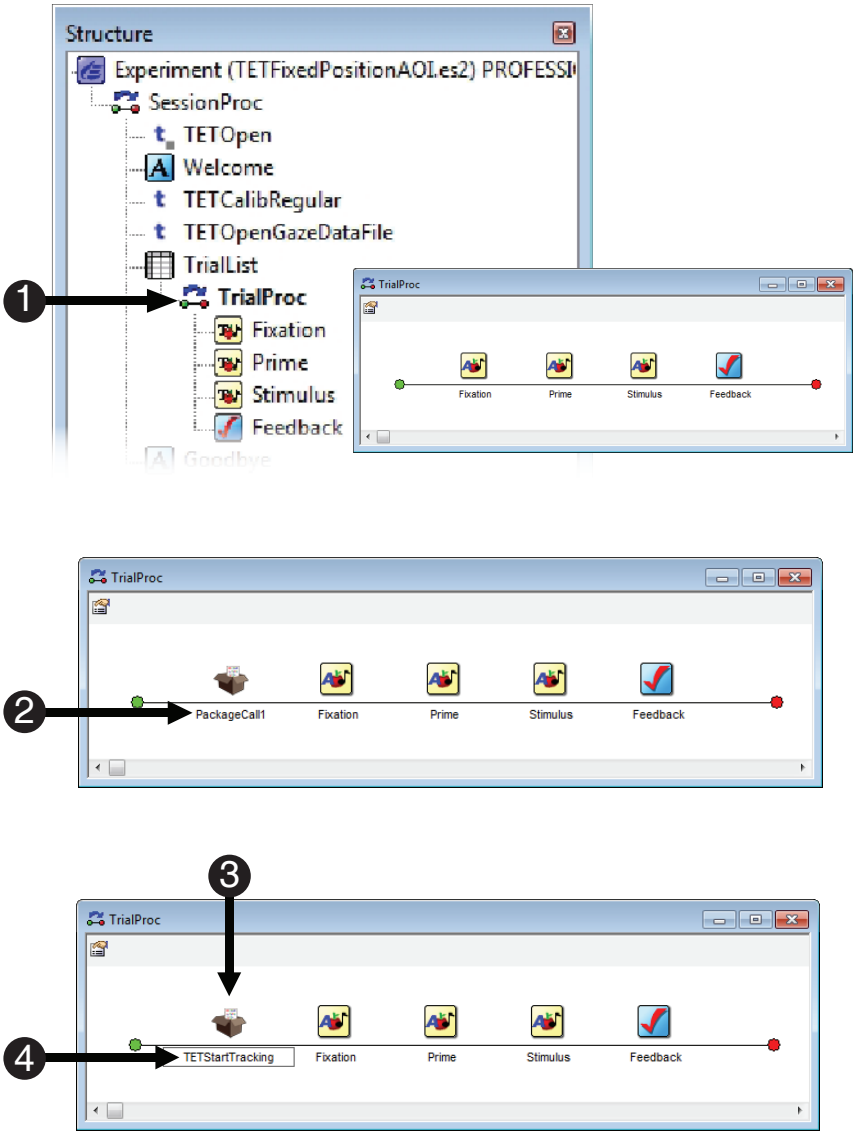
The next step is to add the TETStartTracking PackageCall. This PackageCall should be placed at the location in the procedure where you decide you want to start tracking eye movements. This is often at the beginning of a trial, but may vary depending on the experiment. See **Chapter 3: Tobii PackageCall Reference, (Page 147)**.

1) ***Double click*** the **TrialProc** Object to open it in the workspace.

2) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it *before* the Fixation Object in the **TrialProc** procedure. The object will be given a default name of **PackageCall1**.

3) ***Click*** on the **PackageCall1** to select it, then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.

4) ***Type*** "**TETStartTracking**" as the new object name and then ***press*** **Enter** to accept the change.



Task 9 (continued): Add the TETStartTracking PackageCall to Begin Tracking Eye Movements

Configure the TETStartTracking PackageCall to call the StartTracking routine of the TET package and accept the default properties.

⚠**NOTE:** *The default values of this PackageCall clear the TET history when the eye tracking is started. If you do not wish to clear the TET history, then you must set this value to "False."*

5) ***Double click*** the **TETStartTracking** PackageCall on the **TrialProc** to display its **Property Pages**.

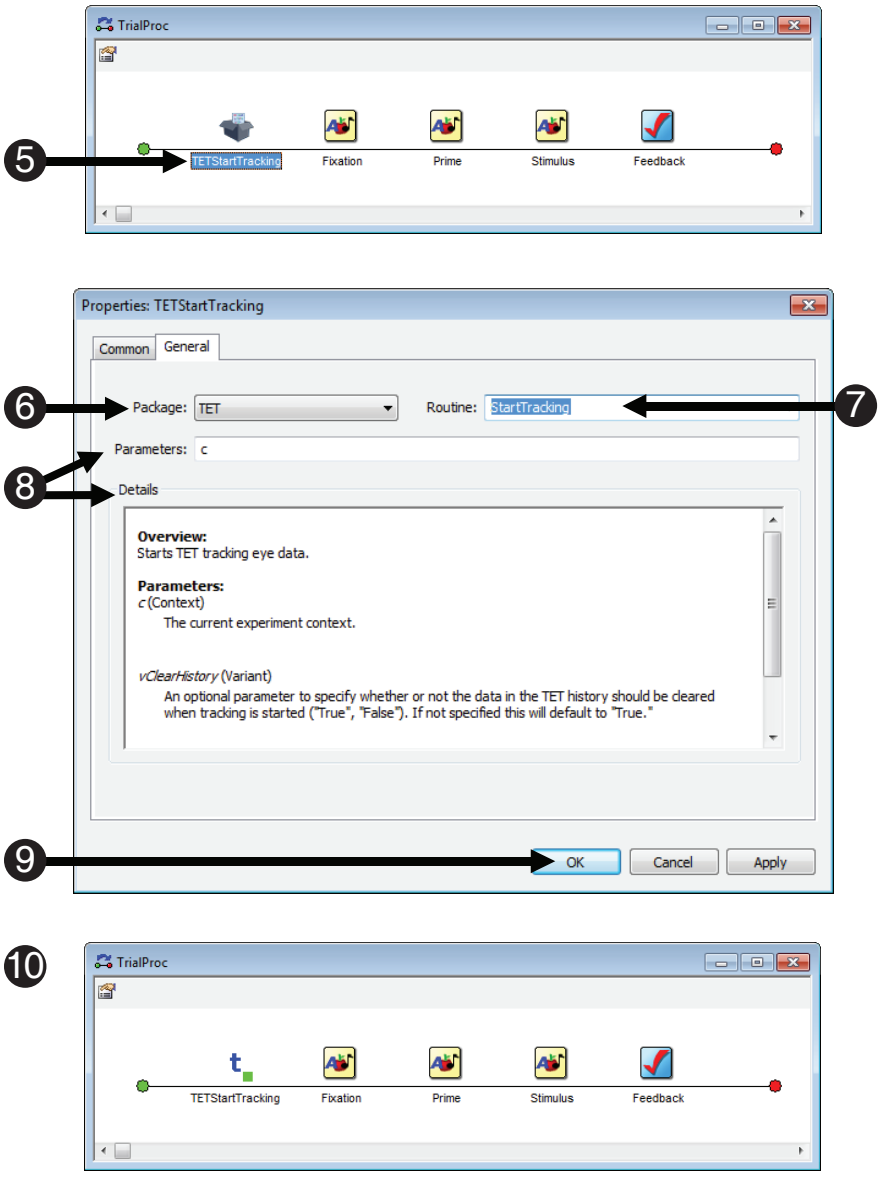
6) ***Select*** **TET** from the **Package** dropdown list.

7) ***Select*** **StartTracking** from the **Routine** dropdown list.

8) ***Review*** the **TETStartTracking** parameters listed in the **Parameters** and **Description** fields.

9) ***Click*** the **OK** button to accept the changes and dismiss the **TETStartTracking Property Pages**.

10) ***Confirm*** your **TrialProc** is identical to the example shown.



Task 10: Add the TETWaitForFixation PackageCall to Halt the Trial until the Participant Fixates on a Particular Object for a Given Amount of Time

Add a PackageCall to the TrialProc and name the PackageCall TETWaitForFixation.

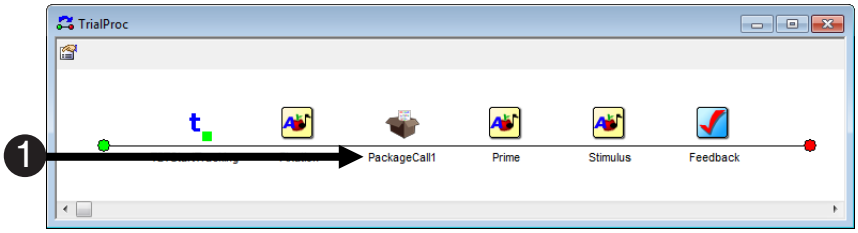
The next step is to add the TETWaitForFixation PackageCall. This PackageCall is used to halt the beginning of a trial until the participant fixates on a particular object on the screen for a set amount of time. A commonly used fixation point is a simple fixation cross, but the particular fixation object is not important and will vary from experiment to experiment. In this example, the PackageCall should follow the Fixation Object.

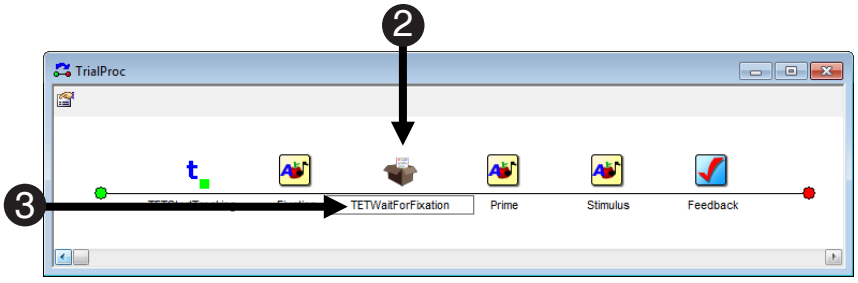
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Fixation** Object in the **TrialProc** procedure. The object will be given a default name of **PackageCall1**.

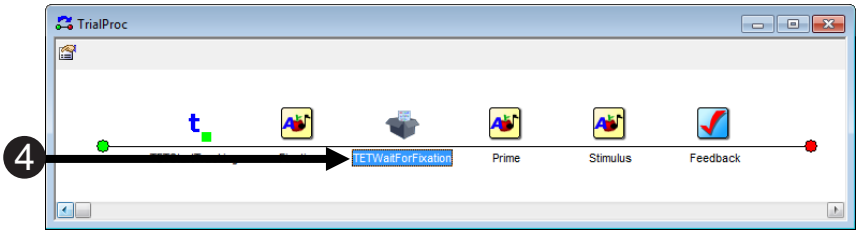
2) ***Click*** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.

3) ***Type*** "**TETWaitForFixation**" as the new object name and then ***press*** **Enter** to accept the change.

4) ***Double click*** the **TETWaitForFixation** PackageCall Object on the **TrialProc** to display its **Property Pages.**







Task 10 (continued): Add the TETWaitForFixation PackageCall to Halt the Trial until the Participant Fixates on a Particular Object for a Given Amount of Time

Configure the TETWaitForFixation PackageCall to call the WaitForFixation routine of the TET package and edit the Parameters.

This PackageCall is used in conjunction with a Slide Object which will be the Fixation Object in this example. When using this call, it is important to account for all of the Parameters and set up your Slide Object accordingly. We will set up the Parameters for the Slide Object now, and in Task 11 and 12 we will set and discuss the Fixation Slide Object in detail.

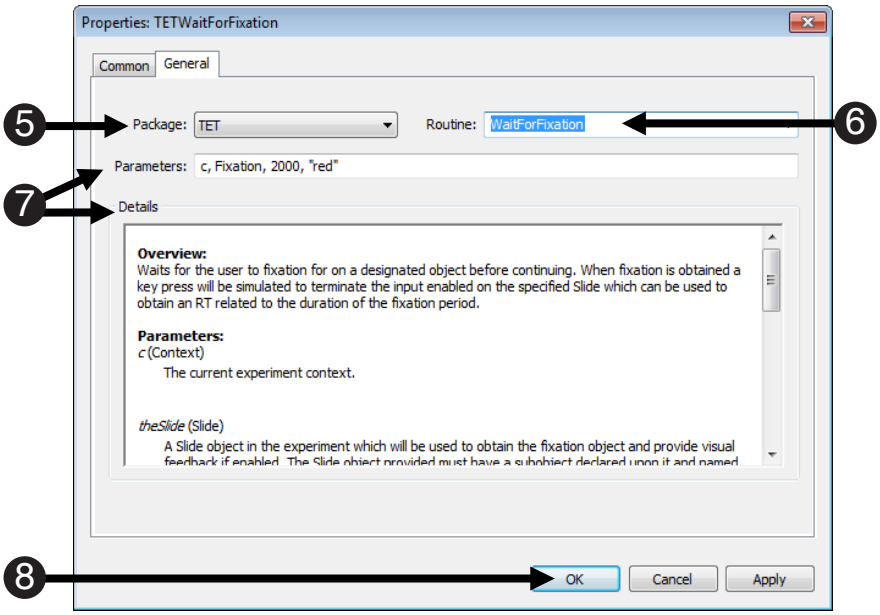
5) ***Select*** **TET** from the **Package** dropdown list.

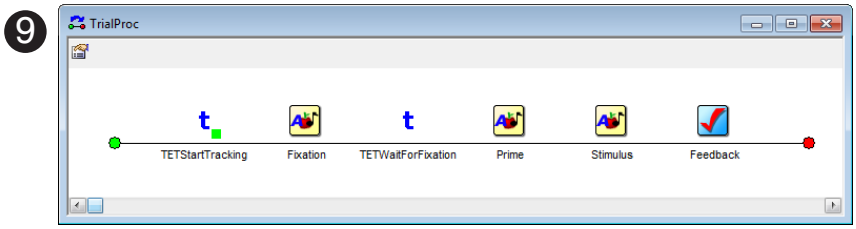
6) ***Select*** **WaitForFixation** from the **Routine** dropdown list.

7) ***Edit*** the **Parameters** to read*:*
   c,Fixation, 2000, “red”
   Tasks 11 and 12 will revisit this issue in depth.

8) ***Click*** the **OK** button to accept the changes and dismiss the **TETWaitForFixation Property Pages**.

9) ***Confirm*** your **TrialProc** is identical to the example shown.





Task 11: Confirm the Sub-Object on the Fixation Slide Object will Display the Desired Fixation.

Add a Sub-Object to the Fixation Slide and Edit the Sub-Object to display a "+" sign.

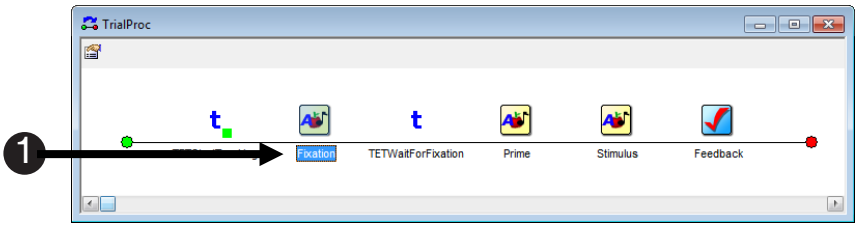
In the last step, we configured the TETWaitForFixation PackageCall to communicate with the Fixation Slide Object. Now we will confirm the Slide Object displays a "+" on the Fixation Sub-Object. This step can be confusing because both the Slide and the Slide Sub-Object are named Fixation. When using the TETWaitforFixation package call, it is imperative to name the Slide Sub-Object Fixation.

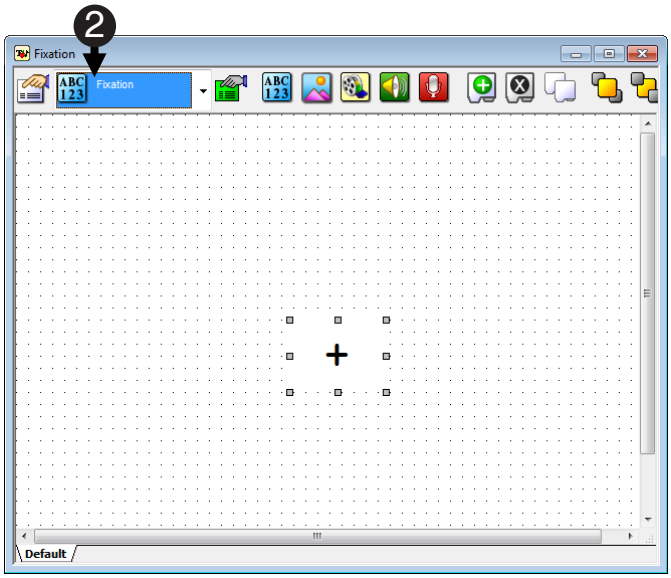
1) ***Double click*** the **Fixation Slide** Object on the **TrialProc** to ***open*** it in the **workspace**.

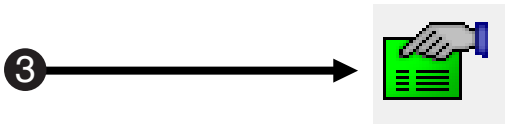
2) ***Select*** **Fixation Sub-Object** from the dropdown list in the **Slide toolbar**.

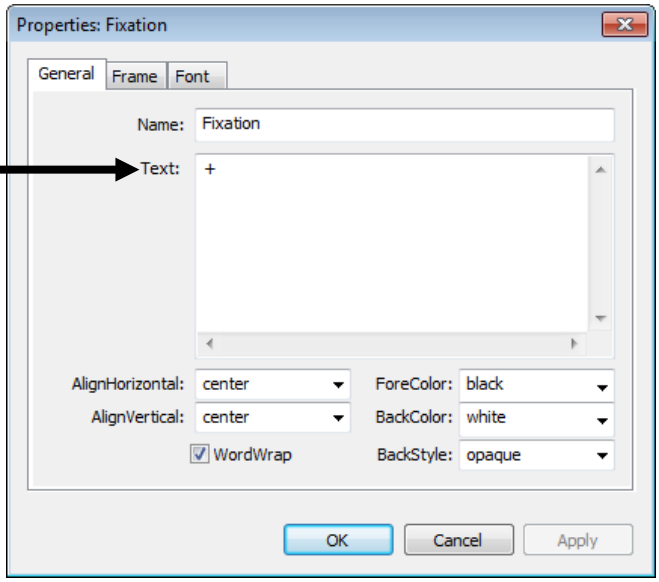
3) ***Click*** the **Property Pages** button to open the **SlideText Sub-Object Fixation Sub-Object Properties.**

4) ***Confirm*** the **Text** reads **+.** ***Press*** **OK** to accept the changes and dismiss the **Fixation Property Pages.**









Task 12: Modify the Fixation Slide Object to work in Conjunction with the TETWaitForFixation PackageCall

Configure the Fixation Object to accept input from the TETWaitForFixation PackageCall.

The general method employed by the TETWaitForFixation PackageCall is one of active processing. The zero duration of the Slide Object will simply ensure that the Object is displayed on the screen, but that the PackageCall is executed immediately after. The Input Mask parameters are necessary because the PackageCall enters a loop that will only exit once a response has been made to the Slide Object (hence the TimeLimit and EndAction property settings). The PackageCall will "keep track" of the participant's fixations by actively accessing the eye tracking data stream. Once a fixation has been made on the Fixation Sub-Object for a predetermined amount of time (as specified in the parameters of the PackageCall), a key press will be simulated, terminating the loop and thus moving on with the rest of the experiment.

1) ***Select*** **Default** from the **Fixation** dropdown menu in the slide toolbar.
⚠ **NOTE:** *We are selecting the entire Slide.*

2) ***Click*** the white **Properties** button.

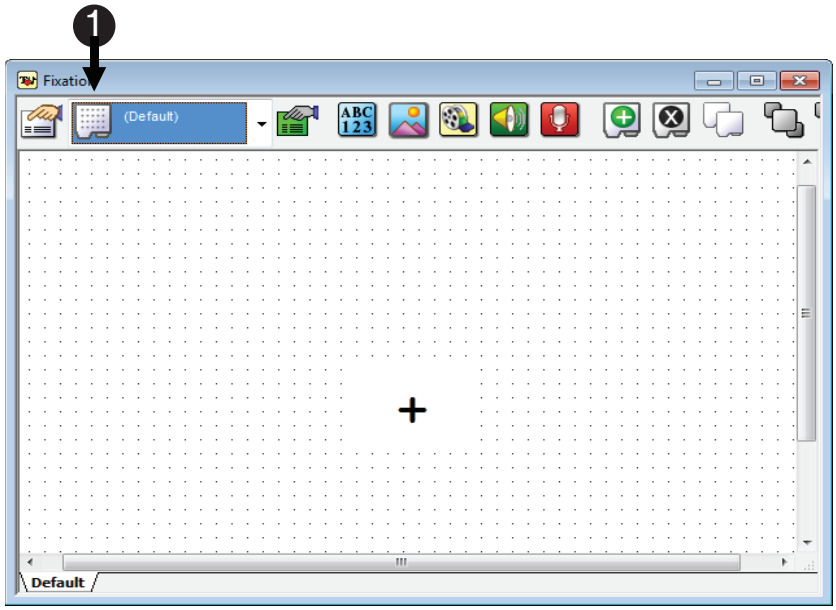
3) ***Select*** the **Duration/Input** tab.

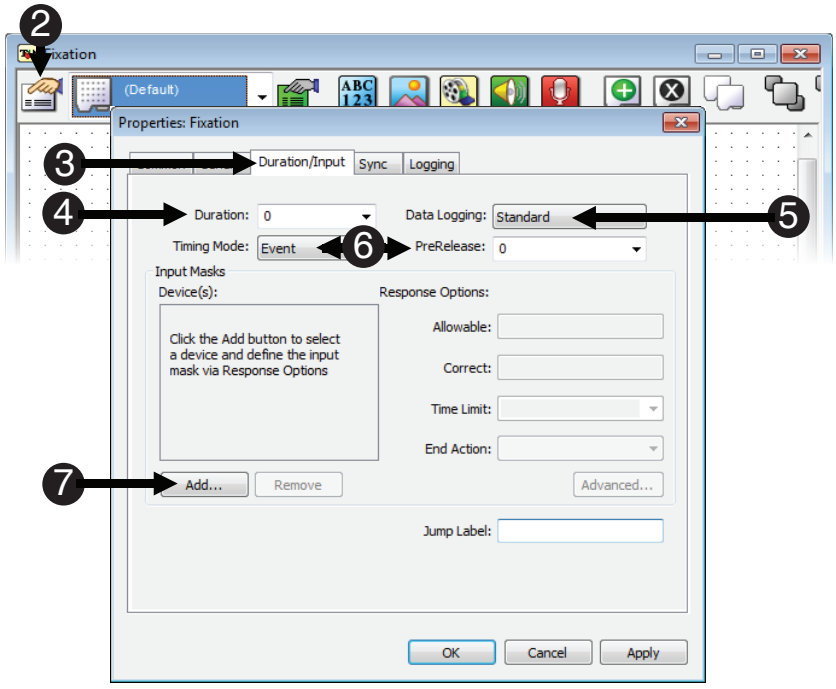
4) ***Edit*** the **Duration** to **0**.

5) ***Confirm*** the **Data Logging Menu** reads **Standard.**

6) ***Verify*** that the **Timing Mode** is **Event** and ***edit*** the **PreRelease** to be **0**.

7) If Keyboard is not listed in the devices, ***click*** the **Add** button in the **Input Masks** area.





Task 12: (continued) Modify the Fixation Slide Object to Work in Conjunction with the TETWaitForFixation PackageCall

Configure the Fixation Object to accept input from the TETWaitForFixation PackageCall.

Using the EET, it is possible to utilize eye gaze data both in “passive” and “active” modes. “Passive” mode refers to the process whereby E-Prime takes in eye gaze data combining it with E-Prime data for later analysis. “Active” mode refers to the additional use of eye gaze data by E-Prime at runtime. In “Active” mode, E-Prime uses the eye gaze data as a response input (similar to the mouse), allowing the paradigm to respond or make decisions based on eye gaze characteristics. The use of TETWaitForFixation is an example of “Active” processing.

8) ***Select*** **Keyboard.**

9) ***Click*** **OK.**

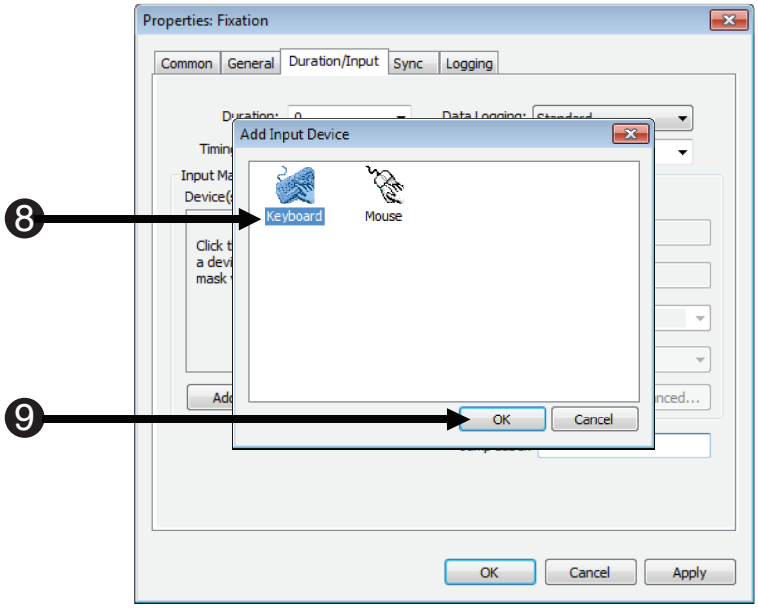
10) ***Verify*** that **Keyboard** is listed *under* **Device(s)** and that it is checked.

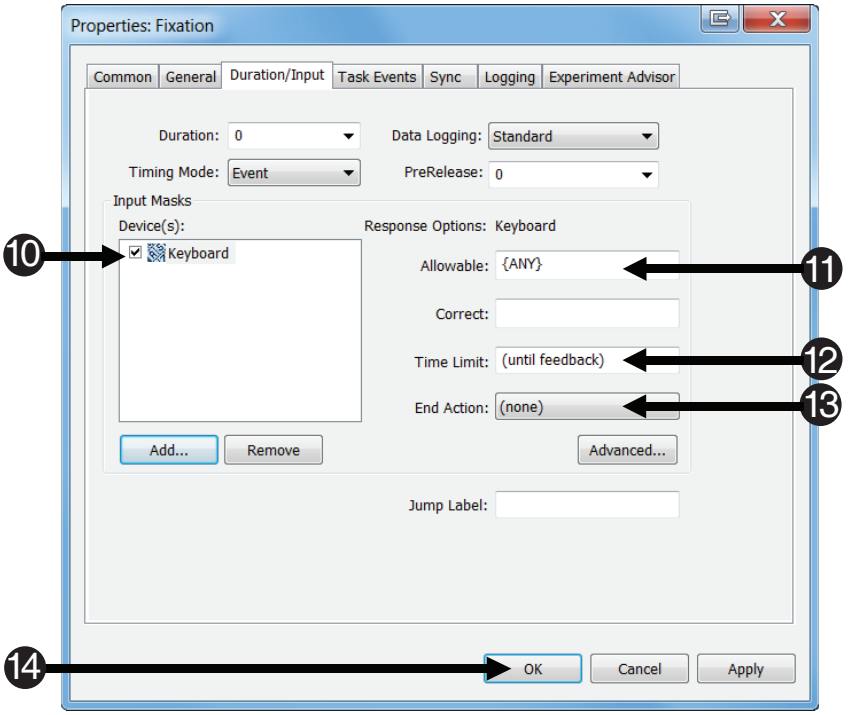
11) ***Verify*** that **Allowable** reads “{ANY}.”

12) ***Verify*** the **Time Limit** is set to “(until feedback).”

13) ***Confirm*** the **End Action** reads “(none).”

14) ***Click*** **OK.**
The RT on the Slide Object will indicate the time elapsed until the participant’s fixation was accepted.





Task 13: Add the TETStopTracking PackageCall to Stop Tracking Eye Movements

Add a PackageCall to the TrialProc and Name the PackageCall TETStopTracking.

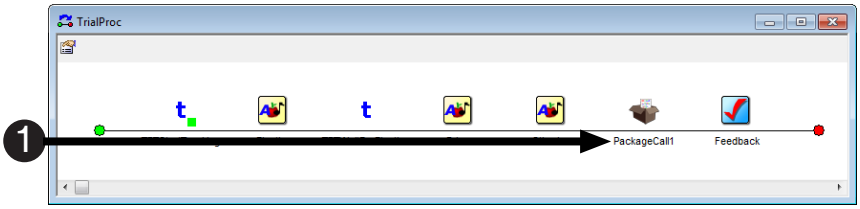
Once the critical stimuli have been presented, it is appropriate to place the TETStopTracking PackageCall. It is rare that an experimenter would want to collect .gazedata on the Feedback Object, so a common place for the PackageCall is immediately after the last critical stimulus and prior to the Feedback Object, although this timing may vary.

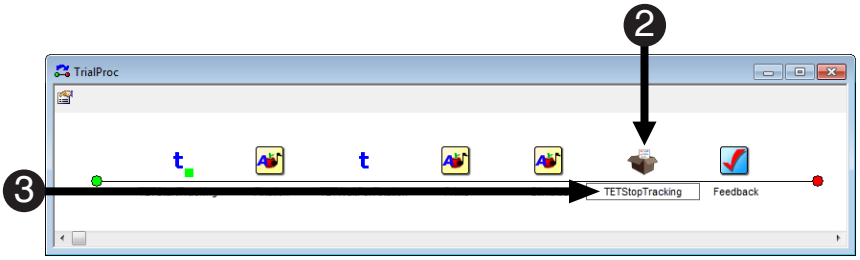
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Stimulus** Object in the **TrialProc** procedure. The object will be given a default name of **PackageCall1**.

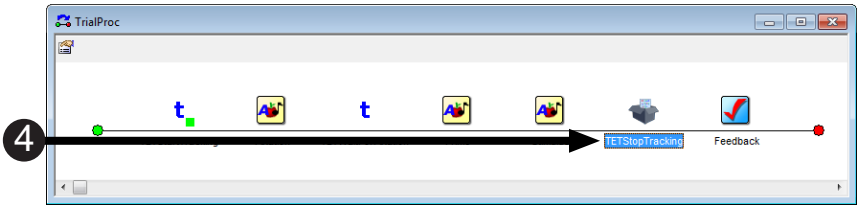
2) **Click** on the **PackageCall1** to select it then ***press*** **F2** to rename the Object.
   You may alternatively right click on the object and select Rename from the context menu.

3) ***Type*** "**TETStopTracking**" as the new object name and then ***press*** **Enter** to accept the change.

4) ***Double click*** the **TETStopTracking** PackageCall Object on the **TrialProc** to display its **Property Pages**.







Task 13 (continued): Add the TETStopTracking PackageCall to Stop Tracking Eye Movements

Configure the TETStopTracking PackageCall to call the StopTracking routine of the TET package and accept the defaults.

Once the PackageCall has been added to the experiment designate TET as the package and StopTracking as the routine. Accept the default parameters.

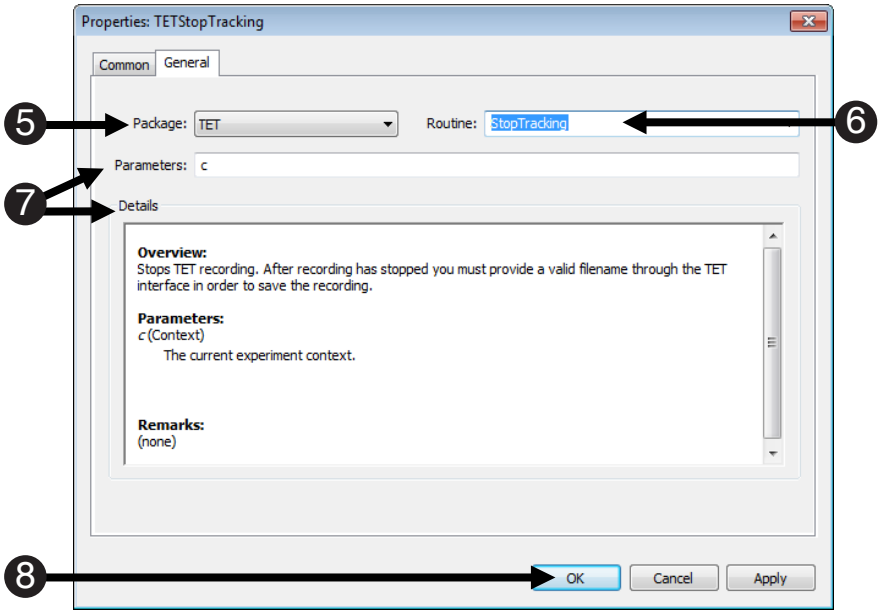
5) ***Select*** **TET** from the **Package** dropdown list.

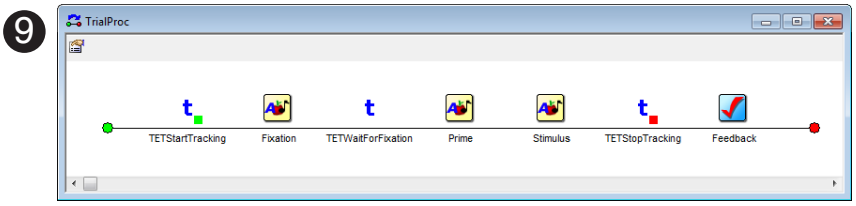
6) ***Select*** **StopTracking** from the **Routine** dropdown list.

7) ***Review*** the **TETStopTracking** parameters listed in the **Parameters** and **Description** fields.

8) ***Click*** the **OK** button to accept the changes and dismiss the **Property Pages.**

9) ***Confirm*** your **TrialProc** is identical to the example shown.



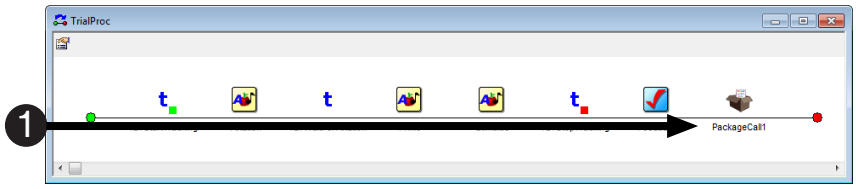


Task 14: Add the TETGazeReplay PackageCall to Visually Replay the Eye Movements

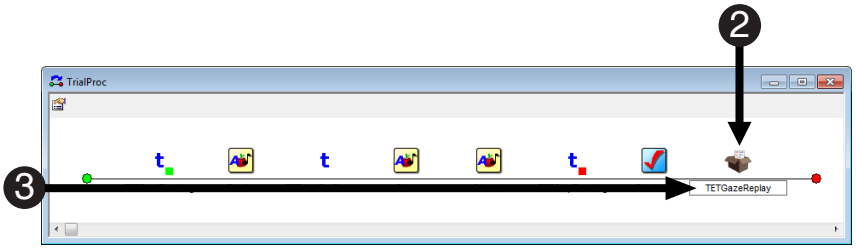
Add a PackageCall to the TrialProc and Name the PackageCall TETGazeReplay.

The paradigm must be explicitly designed to overlay the gaze replay data, while the stimulus is repeated in the same order it was presented to the participant during the experimental run.
It is your responsibility to ensure the stimulus order is accurately synchronized with the gaze replay.

1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Feedback** Object in the **TrialProc** procedure. The object will be given a default name of **PackageCall1**.

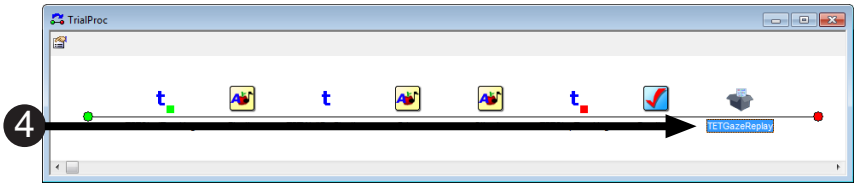


2) **Click** on the **PackageCall1** to select, it then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.



3) ***Type*** "**TETGazeReplay**" as the new object name and then ***press*** **Enter** to accept the change.

4) ***Double click*** the **TETGazeReplay** PackageCall Object on the **TrialProc** to display its **Property Pages**.



Task 14 (continued): Add the TETGazeReplay PackageCall to Visually Replay the Eye Movements

Configure the TETGazeReplay PackageCall to call the GazeReplay routine of the TET package and edit the Parameters.

In the TETFixedPositionAOI experiment, the critical stimulus was presented by the Stimulus Object. The parameters indicated for this experiment are such that the Stimulus Object is presented on the screen and the eye movements are replayed starting at the time of the Prime Object (Prime.OnsetTime), and continuing on until TETStopRecording is called. It is possible to designate a specific time in which the gaze replay should stop via the last optional parameter. However it is unnecessary in this case because the TETStopTracking PackageCall is called immediately after the offset of the Stimulus Object which supplants the use of the Optional parameters.

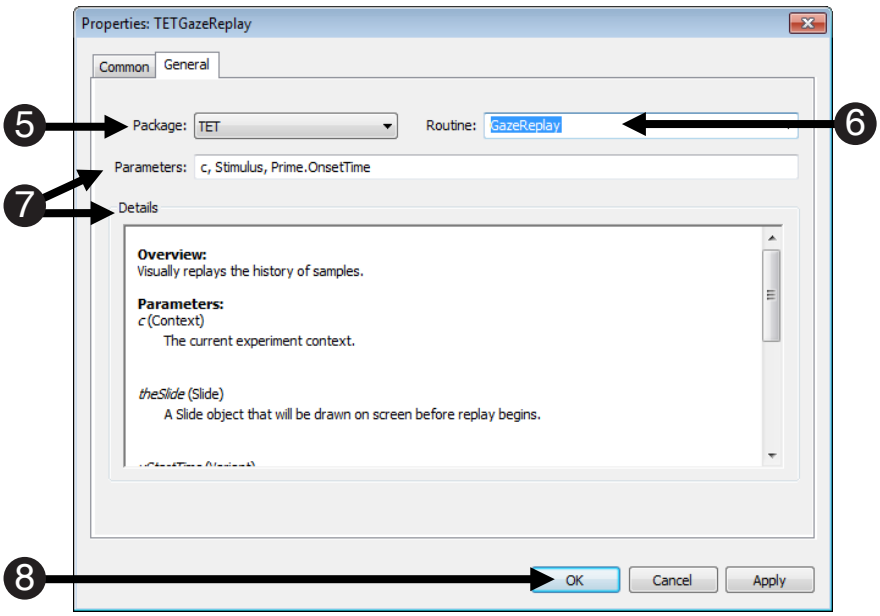
5) ***Select*** **TET** from the **Package** dropdown list.

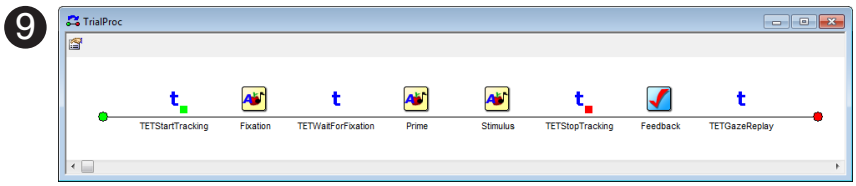
6) ***Select*** **GazeReplay** from the **Routine** dropdown list.

7) ***Edit*** the **TETGazeReplay** parameters listed in the **Parameters** to read: *c, Stimulus, Prime.OnsetTime*

8) ***Click*** the **OK** button to accept the changes and dismiss the **Property Pages.**

9) ***Confirm*** your **TrialProc** is identical to the example shown.





Task 15: Add the TETCloseGazeDataFile PackageCall to End Data Collection

Add a PackageCall at the end of the SessionProc and Name the Object TETCloseGazeDataFile PackageCall.

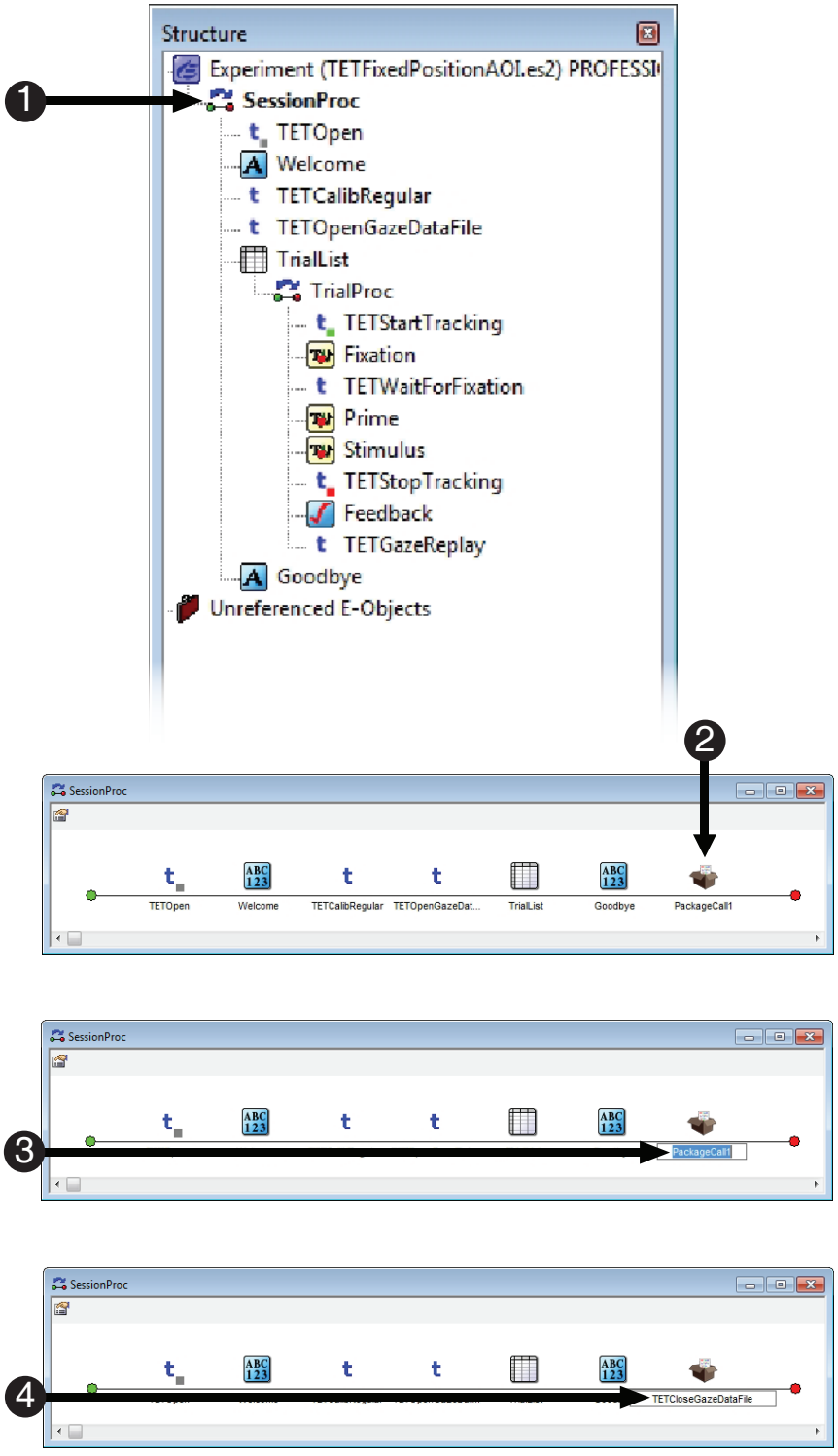
Now it is appropriate to add the TETCloseGazeDataFile PackageCall because we no longer need to save .gazedata during this experiment.

1) ***Double click*** the **SessionProc** Object to open it in the workspace.

2) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Goodbye** Object in the **SessionProc** procedure. The object will be given a default name of **PackageCall1**.

3) ***Click*** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.

4) ***Type*** "**TETCloseGazeDataFile**" as the new Object name and then ***press*** **Enter** to accept the change.



Task 15 (continued): Add the TETCloseGazeDataFile PackageCall to End Data Collection

Configure the TETCloseGazeDataFile PackageCall to call the CloseGazeDataFile routine of the TET package and accept the default properties.

Designate TET as the package and CloseGazeDataFile as the routine. Accept the default parameters.

5) ***Double click*** the **TETCloseGazeDataFile** PackageCall Object on the **SessionProc** to display its **Property Pages**.

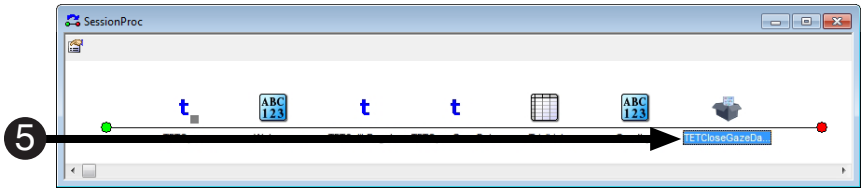
6) ***Select*** **TET** from the **Package** dropdown list.

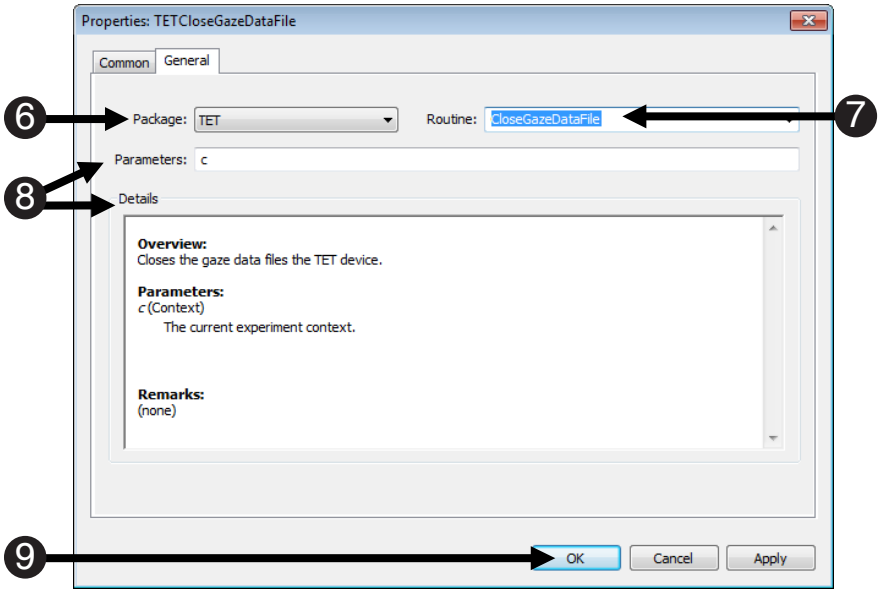
7) ***Select*** **CloseGazeDataFile** from the **Routine** dropdown list.

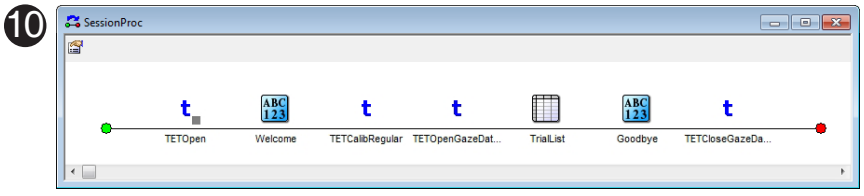
8) ***Review*** the **TETCloseGazeDataFile** parameters listed in the **Parameters** and **Description** fields.

9) ***Click*** the **OK** button to accept the changes and dismiss the **Property Pages**.

10) ***Confirm*** your **SessionProc** is identical to the example shown.







Task 16: Add the TETClose PackageCall to Close the TET Server

Add a PackageCall to the SessionProc and Name the PackageCall TETClose.

The last necessary PackageCall is **TETClose**. This call will close the TET package, and should be implemented once eye tracking is no longer necessary in your experiment.

1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **TETCloseGazeDataFile** PackageCall in the **SessionProc** procedure. The object will be given a default name of **PackageCall1**.

2) **Click** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.

3) ***Type*** "**TETClose**" as the new object name and then ***press*** **Enter** to accept the change.

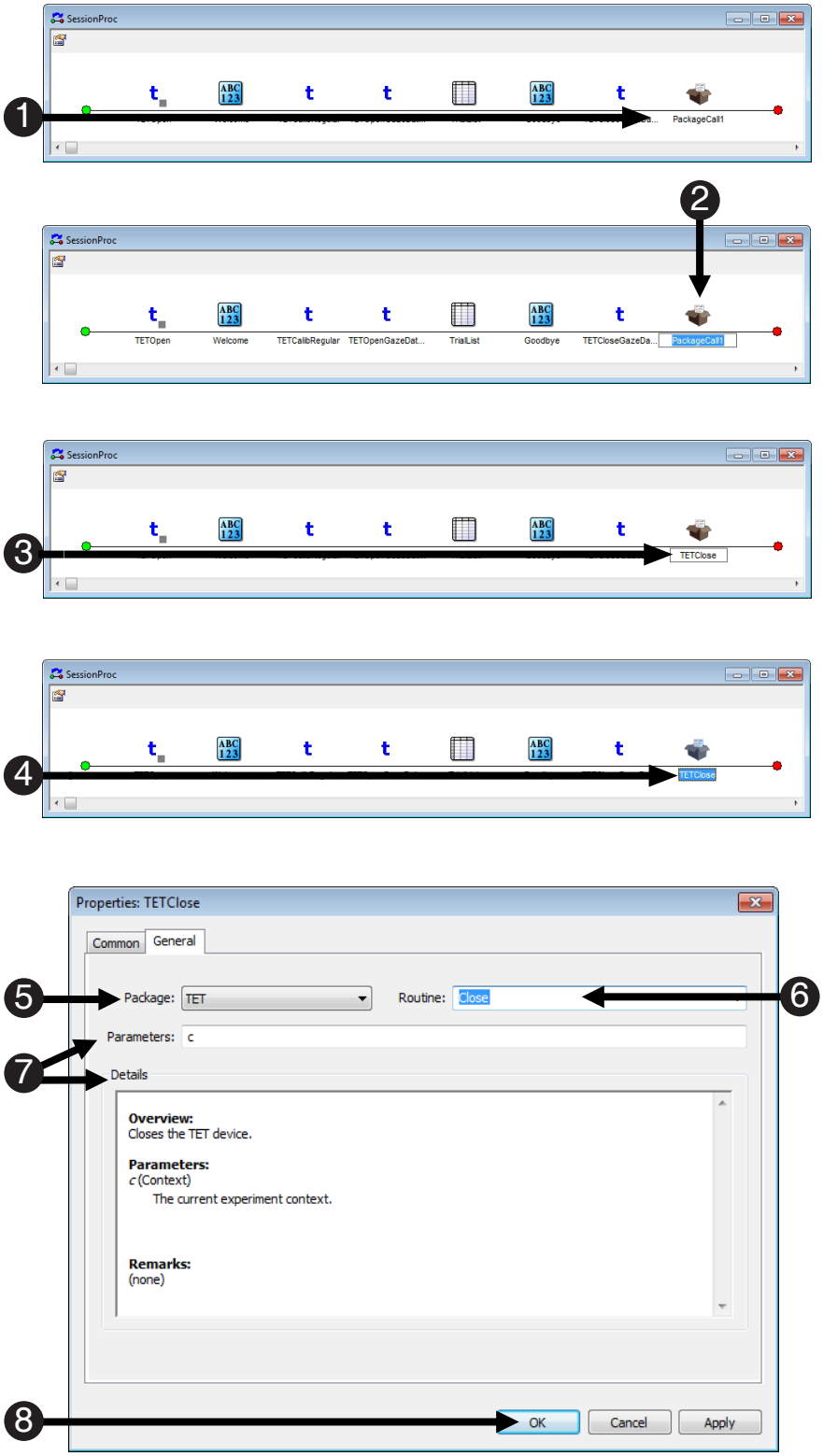
4) ***Double click*** the **TETClose** PackageCall Object on the **SessionProc** to display its **Property Pages**.

5) ***Select*** **TET** from the **Package** dropdown list.

6) ***Select*** **Close** from the **Routine** dropdown list.

7) ***Review*** the **TETClose** parameters listed in the **Parameters** and **Description** fields.

8) ***Click*** the **OK** button to accept the changes and dismiss the **Property Pages**.



Task 16 (continued): Add the TETClose PackageCall to Close the TET Server

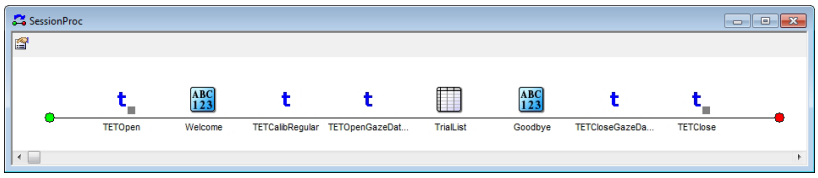
Configure the TETClose PackageCall to call the Close routine of the TET package and accept the default parameters.

Designate TET as the package and Close as the routine. Accept the default parameters.

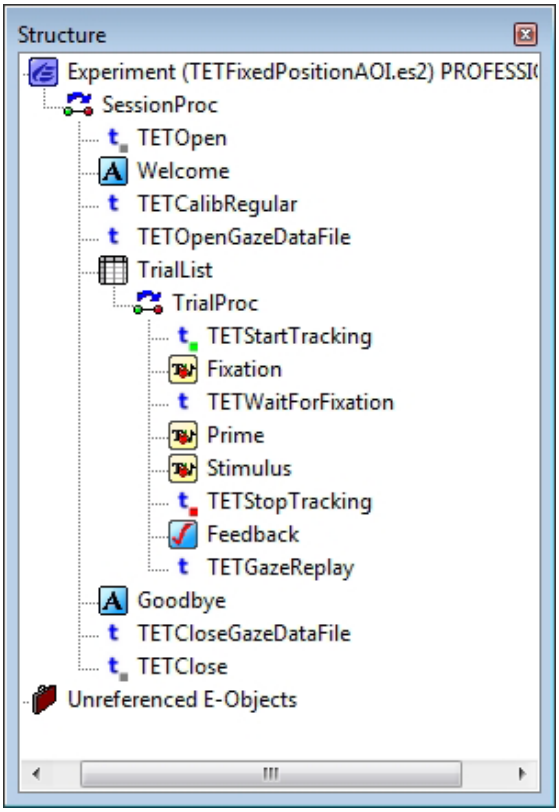
9) ***Confirm*** your **SessionProc** is identical to the example shown.

10) ***Compare*** the structure of the experiment you have created to the one on the right.

❾



❿



Task 17: Run the Experiment

Run the experiment to verify that the eye tracker is working and to ensure the .gazedata file is written.

You have now completed the basic steps necessary to create an E-Prime Extensions for Tobii -enabled paradigm. E-Prime Extensions for Tobii-enabled experiments can be run locally from E-Studio during development and testing. You should always fully test your experiment prior to scheduling actual participants or before using it to collect data.

Before you test the experimental task, it is important that you understand how the task works. You will need to have the sound on and speakers connected to the E-Prime computer to be able to perform the experiment because this experiment plays sound files. The task itself is simple; you will be presented with the name of an animal and simultaneously with a recording of the noise that animal makes. Next, you will see pictures of two animals on screen. One picture will be on the left and one picture will be on the right. If the animal on the right made the sound press the 2 key. If the animal on the left made the sound press the 1 key.

Task 17: Run the Experiment

Run the experiment to verify that the eye tracker is working, and to ensure the .gazedata file is written.

The next steps will walk you through generating the experimental script, starting the experimental paradigm, calibration, and running the experiment.

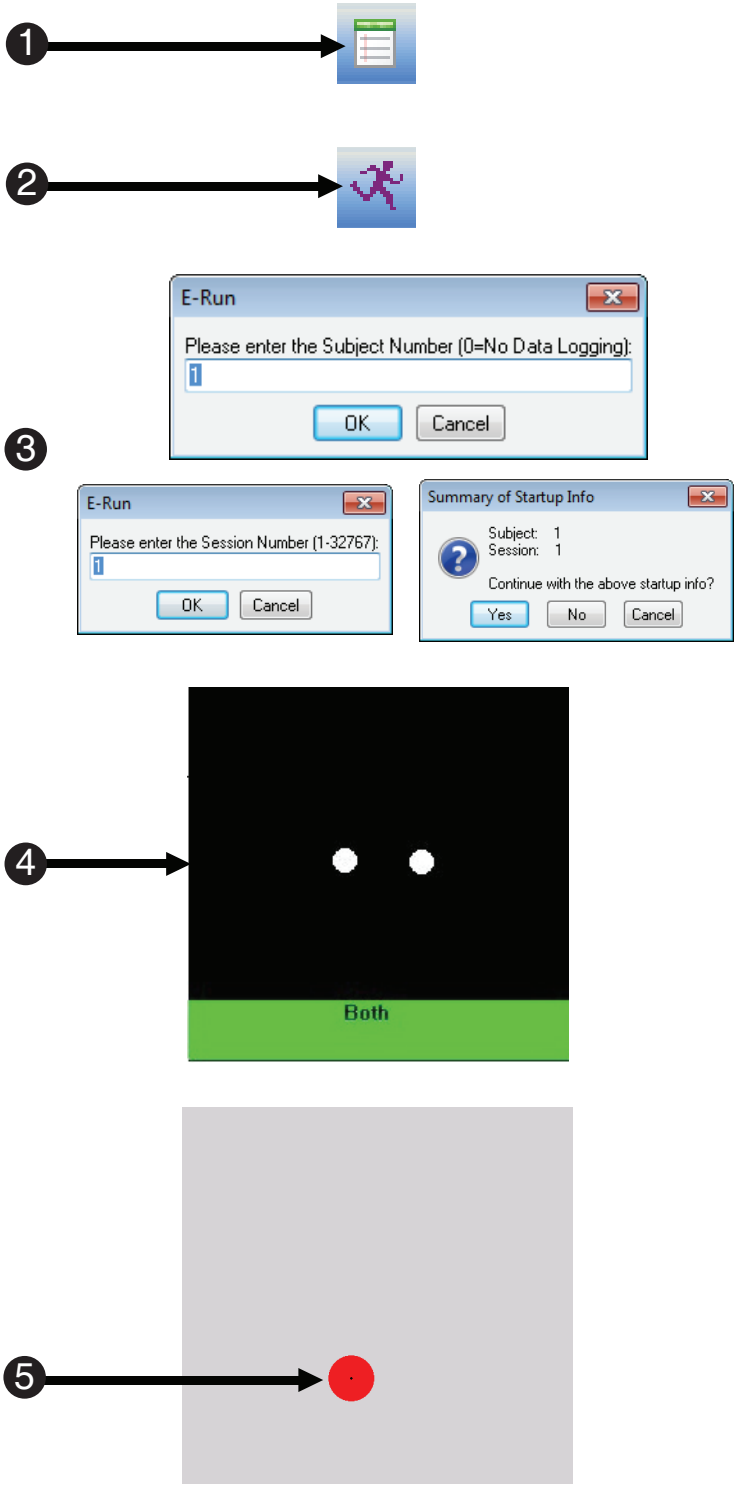
1) ***Press* Ctrl+S** to save your work before continuing. ***Click*** the **generate icon** or ***press* Ctrl+F7** to generate the script and ***check* it** for **errors**.

2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

3) ***Click* OK** to accept the **default values** for **Subject Number**, **Session Number** and **Summary of Startup Info**.

4) ***Look*** at the **screen** to ***verify*** the **eyes** are **stable** in the track status window. ***Ensure*** that there are **two white dots** that appear in the box and that the **bottom bar** is **green**. ***Press* Space** once both eyes are stable.

5) ***Run*** through the **calibration process** by ***following*** the **dots** on the **screen** with your **eyes**.



Task 17 (continued): Run the Experiment

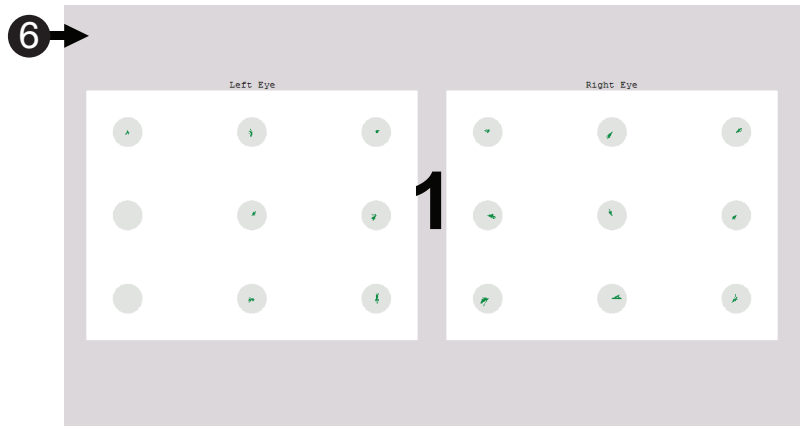
Run the experiment to verify that the eye tracker is working, and to ensure that the .gazedata file is written.

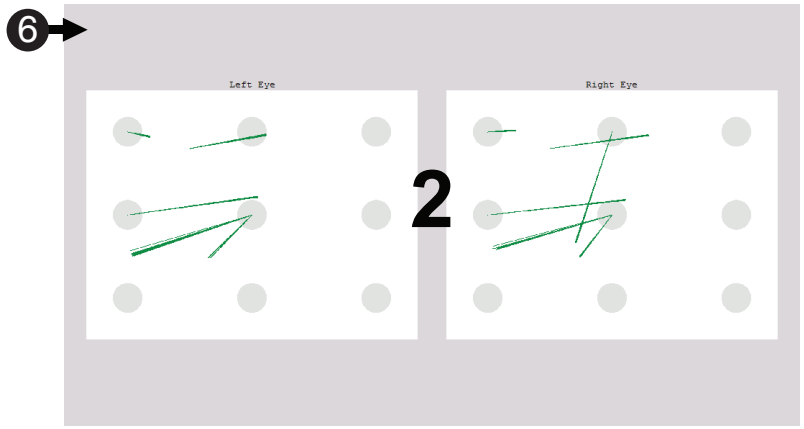
Once you have completed calibration, you will be prompted to verify that the calibration data is acceptable. Use the pictures below to help you determine the difference between a good and bad calibration. If a bad calibration is accepted, it will be difficult to collect valid data during the experiment because the Tobii Eye Tracker will have difficulty determining where the eyes are looking on the monitor.

6) **Picture 1** is a representation of **good** calibration data. Notice how the gaze data is within the circles. This indicates that the eye tracker will be able to collect usable data.

   Picture 2 is a representation of **bad** calibration data. Notice how the gaze data leaves the circles; this is undesirable. The eye tracker will have difficulty collecting usable data with a calibration like this.

7) ***Accept*** the **calibration** and ***perform*** the **experiment**.





Tutorial 2: **Writing to the Tab Delimited .gazedata File**

Summary:

Now that you have incorporated Tobii support into your E-Prime experiment, you will want to enable the experiment to write the data you collect to a file. E-Prime provides extensive capabilities for collecting, reporting, and viewing data. In most designs, E-Prime is used to collect and output one data row per trial. In eye tracking experiments, the goal is to collect and output data for every eye gaze sample (of which there may be several hundred per trial depending on the trial duration). Because of this, EET paradigms are enabled to employ an additional data collection and output technique which results in the creation of a tab delimited text file. This file contains data rows which are a combination of eye gaze data received by E-Prime from the Tobii Eye Tracker and any additional data added by the user via E-Prime.

The type and amount of data you choose to collect is determined by your experimental design and analysis choices. In order to allow maximum flexibility, the methods are performed in script sections of the experiment (as opposed to package calls) to expose both the content and the format of output files to manipulation by the user.

⚠ **NOTE:** *The typical E-Prime data file (.edat2) generated by your experiment is always available to you (unless you take steps to specify otherwise). For most eye tracking studies, the output file described in this section will be the data most used in your data analysis (as opposed to the .edat2 file). It is up to the experimenter to specify the column ordering of this output file and to verify that the correct data is available for the analysis which the experimenter expects to perform on the data.*

⚠ **NOTE:** *What data should you collect in your eye gaze data file? As you will see in this tutorial, the starting design material provided includes the typical non-paradigm specific data you will likely wish to collect. It does not, however, include paradigm specific data which might be critical to your study. This tutorial will show you how to add this additional data. (We recommend a conservative approach to data collection: "unless you are sure you won't need it, collect it". While it is easy to filter or ignore data in your analysis, you cannot retrieve data which you did not collect to begin with!)*

During this tutorial, you will add script to the experiment to manipulate the format of the output file. Additionally, we will discuss the HitTest Method. This method will be used to discern the Areas of Interest (AOI) we are tracking. AOIs are defined areas on the screen which represent critical objects or regions. AOIs are typically used to assist in analysis or in active runtime processing in E-Prime.
We will use the TETFixedPositionAOI.es2 experiment we previously created in the last tutorial.

⚠ **NOTE:** *If you do not want to overwrite your previous work, save the experiment under a different name before you begin this tutorial.*

Tutorial 2: **Writing to the Tab Delimited .gazedata File**

Summary:

If you have not completed **Tutorial 1: Adding Tobii Support to an E-Prime Experiment, (Page 15)** start by opening the experimental paradigm "…My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOI\TETFixedPositionAOI1.es2" Before you start, save the experiment under the name "…My Experiments\Tobii\Tutorials\TET\FixedPositionAOI\TETFixedPositionAOI.es2" in the same folder. This will overwrite any experiment with the same name. If you do not wish to overwrite, choose another name for the experiment that doesn't exist in this folder. Detailed instructions on how to save are found in **Tutorial 1, Task 2: Save the Experiment Under a New Name, (Page 17)**.

Resave:

Only follow the steps below if you are loading the TETFixedPositionAOI.es2 sample.

1) ***Select*** the **File > Save As…** from the application menu bar.
2) ***Type*** "**TETFixedPositionAOI.es2**" as the new name in the File name field.
3) ***Click*** the **Save** button.

Goal:

This tutorial illustrates how to write data to a tab delimited file. Once you complete this tutorial, you will be able to save the gaze data from your experiment.

Overview of Tasks:

- Open the TETFixedPositionAOI.es2.
- Declare a Global User Defined Data Type and name it UserEyeGazeDataType.
- Add a new member to the end of the UserEyeGazeData Type data structure.
- Edit the script to append a tab character and new column label.
- Append an ebTab line for the UserEyeGazeDataType.
- Create the SaveGazeData InLine.
- Modify the SaveGazeData InLine to use the HitTest method to track AOIs.
- Verify the overall experiment structure and run the experiment

Estimated Time: 20-30 minutes

Task 1: Copy the TETFixedPositionAOIUserScript.txt into E-Prime User Script

Open the TETFixedPositionAOIUserScript.txt file, copy the contents and paste them into the User Script tab in E-Studio.

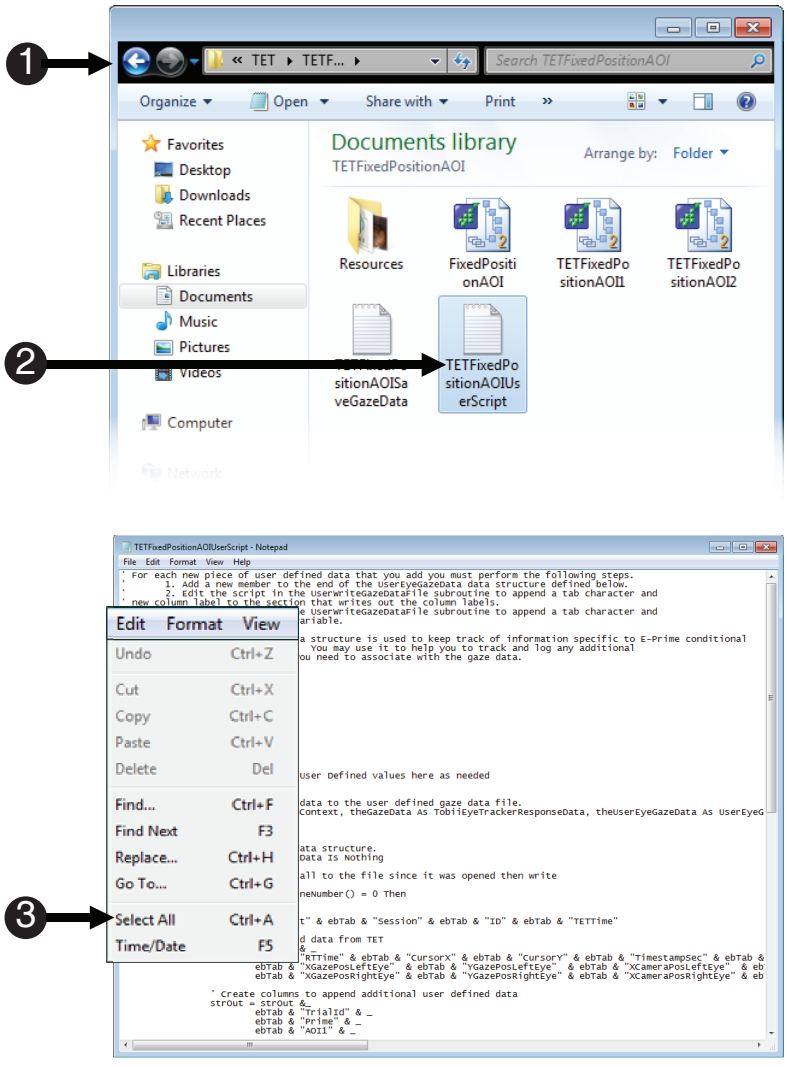
The UserEyeGazeData Type data structure is used to keep track of information specific to E-Prime conditional data per eye gaze observation. You may use it to help you to track and log any additional experiment related data that you need to associate with the gaze data. The quick and efficient way to get started is to copy the entire contents of the TETFixedPositionAOIUserScript.txt included in the "…My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOI\" folder and paste it into the User tab of the Script view. If you do not have the TETFixedPositionAOI1.es2 open, navigate to the "…My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOI folder, and double click the TETFixedPositionAOI1.es2.

⚠ The script provided in the TETFixedPositionAOIUserScript.txt and the TETFixedPositionAOISaveGazeData.txt (**Tutorial 2, Task 7: Copy Script from TETFixedPositionAOISaveGazeData.txt to SaveGazeData InLine, to Tutorial 2, Task 8: Edit the SaveGazeData InLine, Page 54-56**) files can be copied to every experiment that you create. It was written to be exported easily and contains commonly used variables. For each new experiment, you will need to alter some of the variable equivalencies to be specific to that particular experiment.

1) ***Navigate*** to "**…My Experiments\ Tobii\Tutorials\TET\ TETFixedPositionAOI**"

2) ***Double click*** the **TETFixedPositionAOIUserScript.txt** file to open the file.

3) ***Select*** **Edit > Select All (Ctrl+A)** to highlight the contents of the file.



Task 1 (continued): Copy the TETFixedPositionAOIUserScript.txt into E-Prime User Script

Open the TETFixedPositionAOIUserScript.txt file, copy the contents and paste them into the User Script tab in E-Studio.

The UserEyeGazeData Type data structure is used to store that part of the data which is collected or calculated by E-Prime. It is later combined with the eye gaze data collected from the eye tracker to create a single data row per eye gaze acquisition. Therefore, the UserEyeGazeData Type must contain any experimental data from E-Prime that is to be associated with eye gaze data in analysis.

4) ***Select*** **Edit > Copy (Ctrl+C)** to ***copy*** the contents of the file.

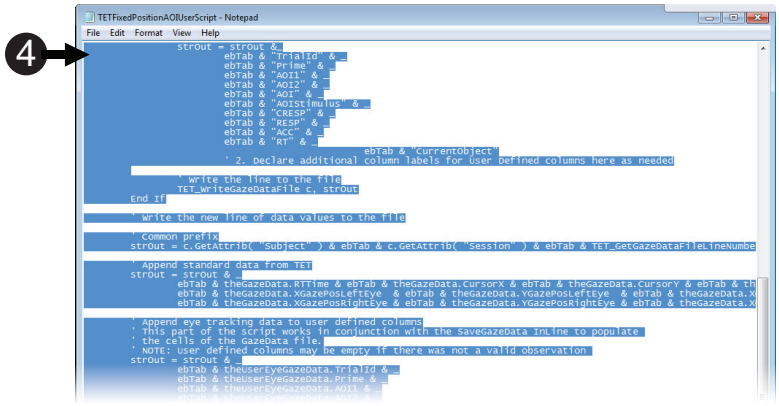
5) ***Click*** the **Script** view (Alt+5 to open) in **E-Studio.**

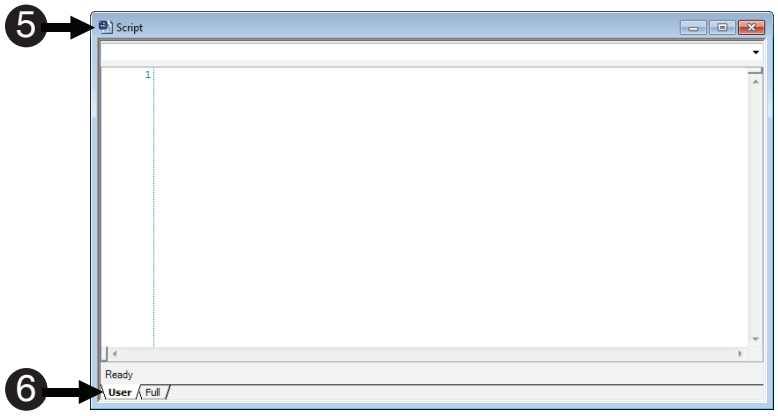
6) ***Select*** the **User tab.**

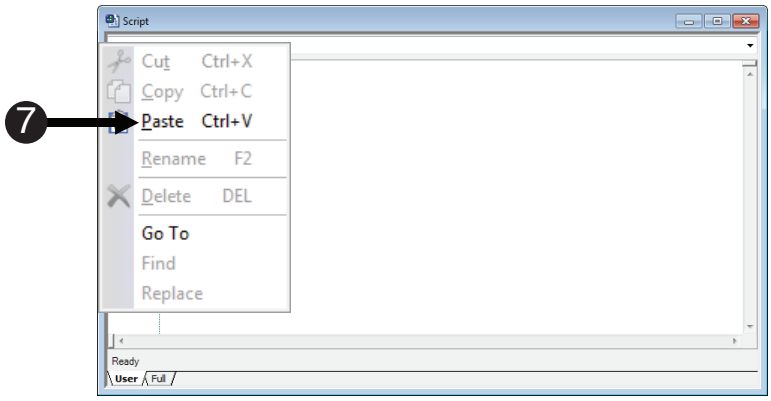
7) ***Select*** **Edit > Paste (Ctrl+V)** to ***paste*** the contents of the file into **E-Studio.**

8) ***Page up*** to the beginning of the object.

9) ***Press*** **Ctrl+S** to save your work before continuing.







Task 2: Understanding the UserEyeGazeData Type

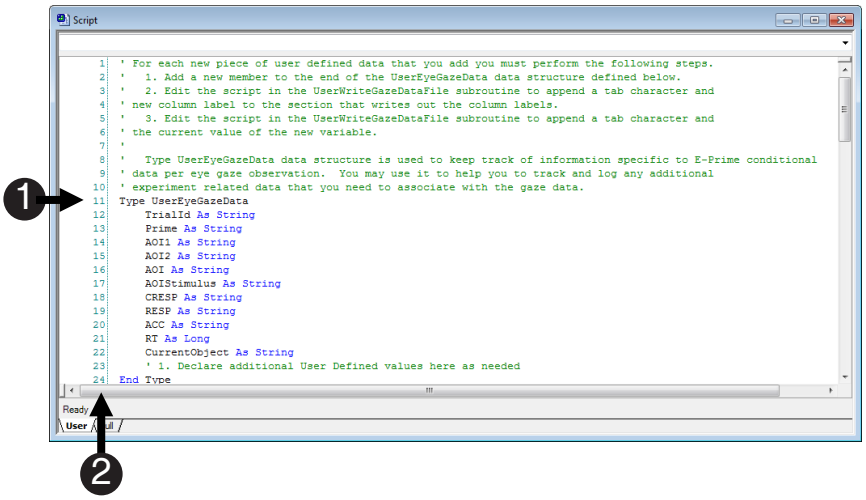
Select the User tab, and examine the TETFixedPositionAOI.es2 script.

The UserEyeGazeData Type is used to store any experimental data from E-Prime that is to be associated with the Tobii eye gaze data in analysis. E-Prime provides access to a wealth of experimental and contextual data. Anything related to experimental design, stimulus presentation, user responses, timing audit results, and user calculated data can be captured and contextualized with eye gaze data. Please refer to E-Prime's documentation for the full range of options. As an example for this tutorial, if your experiment is displaying an image on a colored background, you may wish to log the color of the background. You would then declare a parameter called BackColor of type String.

Take a moment to become familiar with the template which you have just pasted into the User Script. The variables provided represent typical data which would be collected in most studies. For readability, it is recommended that you make any additional User Defined values at the bottom of the structure. The order is not important as long as it is consistent ordering throughout the steps which follow. In some designs it may not be necessary to log the Prime, or the RT, etc. If you have determined that you do not need them in analysis, you may remove values that are not necessary to your data file (so long as you make sure to delete them in the next two steps).

1) ***Locate*** the **Type UserEyeGazeData**.

2) The numbers located to the left of the script window will tell you what line you are at in the script. You can use this to find specific places within the script. For example, Type UserEyeGazeData is located at line 11.



Task 3: Add a New Member to the End of the UserEyeGazeData Type

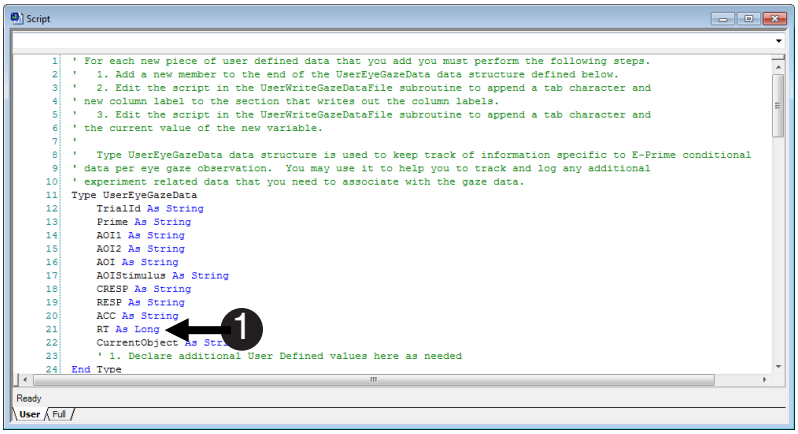
Edit the script at the end of the UserEyeGazeData Type and declare the variable BackColor as a string.

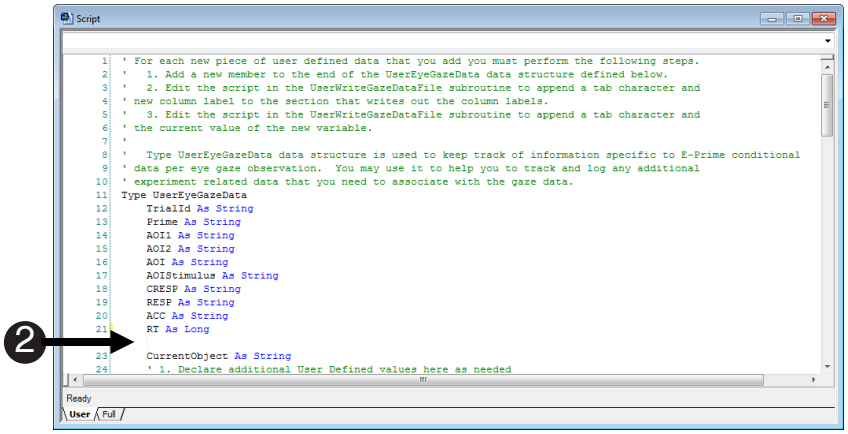
In this step, you will add the background color of the slide as a variable related to the context of the stimulus presentation. E-Prime will determine the value for this variable and will associate it with every eye gaze sample acquired while an instance of the slide was being presented.

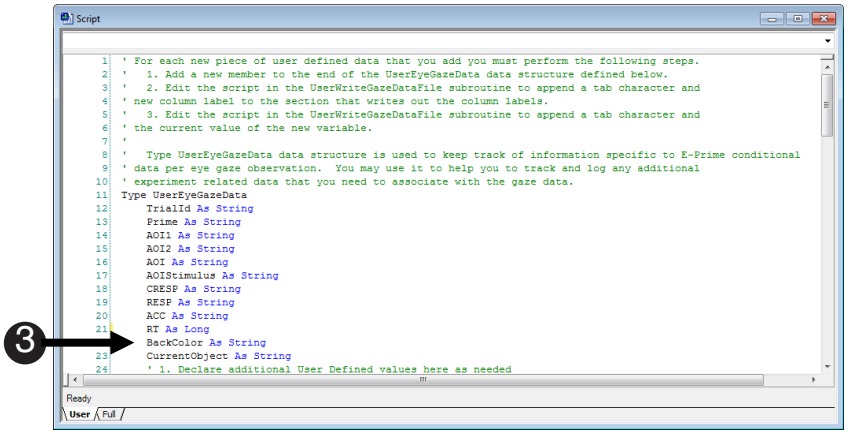
1) ***Position*** your cursor after the **g** on the line that reads "**RT As Long**."

2) ***Press*** **Enter.**

3) ***Type*** "**BackColor As String**"







Task 4: Edit User Script to Append a Column for the BackColor Variable

Edit the script labeled 'Create columns to append additional user defined data to label the BackColor column.

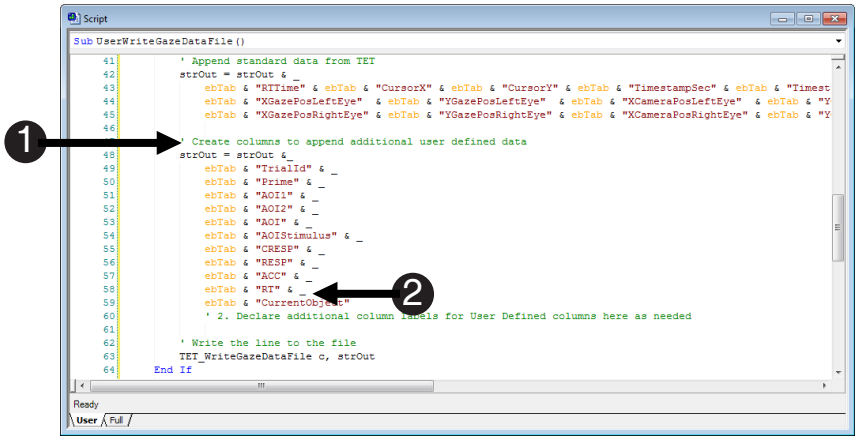
In order to add a column to the .gazedata file for the BackColor variable, you will need to add an ebTab line for each of the new data values you created in the previous step. This can be done in a section of the script labeled "Create columns to append additional user defined data." The ebTab command is a constant that is used to create columns in the data file by creating a tab. Using the proper syntax, you can list the name of the variables you want to create columns for. For readability purposes, keep the order consistent.

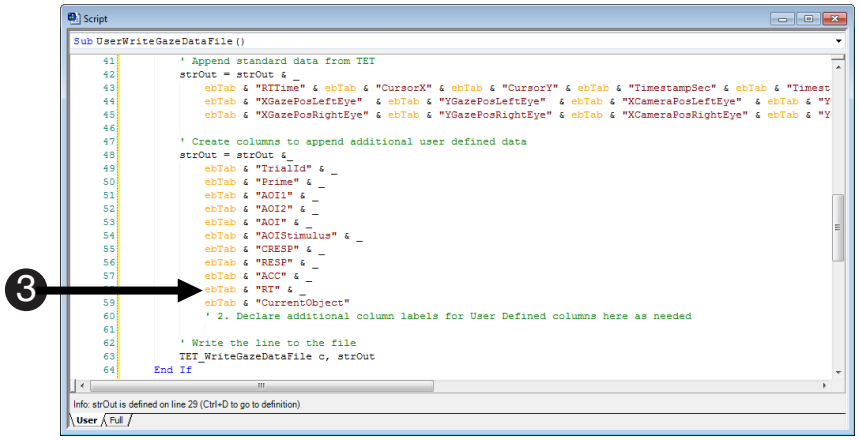
1) ***Locate*** 'Create columns to append additional user defined data in the ebTab group in the **User Script** (it will be about the middle of the script).

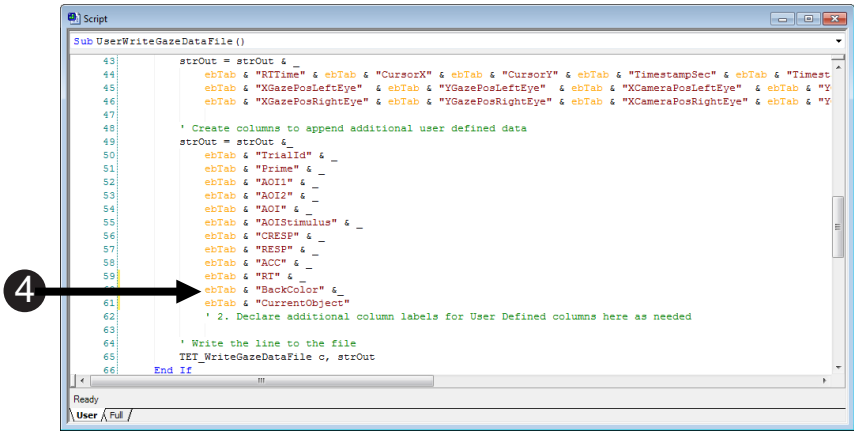
2) ***Position*** your cursor after the _ (underscore) on the line that reads **ebTab & "RT" & _**

3) ***Press*** **Enter**.

4) ***Type*** "**ebTab & "BackColor" & _**"







Task 5: Edit User Script to Append Value of BackColor

Edit the script labeled 'Append eye tracking data to user defined columns in the UserEyeGazeData Group to include the value of the BackColor variable to the data file.

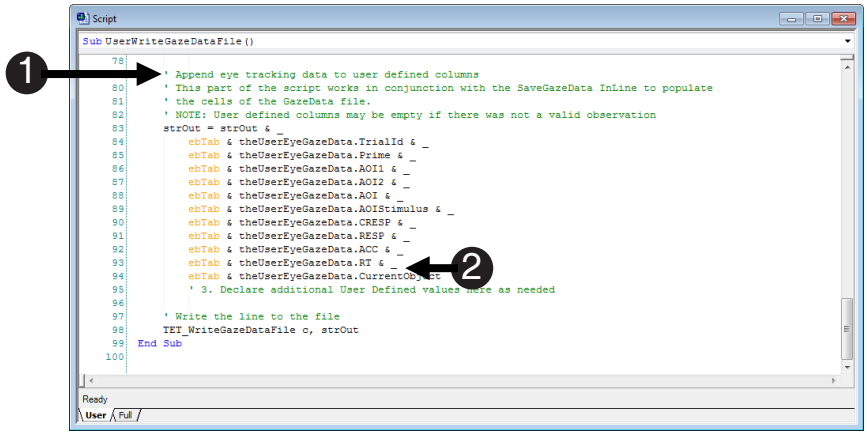
You will need to add an ebTab line in theUserEyeGazeData group for each of the new data values you created in the previous step. This will use the SaveGazeData InLine to populate the cells in the .gazedata file. The order of the variables is very important and needs to match the order of the columns from the previous step, as the ebTab command is only able to populate the cells in the order that it is given. If the order of the columns listed in the "Create columns to append additional user defined data" section and the order of the variables in the "Append eye tracking data to user defined columns" section do not correspond, then the data will not line up properly; the cells will be populated with data intended for another cell.

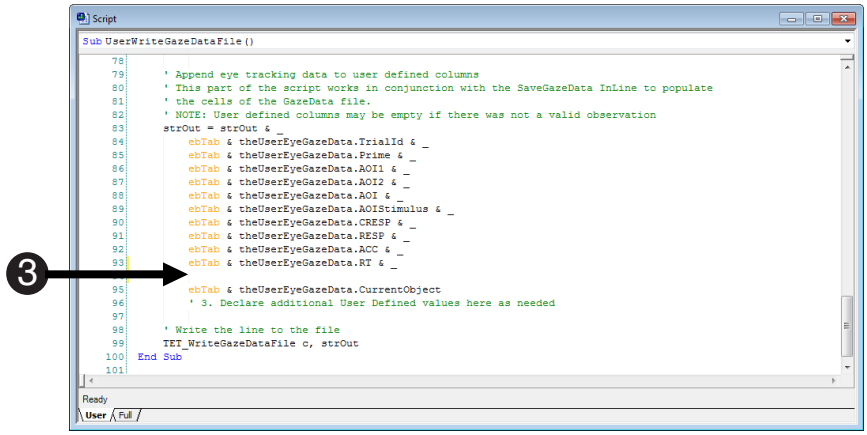
1) ***Locate*** 'Append eye tracking data to user defined columns near the end of the **User Script** (it will be before the TET_WriteGazeDataFile line).

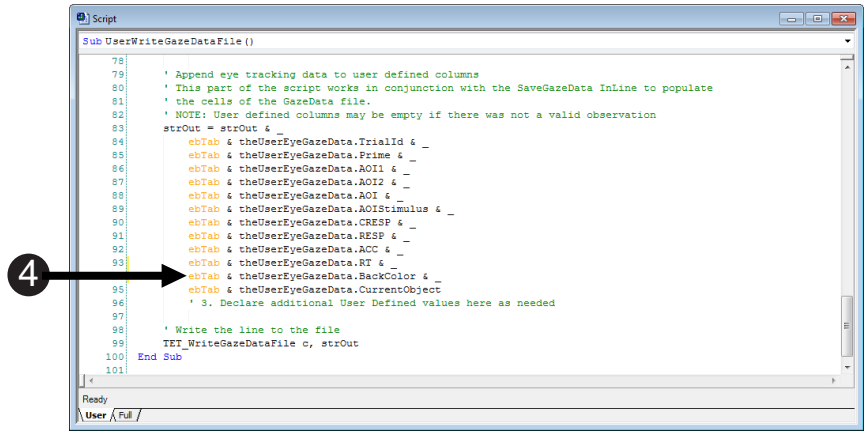
2) ***Position*** your cursor after the **_** (underscore) on the line that reads: **eb Tab & theUserEyeGazeData.RT & _**

3) ***Press*** **Enter.**

4) ***Type***: **ebTab & theUserEyeGazeData.BackColor & _**







Task 6: Add an InLine Object to the TrialProc

Add an InLine Object to the TrialProc and rename it to SaveGazeData.

The SaveGazeData InLine Object is used to collect the eye tracking data and works in conjunction with the User Script to write the data to a delimited .gazedata file. The .gazedata file can be viewed via Excel, or a similar spreadsheet application. The first step in creating the SaveGazeData InLine is to add an InLine Object to the experiment after you have stopped collecting eye tracking data. In this experiment, the InLine should follow the TETStopTracking PackageCall.

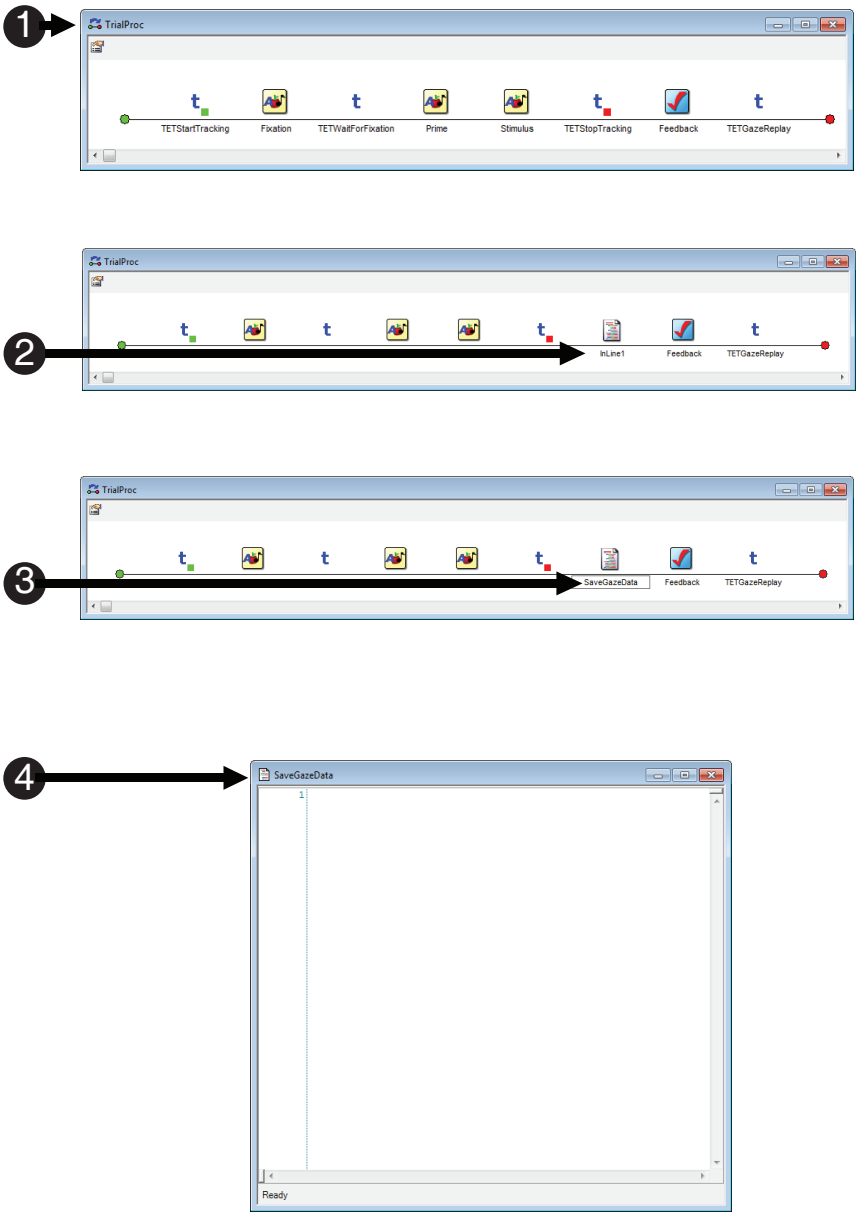
⚠ **NOTE:** *The script provided in the TETFixedPositionAOIUserScript.txt (from previous Task 2) and the TETFixedPositionAOISaveGazeData.txt files can be copied to every experiment that you create. It was written to be exported easily and contains commonly used variables. For each new experiment, you will need to alter some of the variable equivalencies to be specific to that particular experiment.*

1) ***Double click*** the **TrialProc** to open it in the workspace.

2) ***Drag*** a new **InLine** Object from the E-Prime Toolbox and **drop** it ***after*** the **TETStopTracking** PackageCall. The **InLine** will be given a default name of **InLine1**.

3) ***Click*** the **InLine1** Object, ***press*** **F2** to ***rename*** the Object to **SaveGazeData**. ***Press*** **Enter** to accept the change.

4) ***Double click*** the **SaveGazeData InLine** to open it in the workspace.



Task 7: Copy Script from TETFixedPositionAOISaveGazeData.txt to SaveGazeData InLine

Open the TETFixedPositionAOISaveGazeData.txt file, copy the contents, and paste them into the SaveGazeData InLine in E-Studio.

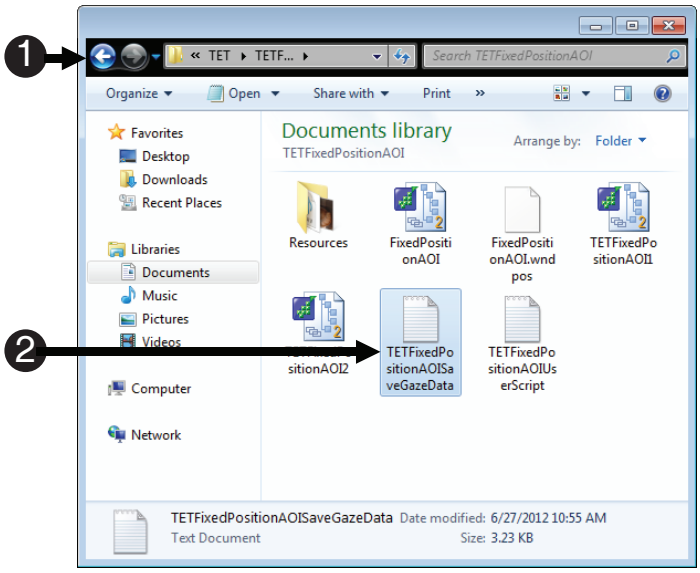
Creating the .gazedata file is a complicated step, and will vary greatly from experiment to experiment. The quick and efficient way to get started is to copy the entire contents of the TETFixedPositionAOISaveGazeData.txt included in the "…My Experiments\Tobii\Samples\Tutorials\TET\TETFixedPositionAOI\" and paste it into the User tab of the Script view in E-Studio.

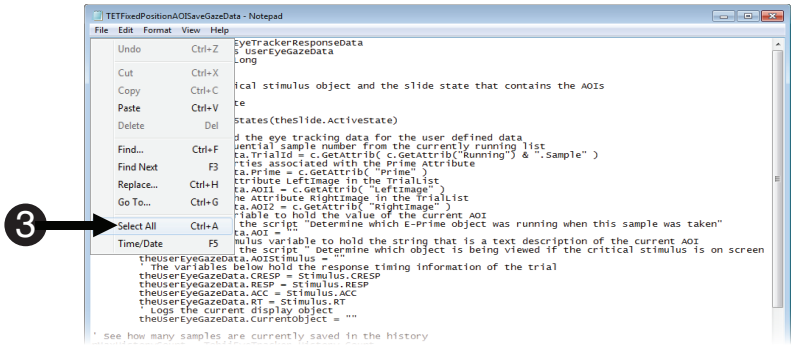
1) ***Navigate*** to "…My Experiments\Tobii\Samples\Tutorials\TET\TETFixedPositionAOI\."

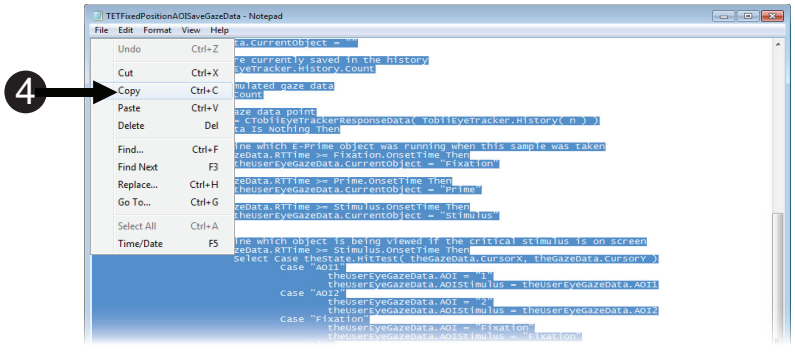
2) ***Double click*** the **TETFixedPositionAOISaveGazeData.txt** file to open the file.

3) ***Select*** **Edit > Select All (Ctrl+A)** to highlight the contents of the file.

4) **Select Edit > Copy (Ctrl+C)** to ***copy*** the contents of the file.







Task 7 (continued): Copy Script from TETFixedPositionAOI SaveGazeData.txt to SaveGazeData InLine

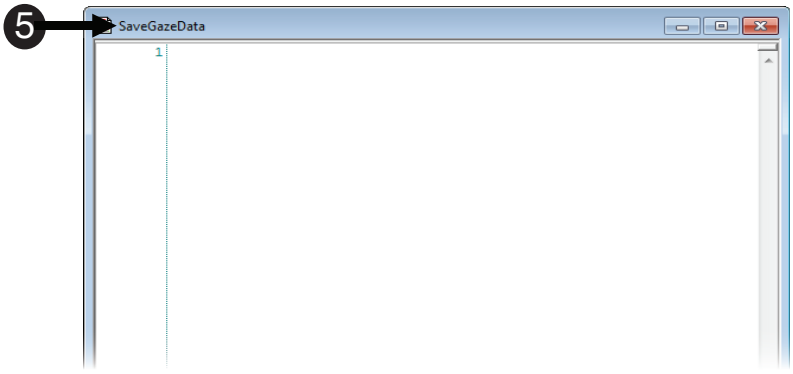
Open the TETFixedPositionAOISaveGazeData.txt file, copy the contents, and paste them into the SaveGazeData InLine in E-Studio.

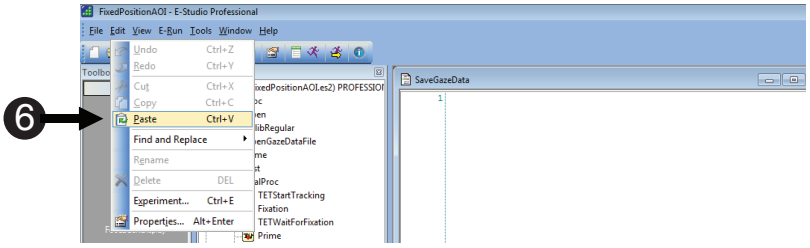
5) ***Click*** the **SaveGazeData InLine** in **E-Studio.**

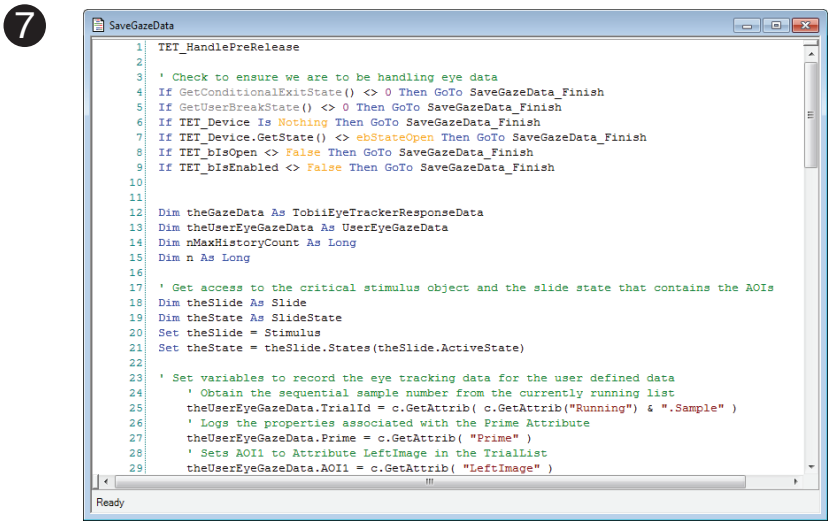
6) ***Select*** **Edit** > **Paste (Ctrl+V)** to ***paste*** the contents of the file into **E-Studio.**

7) ***Page up*** to the beginning of the file.

8) ***Press*** **Ctrl+S** to save your work before continuing.







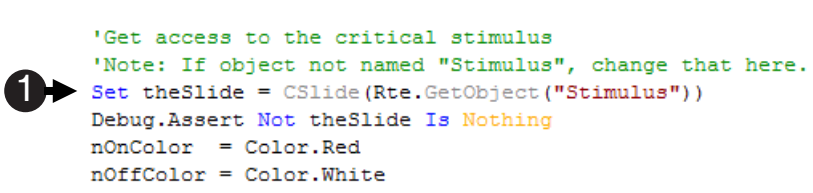
Task 8: Edit the SaveGazeData InLine

Open the TETFixedPositionAOISaveGazeData.txt file, copy the contents and paste them into the SaveGazeData InLine in E-Studio.

Next we will modify the SaveGazeData InLine. The script you copied in the previous steps was not paradigm specific, so that it might be imported into and adapted for other experiments. For each new experiment you will need to check that the variables are set correctly. The first variable we need to check is the Slide variable. This variable needs to correspond to the object that displays your critical stimulus. The critical stimulus is the object you are manipulating to measure the dependent variable of your hypothesis. In this experiment, the critical stimulus is the object that displays the images to the left and right visual fields, Stimulus. In addition to ensuring the Slide Object is associated with the correct object, you must add a line to account for any additional properties you defined in the User Script, e.g. BackColor, and set a default value for the additional properties. If you do not wish to set a default value, simply use empty quotation marks to indicate an empty string value.

1) ***Confirm*** **Set theSlide = CSlide (Rte.GetObject ("Stimulus"))**

2) ***Press*** **Ctrl+S** to save your work before continuing.



Task 9: Understand theUserEyeGazeData Type Variable Equivalents

Understand how the variables in theUserEyeGazeData type correspond to other objects in the experiment.

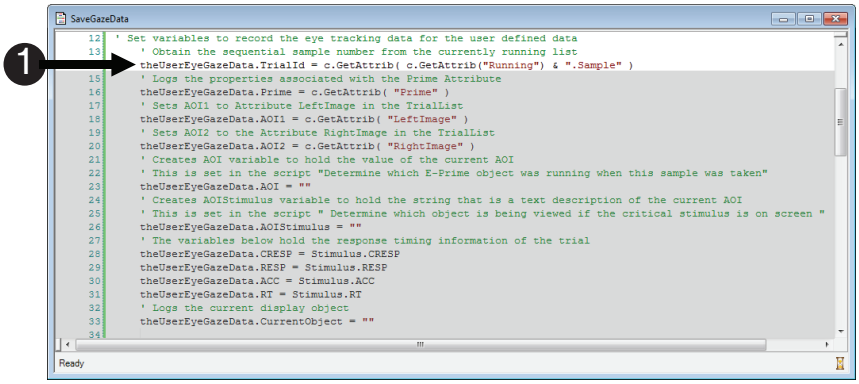
Below the declaration made in the previous task is the script that assigns values to the columns that will be output into the .gazedata file. The variables are used to assign a unique ID to the trial and collect other information about the trial. AOI refers to an Area of Interest (see **Tutorial 2, Task 13: Run the Experiment, (Page 63)**). Look over the variable descriptions below to get a basic understanding of how they function in the script. This will differ from experiment to experiment depending on the variable names and the data that is being saved in the .gazedata file.

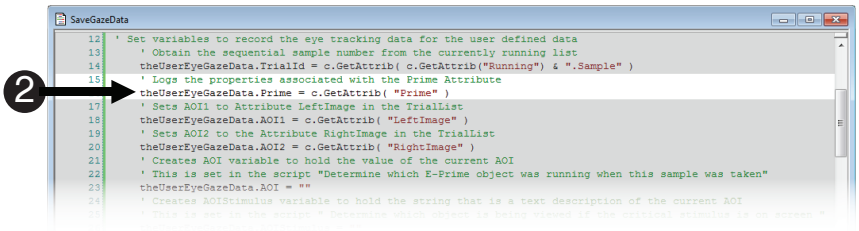
1) **theUserEyeGazeData.TrialId**
   Obtain the sequential sample number from the currently running list.

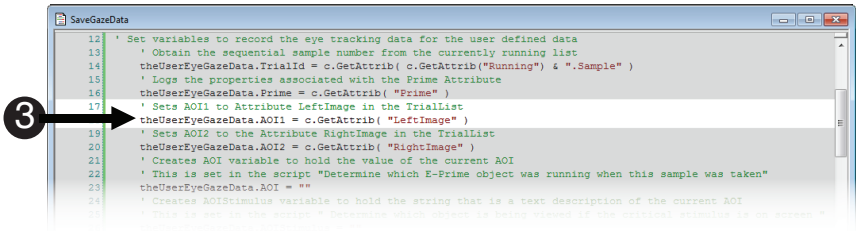
2) **theUserEyeGazeData.Prime**
   Logs the properties associated with the Prime Attribute.

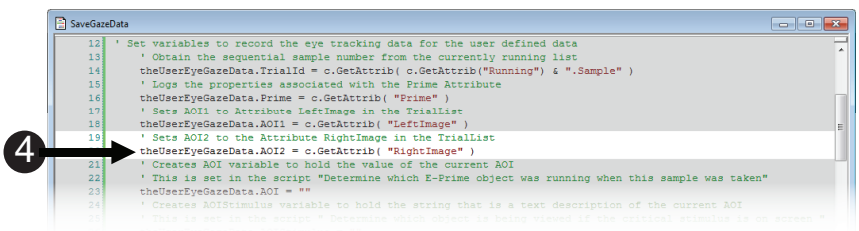
3) **theUserEyeGazeData.AOI1**
   Sets AOI1 to Attribute LeftImage in the TrialList.

4) **theUserEyeGazeData.AOI2**
   Sets AOI2 to the Attribute RightImage in the TrialList.









Task 9: (continued) Understand theUserEyeGazeData Type Variable Equivalents

Understand how the variables in theUserEyeGazeData type correspond to other objects in the experiment.

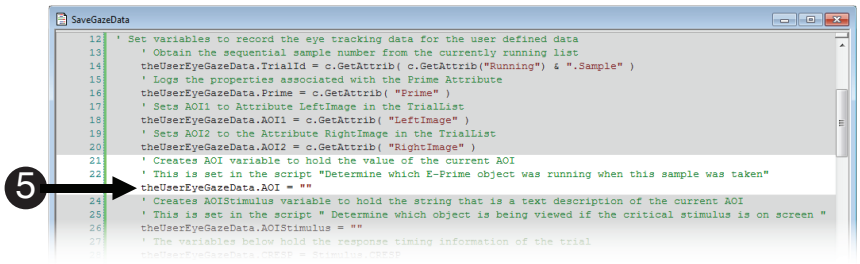
The variables below add information that will be used in the data analysis to the .gazedata file. The AOI and AOIStimulus variables will identify where the participant was looking during a given trial. The BackColor variable will need to be added to the list for its information to be added to the .gazedata file.

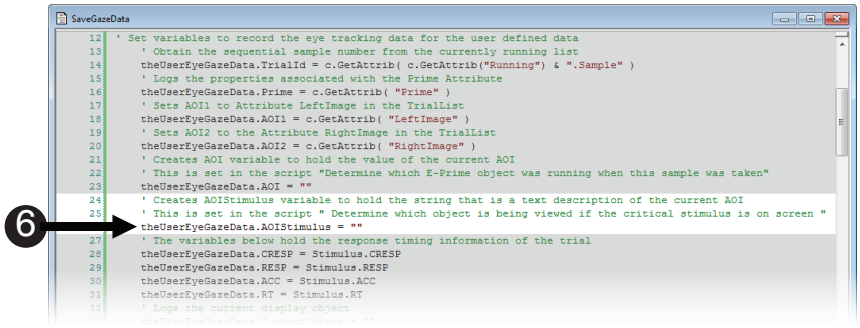
5) **theUserEyeGazeData.AOI**
Creates AOI variable to hold the value of the current AOI. This is set in the script “Determine which E-Prime object was running when this sample was taken”

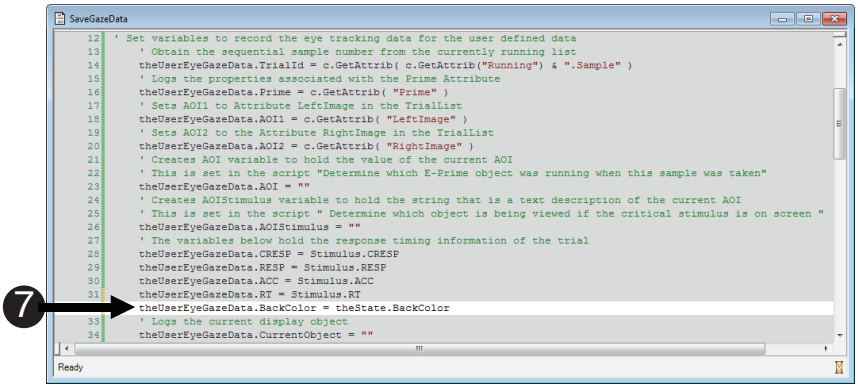
6) **theUserEyeGazeData.AOIStimulus**
Creates AOIStimulus variable to hold the string that is a text description of the current AOI. This is set in the script “Determine which E-Prime object was running when this sample was taken”

7) ***Add* theUserEyeGazeData.BackColor = theState.BackColor**. before continuing.

8) ***Press* Ctrl+S** to save your work before continuing.







Details on how the SaveGazeData InLine Saves Data

⚠ **NOTE:** *The methods employed in the SaveGazeData InLine are not designed to be executed when E-Prime is performing tasks related to data collection and stimulus presentation. In most experimental designs this will be performed between trials. In this tutorial, samples from the eye tracker are stored in memory until the end of the trial, at which point this script is executed.*

Once the trial is over, the SaveGazeData InLine will process all of the eye gaze data which has been collected since its last execution. The SaveGazeData InLine will then perform any calculations required to fill the variables pertaining to the current trial. This is accomplished in the script sample in Task 12. This loop will run through each of the acquired gaze data points and log the appropriate E-Prime related data.

Task 10: Log the Objects on Screen Via .OnsetTime

Use the .OnsetTime property to log the time at which the object was placed on the screen.

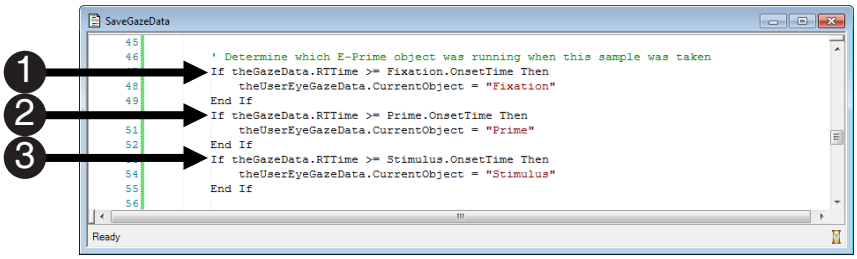
Below is the script that controls how the .gazedata is associated with the objects presented on screen. The .RTTime property of theGazeData will give you a timestamp of when the current gaze data point was captured by E-Prime. By using conditional statements, you can compare that timestamp to the .OnsetTime properties of the objects in your procedure. In the following example, theUserEyeGazeData.CurrentObject was used to log which object was on the screen when a particular gaze data point was acquired.

1) ***Checks*** if the **current Object** on screen is the **Fixation Object**.

2) ***Checks*** if the **current Object** on screen is the **Prime Object**.

3) ***Checks*** if the **current Object** on screen is the **Stimulus Object**.

4) ***Press Ctrl*+S** to save your work before continuing.



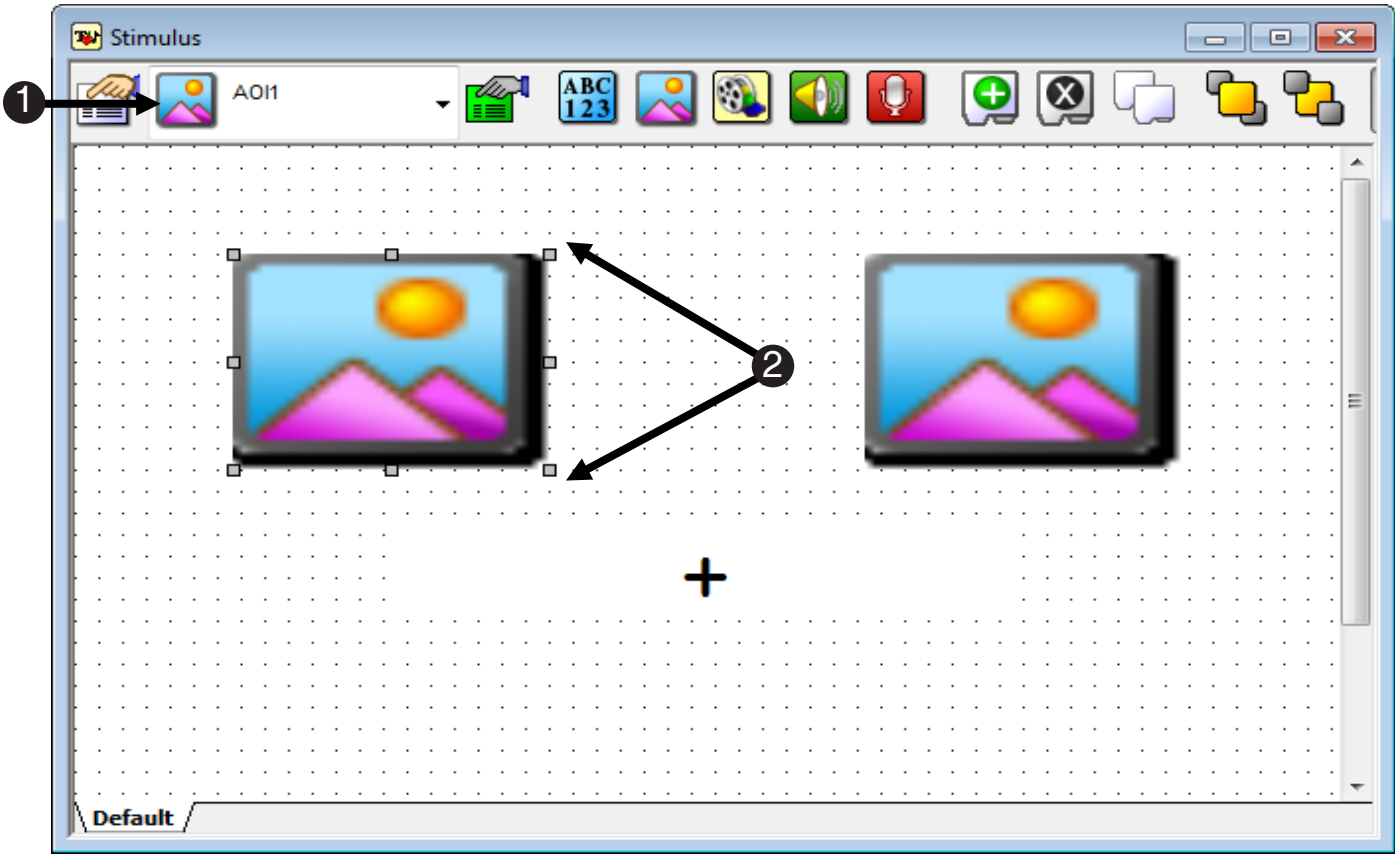
Task 11: Declare AOIs as Slide sub-objects

Associate Area Of Interest (AOI) Objects with eye gaze data.

Recall that we previously defined an Areas of Interest (AOIs) as areas on the screen which represent critical objects or regions. AOIs are typically used to assist in analysis or in active runtime processing in E-Prime. Now we need to populate the AOI columns we declared in the User script with data. In **Tutorial 2, Task 8: Edit the SaveGazeData InLine, (Page 56)** we defined the critical stimulus as the Stimulus Object. The Stimulus Object contains three sub-objects; Fixation, AOI1, and AOI2. We have already associated the AOI1 label to the images presented on the left side of the screen and the AOI2 label to the images presented on the right side of the screen in the script when the experiment was originally created.

1) ***Select* AOI1**.

2) When AOI1 is selected from the dropdown list, the image associated with the name, AOI1, will be indicated with gray boxes.



Task 12: The HitTest Method

Use the HitTest Method to determine which Sub-Object a given .gazedata point is within.

The HitTest Method determines the on screen location of the cursor or mouse pointer on a Slide Object by using the X and Y coordinates. The .gazedata point can be recorded with this method by substituting the .CursorX and .CursorY. This will allow you to determine whether or not the current point was within the given Sub-Object by referencing the name of the Slide Sub-Object, e.g., AOI1, AOI2, Fixation. The "Else" case will log a null value for the AOI in the event that the given gaze data point is not within any of the AOIs. You will need to add to the script below to include any of the possible AOIs which you present in addition to removing the AOIs that do not exist in your particular experiment. For more info see HitTest in E-Basic Help.

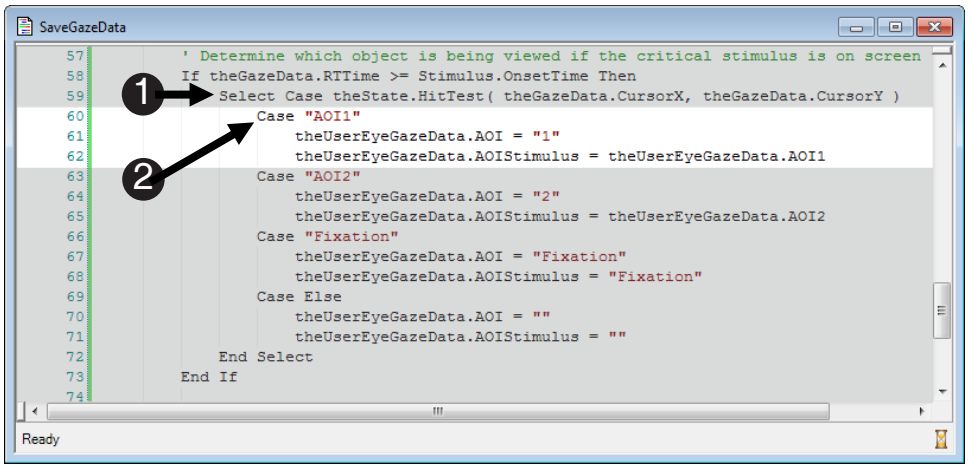
1) Command line that executes **HitTest**.

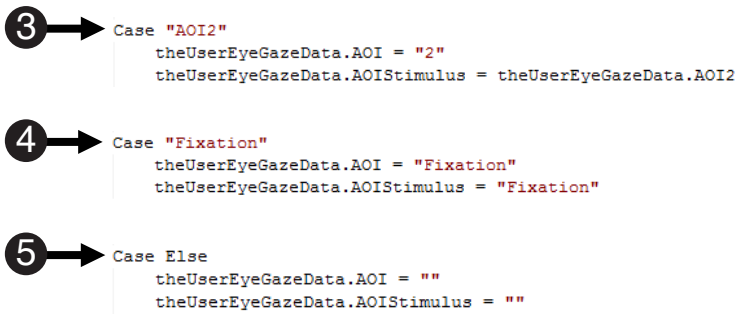
2) **Case "AOI1"**: ***Defines*** the **criteria** for the **1st AOI**. ***Assigns*** the **current AOI** to **1** and ***populates*** **AOIStimulus** with a **text description** of AOI1.

3) **Case "AOI2"**: ***Defines*** the **criteria** for the **2nd AOI**. ***Assigns*** the **current AOI** to **2** and ***populates*** **AOIStimulus** with a **text description** of AOI2.

4) **Case "Fixation"**: ***Defines*** the **criteria** for the **Fixation** condition. ***Assigns*** the **current AOI** to **Fixation** and ***populates*** **AOIStimulus** with the text **"Fixation"**.

5) **Case "Else"**: ***Defines*** the **criteria** for the **AOI null** condition. ***Assigns*** the **current AOI** to an **empty string** and ***populates*** **AOIStimulus** with an **empty string** (makes it blank.)





Task 13: Run the Experiment

Run the experiment to verify that the eye tracker is working and to ensure the .gazedata file is written.

The next steps will walk you through generating the experimental script, starting the experimental paradigm, calibration, and running the experiment.

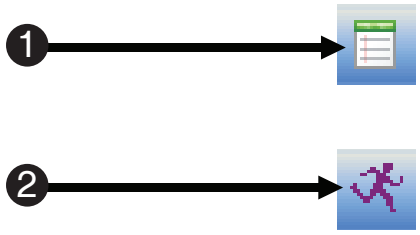
1) ***Press* Ctrl+S** to save your work before continuing. ***Click*** the **generate icon** or ***press* Ctrl+F7** to generate the script and ***check* it** for **errors**.

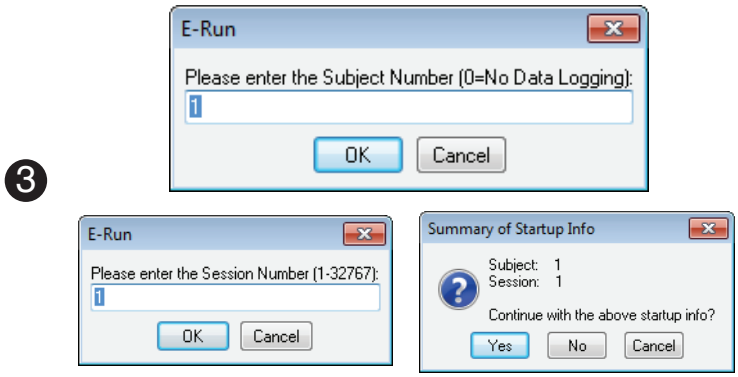
2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

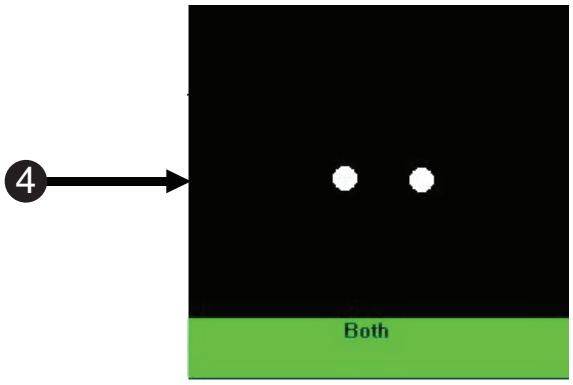
3) ***Click* OK** to accept the **default values** for **Subject Number**, **Session Number** and **Yes** for **Summary of Startup Info**.

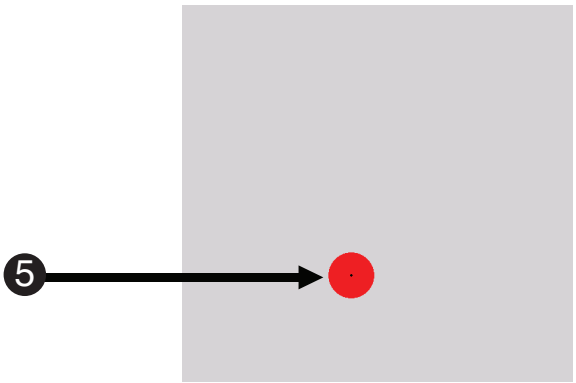
4) ***Look*** at the **screen** to ***verify*** the **eyes** are **stable** in the track status window. ***Ensure*** that there are **two white dots** that appear in the box and that the **bottom bar** is **green**. ***Press* Space** once both eyes are stable.

5) ***Run*** through the **calibration process** by ***following*** the **dots** on the **screen** with your **eyes**.









Task 13 (continued): Run the Experiment

Run the experiment to verify that the eye tracker is working, and to ensure the .gazedata file is written.

Once you have completed calibration, you will be prompted to verify that the data is acceptable.

6) ***Accept*** the **calibration**.

⚠ **NOTE:** *Please refer to* **Tutorial 1, Task 17 (continued): Run the Experiment, (Page 44)** *for additional information about calibration.*

Task 14: Open the Eye .gazedata File

Verify the .gazedata file exists and can be opened.

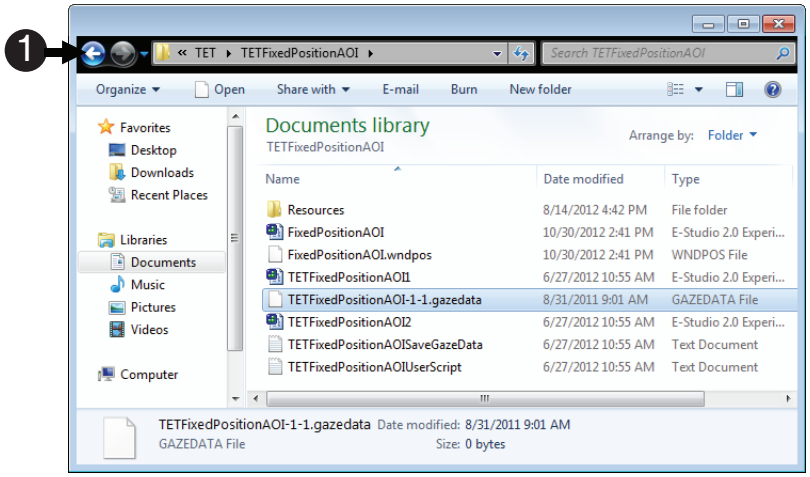
The .gazedata file contains both information about the stimulus presentation, e.g. what was on screen and when it was on screen, and the eye tracking data, e.g. where the eyes were looking. This file will facilitate data analysis. After the experiment is created, it is important that you check this file thoroughly to make sure all of the components you will need for your data analysis are included in the file before you begin running study participants. In this tutorial we will simply verify that the file exists and can be opened. In the next tutorial, **Tutorial 3: Introduction to the .gazedata File**, we will explain the gazedata output in detail. A filename will be created in the form of DataFile.BaseName. DataFile.BaseName defaults to [ExperimentName]-[Subject#]-[Session#]. In this example the file is named TETFixedPositionAOI-1-1.gazedata.

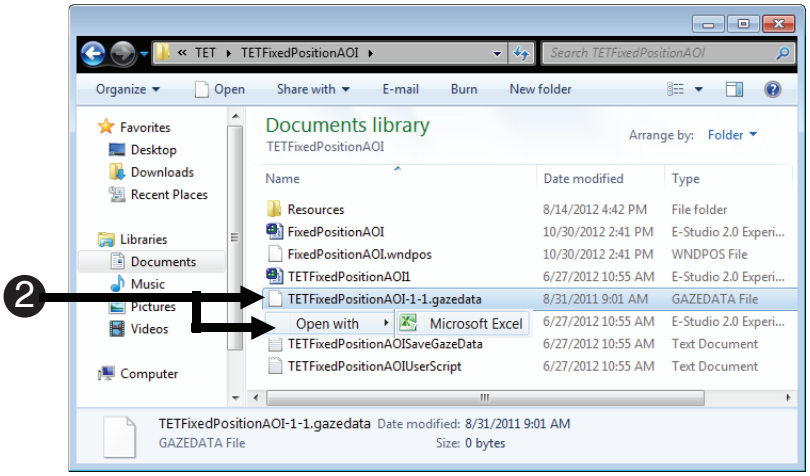
1) ***Navigate*** to **…My Experiments\ Tobii\Tutorials\TET\ TETFixedPositionAOI\ FixedPositionAOI-1-1.gazedata.**

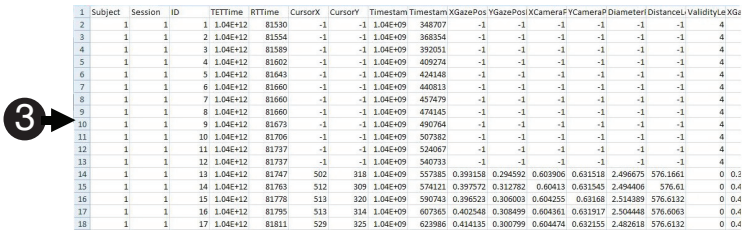
2) ***Right click*** the **TETFixedPosition AOI-1-1.gazedata file**. ***Select*** **Open With > Microsoft Office Excel**. If prompted, accept all defaults. ***Click*** **Finish**.

⚠ **NOTE:** *The file name will vary depending on what subject number and session number you chose to run the experiment under.*

3) ***Verify*** that the **file** opens and looks similar to the image to the right.







Tutorial 3: **Introduction to .gazedata File**

⚠ **NOTE:** *In order to get the full benefit of this tutorial, you will need to run through the TETFixedPositionAOI.es2 file from start to finish. Otherwise you will not have a data file to compare to the screen shots in the manual.*

If you have not completed Tutorial 2, start by opening the experimental paradigm "…My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOI\TETFixedPositionAOI2.es2." Before you start, save the experiment under the name "TETFixedPositionAOI.es2" in the same folder. Detailed instructions on how to save can be found in **Tutorial 1, Task 2: Save the Experiment Under a New Name., Page 67**.

Resave:
Only follow the steps below if you are loading the TETFixedPositionAOI.es2 sample.

1) ***Select*** the **File > Save As…** from the application menu bar.
2) ***Type*** "**TETFixedPositionAOI.es2**" as the new name in the File name field.
3) ***Click*** the **Save** button.

Summary:

In the previous tutorial you enabled TETFixedPositionAOI.es2 to write eye tracking and critical timing data collected during the experiment to a file. A tab delimited text file containing a combination of eye gaze data from the Tobii Eye Tracking device, as well as E-Prime critical timing data was created. The mechanisms that formatted this output file are the UserScript and SaveGazeData InLine. This tutorial offers an understanding of those mechanisms.

In this tutorial, you walk through the UserScript and SaveGazeData InLine to understand the role of important commands in the script. The .gazedata file is created and formatted within the script of the TETFixedPositionAOI.es2. The UserScript can be viewed as a way to format the structure of the output file. This script organizes the output file into columns, writes the column headers, and defines the structure of the output file. The SaveGazeData InLine collects the real-time eye tracking data in the buffer until the trial is complete and the data can be written to the .gazedata file. An understanding of the script's function enables you to output all necessary data for analysis and aids in troubleshooting, should problems arise during testing. Once you finish the tutorial, you will understand the relationship between the UserScript and the SaveGazeData InLine and how they work together to create the .gazedata file.

Tutorial 3: **Introduction to .gazedata File**

Goal:

This tutorial provides insight into the process used to create the .gazedata file, the way in which data is written to it, and an explanation of the script used to do this.

Overview of Tasks:

- Familiarize yourself with the basic structure of the .gazedata file.
- Open the .gazedata file for TETFixedPositionAOITracking to verify the data written to the file.
- Look at the specific sections of script from the SaveGazeData InLine and the TETFixedPositionAOI User Script side by side with the columns of the .gazedata file the script created.
- Learn how the .gazedata file is created.

Estimated Time: 10-15 minutes

Task 1: Introduction to .gazedata File

Understand the general output from the .gazedata file.

The .gazedata file contains information about eye movements from the Tobii Eye Tracker and information about the stimulus presentation from E-Prime. The data consists of general information regarding the eye gaze data that is continuously collected as well as user-defined data. This data is compiled in one file to make analysis easier. The table below summarizes the general information included in the .gazedata file. User-defined data will be discussed later in this tutorial. Take some time to familiarize yourself with the basic output and the information that it provides. This way you will be able to determine what you will need to add to the data file in the way of user defined columns.

.gazedata File	
Column	***Definition***
Subject	Subject number (from E-Prime Startup Info).
Session	Session number (from E-Prime Startup Info).
ID	Counter of the current numbered gaze data point acquired.
TETTime	A timestamp from the TET hardware, derived from the TimestampSec value, converted to milliseconds.
RTTime	The timestamp value based on E-Prime's clock. Delay time on packet is determined and subtracted out using the TETServer time. This is dependent on the synchronization type chosen in the TobiiEyeTracker Device. The default is Local, but this feature can be turned off via the TET Device in the Experiment Object properties.
CursorX	The X axis pixel location corresponding to where the gaze is located on the screen and accounting for screen resolution (width).
CursorY	The Y axis pixel location corresponding to where the gaze is located on the screen and accounting for screen resolution (height).
TimeStampSec	A timestamp from the TET hardware when the gaze data point was recorded (seconds).
TimeStampMicroSec	A timestamp from the TET hardware when the gaze data point was recorded (microseconds).
XGazePosLeftEye	The horizontal position of the gaze point on a normalized scale (0 is left side of screen, 1 is right side of screen).
YGazePosLeftEye	The vertical position of the gaze point (0 is top, 1 is bottom).
XCameraPosLeftEye	The horizontal location of the pupil in the camera image (0 is left edge, 1 is right edge).
YCameraPosLeftEye	The vertical location of the pupil in the camera image (0 is top, 1 is bottom).
DiameterPupilLeftEye	The size of the pupil in millimeters.
DistanceLeftEye	The distance from the camera to the eye in millimeters (measured from surface of eye to center of the filter in front of the camera).
ValidityLeftEye	Validity of the gaze data.
XGazePosRightEye	The horizontal position of the gaze point on a normalized scale (0 is left side of screen, 1 is right side of screen).
YGazePosRightEye	The vertical position of the gaze point (0 is top, 1 is bottom).
XCameraPosRightEye	The horizontal location of the pupil in the camera image (0 is left edge, 1 is right edge).
YCameraPosRightEye	The vertical location of the pupil in the camera image (0 is top, 1 is bottom).
DiameterPupilRightEye	The size of the pupil in millimeters.
DistanceRightEye	The distance from the camera to the eye in millimeters (measured from surface of eye to center of the filter in front of the camera).
ValidityRightEye	Validity of the gaze data.

Task 2: Validity Ratings

Familiarize yourself with the validity ratings for the .gazedata output.

The rows corresponding to the columns listed on the previous page contain the values of the output data. In some cases, the values need no explanation because they are timestamps or millimeters. In other cases, the data values are single digit numbers. When this is the case it is necessary to know the meaning assigned to the numeric value. The table below defines the meaning of the number.

Validity Ratings	
0	The system is certain that it has recorded all relevant data for the particular eye and that the data recorded belongs to the particular eye (no risk of confusing left eye with right eye by the system).
1	The system has only recorded one eye, and has made some assumptions and estimations regarding if the recorded eye is left or right. However, it is still highly probable that the estimations done are correct. The validity code on the other eye is in this case always set to 3.
2	The system has only recorded one eye and has no way of determining if this is the left or the right eye.
3	The system is fairly confident that the actual gaze data is incorrect or corrupted. The other eye will always have a validity code of 1.
4	The actual gaze data is missing or incorrect. A couple of gaze data with validity code of 4 on both eyes, followed by a number of gaze data with validity code of 0 on both eyes, is usually a sign of a blink.

Task 3: Open the .gazedata File and Examine the Structure and Content

View the User-defined variables in the .gazedata file

Data is collected at the rate of your Tobii Eye Tracker device. When a sample is timestamped, a row is written to the .gazedata file populating the columns. The next steps in the tutorial will explain the user defined columns in the .gazedata file and highlight the script used to create them. If you do not have the .gazedata file open, see **Task 14: Open the Eye .gazedata File**, (Page 65) for instructions on how to navigate to and open the .gazedata file. The SaveGazeData InLine begins with the TET_HandlePreRelease line. The purpose of this code is to prevent you from losing any data. For more information, please refer to **Chapter 3: Tobii PackageCall Reference**, (Page 147).

1) **TrialID Column**: Script used to create assign variable. (location: SaveGazeData InLine):

   theUserEyeGazeData.TrialId = c.GetAttrib(c.GetAttrib ("Running") & ".Sample")

   Purpose: Unique identifier used to keep track of the trial number.

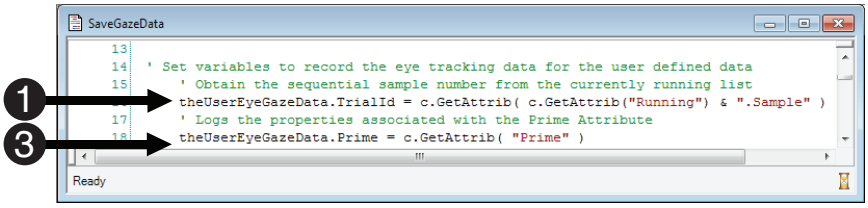
2) The **TrialID column *increases*** for each trial. The **first trial** is labeled **one** and the **second trial** is labeled **two**.

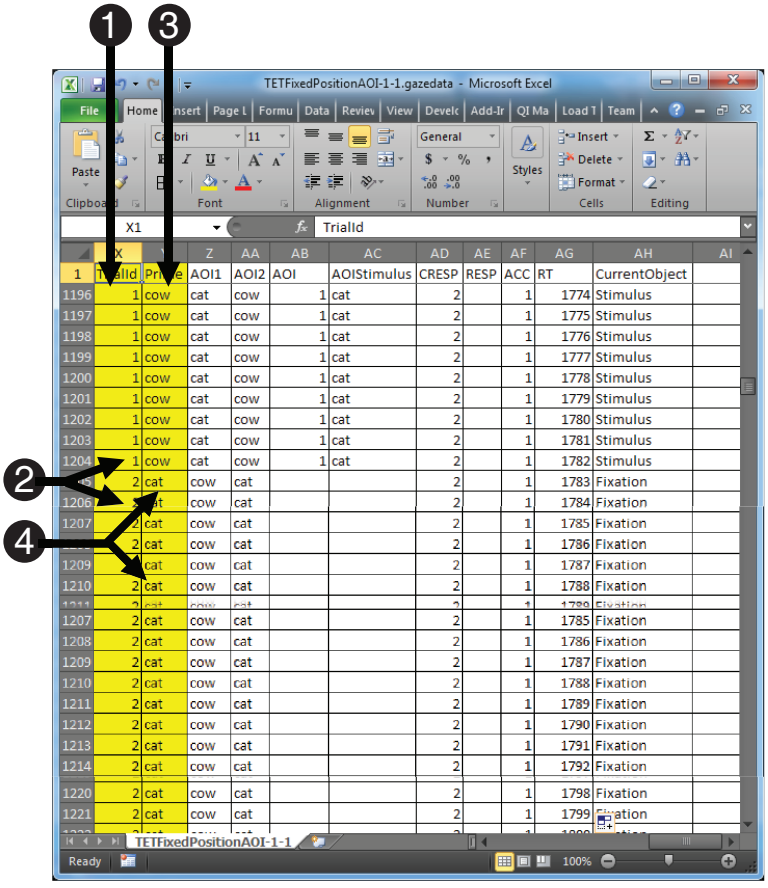
3) **Prime Column**: Script used to assign variable. (location: SaveGazeData InLine):

   theUserEyeGazeData.Prime = c.GetAttrib("Prime")

   Purpose: Logs properties associated with Prime.

4) The **Prime column *shows*** what sound file was played when the Prime Object was on screen. The **first Prime** was **cat**, and the **second Prime** was **cow**.





Task 3 (continued): Open the .gazedata File and Examine the Structure and Content

View the User-defined variables in the .gazedata file

When running an experiment you will want to keep track of your AOIs in the .gazedata file. The script below creates and populates the AOI variables for the TETFixedPositionAOI.es2 experiment. In this experiment there are only two AOIs. However, your experiment may need to have more variables to hold the information about your AOIs.

5) **AOI1 Column**: Script used to assign variable.
(location: SaveGazeData InLine):

   theUserEyeGazeData.AOI1 = c.GetAttrib(“LeftImage”)

   Purpose: Sets the LeftImage Attribute (TrialList) as AOI1.

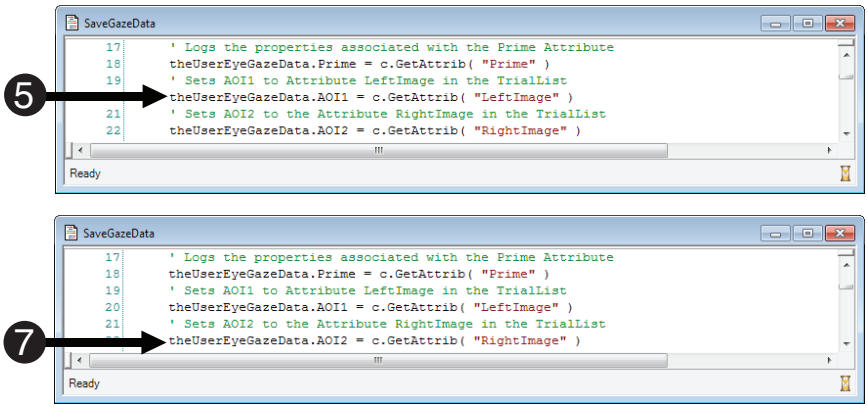
6) The **AOI1** is ***assigned*** the **LeftImage attribute** from the TrialList. The **AOI1 column *shows*** what image was displayed on the left side of the screen on a trial by trial basis. In this example, the LeftImage is a cat.

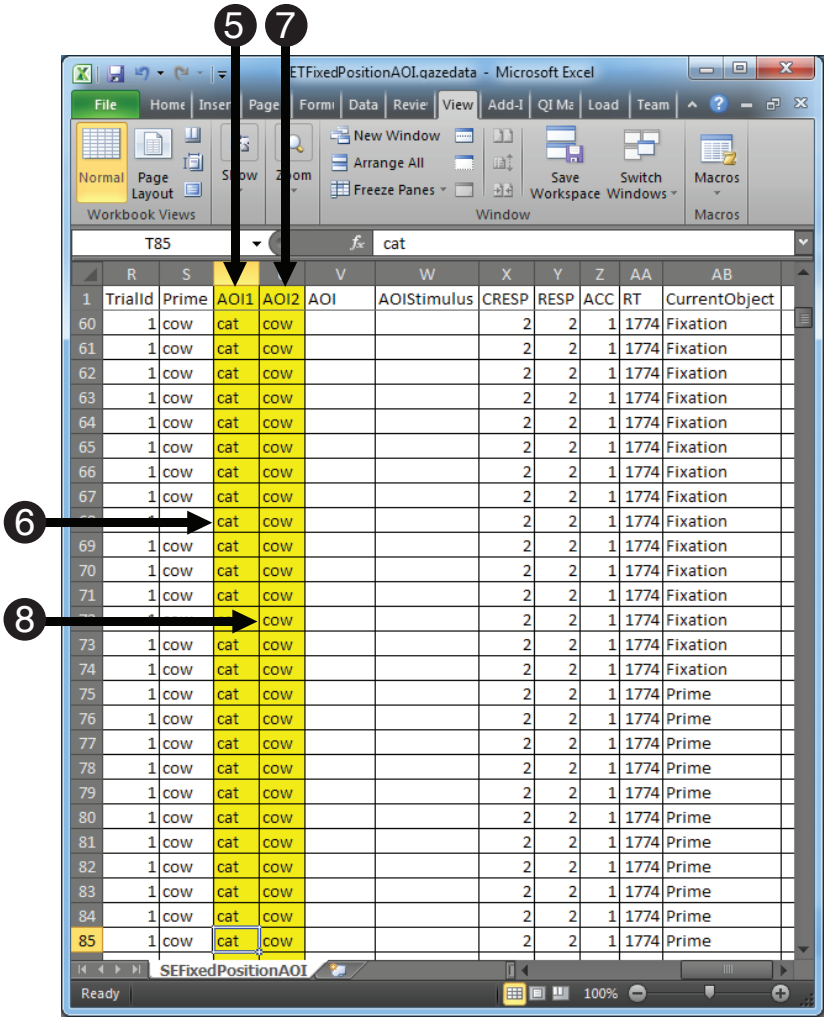
7) **AOI2 Column**: Script used to assign variable.
(location: SaveGazeData InLine):

   theUserEyeGazeData.AOI1 = c.GetAttrib(“RightImage”)

   Purpose: Sets the RightImage Attribute (TrialList) as AOI2.

8) The **AOI2** is ***assigned*** the **RightImage attribute** from the TrialList. The **AOI2 column *shows*** what image was displayed on the right side of the screen on a trial by trial basis. In this example, the RightImage is a cow.





Task 4: Understand the AOI and AOIStimulus Variables

Learn how the AOI and AOIStimulus variables are created and what their purpose is.

The AOI and AOIStimulus variables are special variables that hold information about what was on screen at a certain point in time. This information is determined in the SaveGazeData InLine within the experiment. Below, in the screenshot labeled #1 are the variables that indicate which AOI the participant is currently viewing, and the name corresponding to that AOI. Then the script goes on to determine which case is true (discussed on next page). The SaveGazeData InLine and the UserScript both begin with the TET_HandlePreRelease line. The purpose of this script to will prevent you from losing any data. For more information see **Chapter 3: Tobii PackageCall Reference**, (Page 147).

1) The variables that indicate which AOI the participant is currently viewing, and the name corresponding to that AOI.

2) **AOI Column**:
Script used to create variable. (location: SaveGazeData InLine):

theUserEyeGazeData.AOI = " "

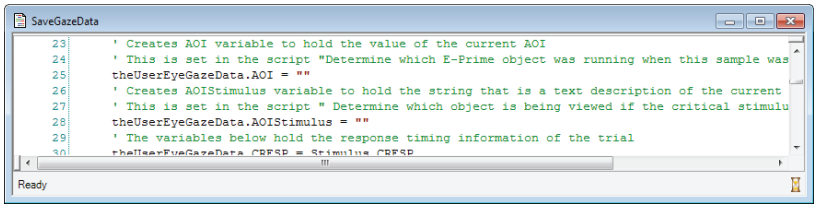
Purpose: Creates AOI variable to hold the value of the AOI the participant is currently looking at.

3) **AOIStimulus Column**:
Script used to create variable. (location: SaveGazeData InLine):

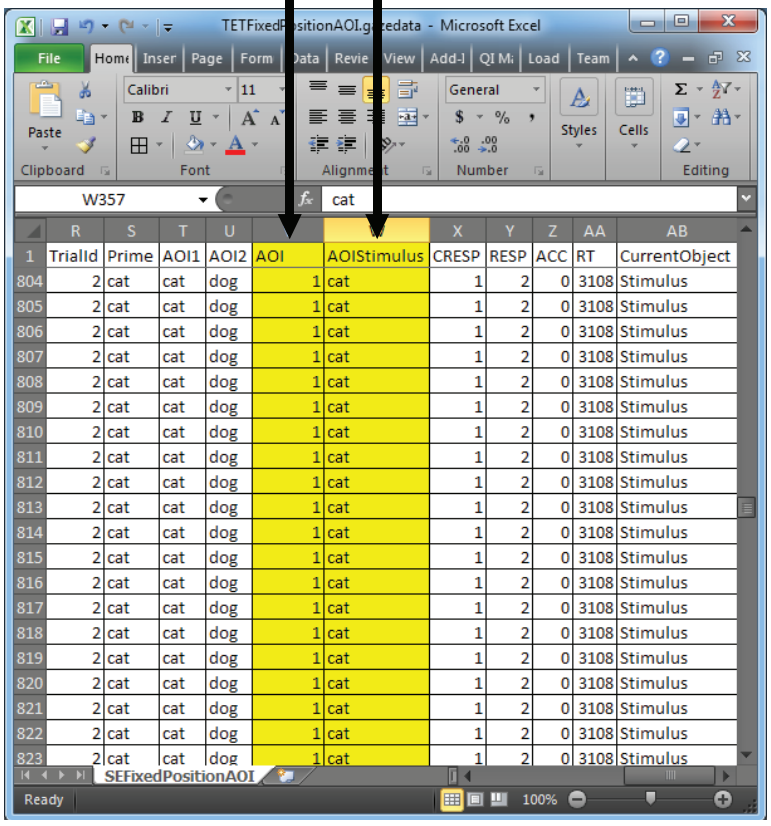
theUserEyeGazeData.
AOIStimulus = " "

Purpose: Creates AOIStimulus variable to hold the string that is a text description of the current AOI.

1



2 3



	R	S	T	U	V	W	X	Y	Z	AA	AB
1	TrialId	Prime	AOI1	AOI2	AOI	AOIStimulus	CRESP	RESP	ACC	RT	CurrentObject
804	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
805	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
806	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
807	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
808	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
809	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
810	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
811	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
812	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
813	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
814	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
815	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
816	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
817	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
818	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
819	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
820	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
821	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
822	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus
823	2	cat	cat	dog	1	cat	1	2	0	3108	Stimulus

Task 4: (continued) Understand the AOI and AOIStimulus Variables

Learn how the AOI and AOIStimulus variables are created and what their purpose is.

Once the hit test has determined where the object was on the screen, the script uses a set of assignment statements to determine which AOI is in that position. This is accomplished by the Case statements below. Once a case is determined to be true, it is assigned to the AOI and AOIStimulus variables accordingly.

4) This line of script performs a hit test.

5) Case "AOI1" is ***true*** when the **hit test coordinates *correspond*** with the location of **Stimulus Slide Sub-Object** named AOI1.

6) Case "AOI2" is ***true*** when the **hit test coordinates *correspond*** with the location of **Stimulus Slide Sub-Object** named AOI2.

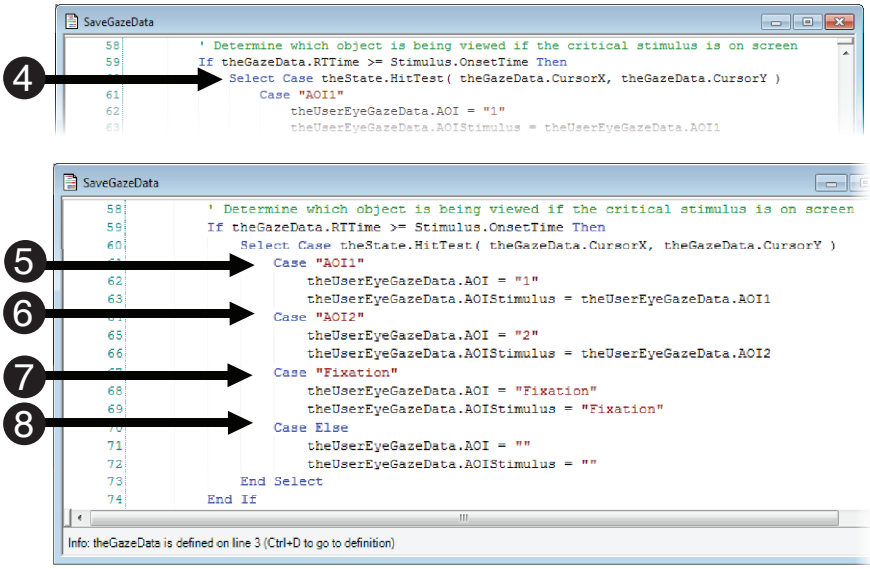
7) Case "Fixation" is ***true*** if the **hit test coordinates *correspond*** with the location of the **Stimulus Slide Sub-Object** Fixation.

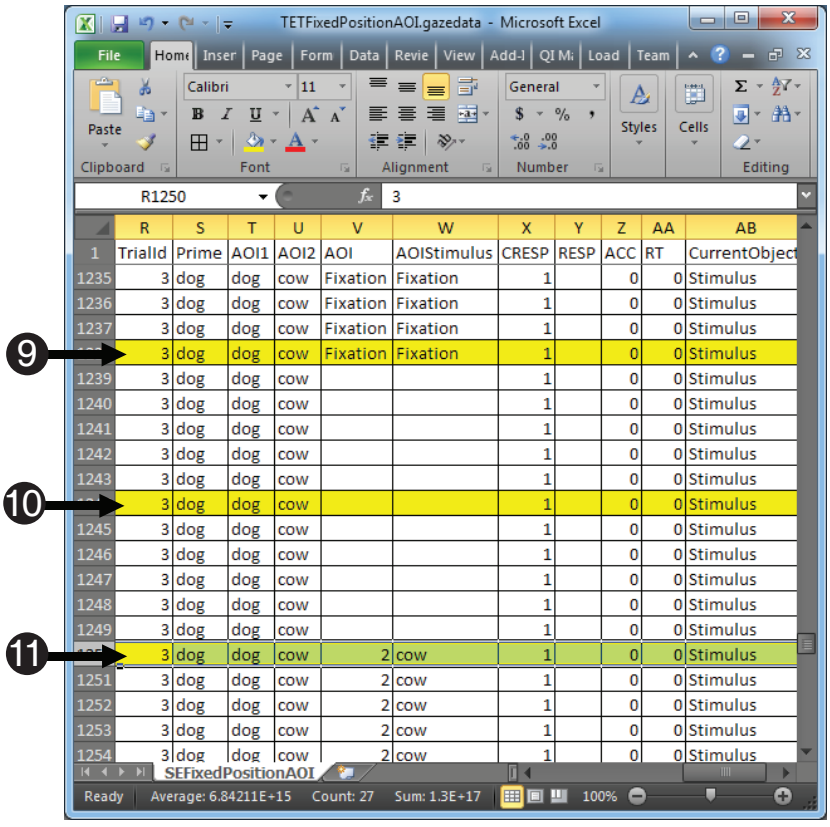
8) Case "Else" is ***true*** if the **hit test coordinates *correspond*** to a location on the **Stimulus Slide** that does not have a **Sub-Object** associated with it.

9) Example of when Case = "Fixation"

10) Example of when Case = "Else"

11) Example of when Case = "AOI1"





Task 5: Understand the Remaining Variables in the .gazedata File

Learn how the remaining variables in the .gazedata file are created and what their purpose is.

The variables CRESP, RESP, ACC, and RT tell us information about accuracy, the response made, and reaction time. This is important information to have when doing the analysis.

1) **CRESP Column**: Script used to create variable. (SaveGazeDataInline):

   theUserEyeGazeData.CRESP = Stimulus.CRESP

   Purpose: Records correct response to Stimulus Object for the trial.

2) **RESP Column**: Script used to create variable. (SaveGazeData InLine):

   theUserEyeGazeData.RESP = Stimulus.RESP

   Purpose: Records response to Stimulus Object for the trial.

3) **ACC Column**: Script used to create variable. (SaveGazeData InLine):

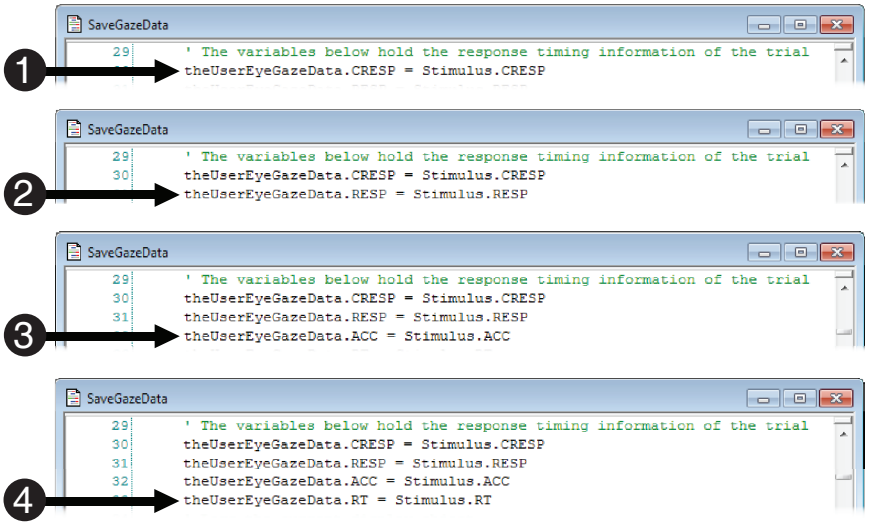
   theUserEyeGazeData.ACC = Stimulus.ACC

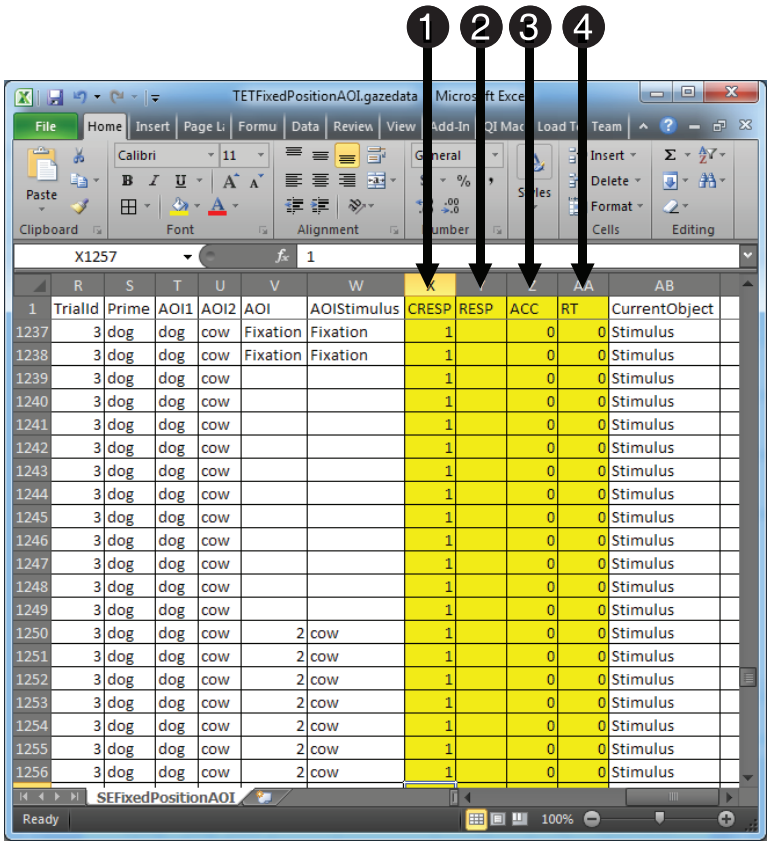
   Purpose: Scores accuracy of response to Stimulus Object for the trial.

4) **RT Column**: Script used to create variable. (SaveGazeData InLine):

   theUserEyeGazeData.RT = Stimulus.RT

   Purpose: Records time of response to Stimulus Object for the trial.





Task 5: (continued) Understand the Remaining Variables in the .gazedata File

Learn how the remaining variables in the .gazedata file are created and what their purpose is.

The remaining variable CurrentObject tells us what object was on screen during the trial.

5) **CurrentObject Column:** Script used to create variable. (SaveGazeData InLine):

   theUserEyeGazeData.
   CurrentObject = “ ”

   Purpose: Records the current Object for the gaze point.

⚠**NOTE:** *The script in screen shot #5 determines the value for this variable via the .OnsetTime property.*

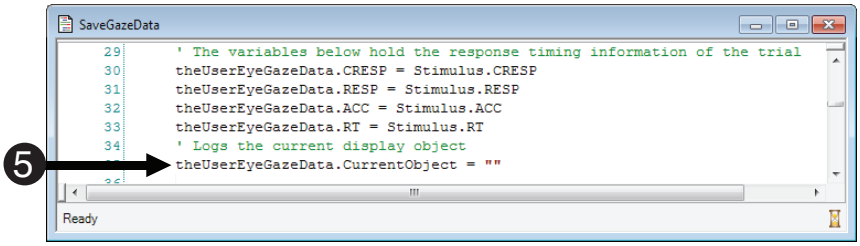
6) If statement used to determine if the Object is the Fixation Object. If the statement is **true**, **CurrentObject** is ***assigned*** the text value “Fixation”.

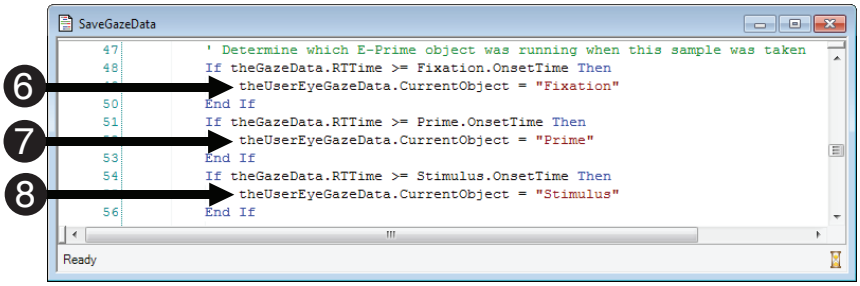
7) If statement used to determine if the Object is the Prime Object. If the statement is **true**, **CurrentObject** is ***assigned*** the text value “Prime”.

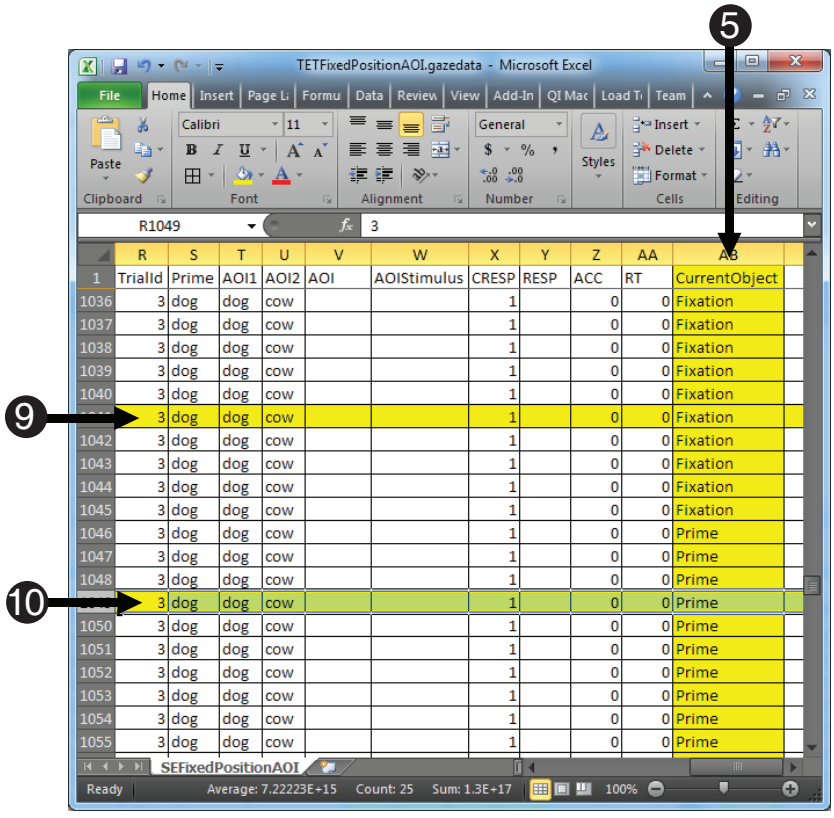
8) If statement used to determine if the Object is the Stimulus Object. If the statement is **true**, **CurrentObject** is ***assigned*** the text value “Stimulus”.

9) Example when the CurrentObject is Fixation.

10) Example when the CurrentObject is Prime.







Task 6: Run the Experiment

Run the experiment to verify that the eye tracker is working and to ensure the .gazedata file is written.

Now you should run the experiment to generate a data file. The experiment is the same experiment in the previous tutorial. Details of this experiment can be found in, **Tutorial 1, Task 17 (continued): Run the Experiment, (Page 44)** of this manual.

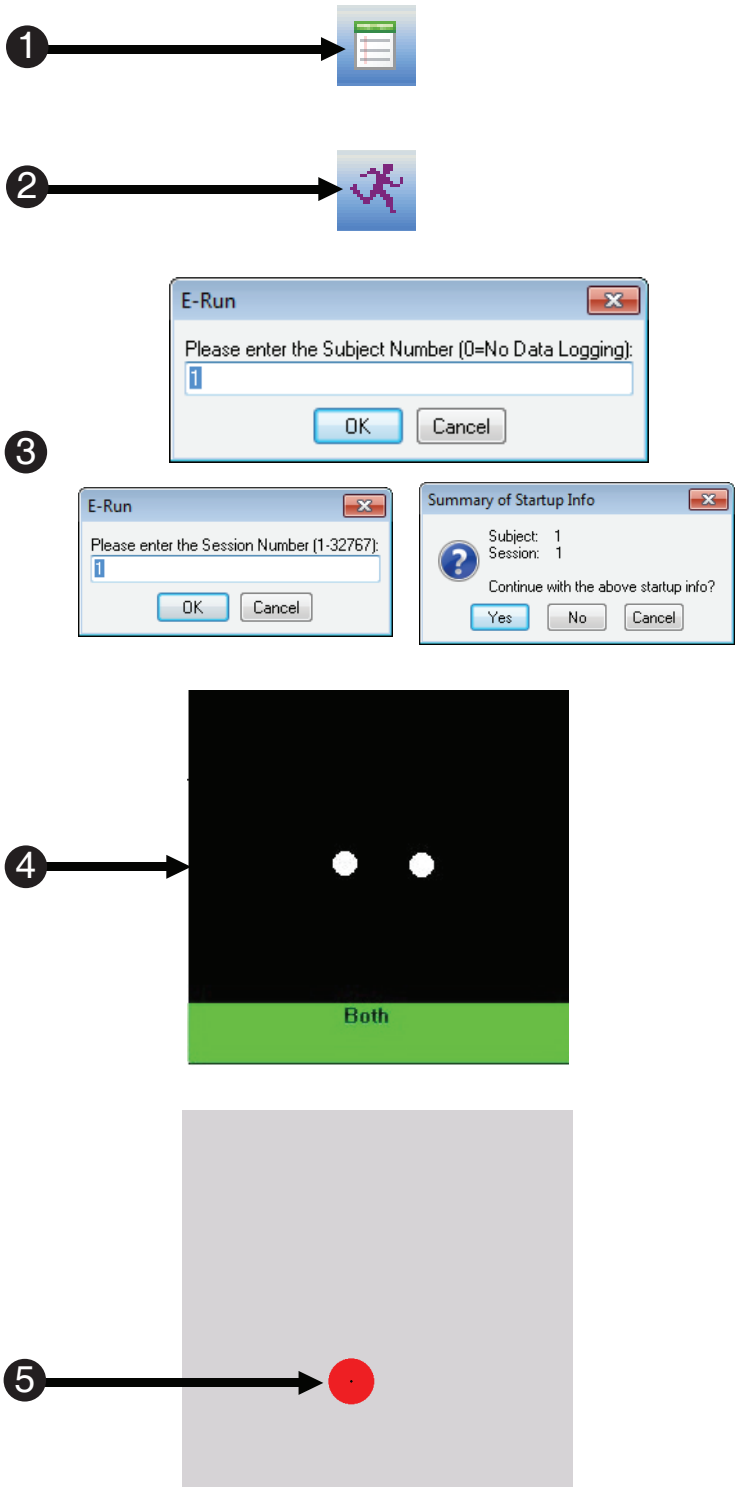
1) ***Press* Ctrl+S** to save your work before continuing. ***Click*** the **generate icon** or ***press* Ctrl+F7** to generate the script and ***check* it** for **errors**.

2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

3) ***Click* OK** to accept the **default values** for **Subject Number**, **Session Number** and **Yes** for **Summary of Startup Info**.

4) ***Look*** at the **screen** to ***verify*** the **eyes** are **stable** in the track status window. ***Ensure*** that there are **two white dots** that appear in the box and that the **bottom bar** is **green**. ***Press* Space** once both eyes are stable.

5) ***Run*** through the **calibration process** by ***following*** the **dots** on the **screen** with your **eyes**.



Task 6 (continued): Run the Experiment

Run the experiment to verify that the eye tracker is working and to ensure the .gazedata file is written.

Once you have completed calibration, you will be prompted to verify that the data is acceptable.

6) ***Accept*** the **calibration**.

⚠ **NOTE:** *Please refer to* **Tutorial 1, Task 17 (continued): Run the Experiment, (Page 44)** *for additional information about calibration.*

Tutorial 4: **Working with Eye Gaze Data Interactively**

Summary:

In the previous tutorials, eye gaze data was “passively” collected by E-Prime and combined with E-Prime data to create an output file. When used “actively”, eye gaze data may be used to allow the paradigm to “interact” with the raw eye gaze data properties during the trial (as with the “Wait for Fixation” example). The current tutorial further illustrates techniques for using eye gaze data interactively.

E-Prime can also be used to perform calculations for analysis at runtime. Runtime analysis results can then be passed on to the data file for use in post hoc analysis. In this tutorial, you will analyze the eye gaze data while it is being collected to determine which object on the screen is being fixated on by the participant.

During this tutorial, you will set up the TETVaryingPositionAOITracking.es2 experiment to actively access the .gazedata during a trial. If you are not familiar with the TETVaryingPositionAOITracking.es2 experiment, it is recommended that you run the experiment now.

In this experiment you will be shown a target and asked to identify it later. The target consists of a letter with a specific orientation. You are shown the target, e.g., an upside down A. You are then shown a fixation followed by a randomly generated screen of the same letter in different orientations. The object is to identify the original target and to locate that exact letter and orientation. As you are looking through the letters, each is surrounded by a red box.

Because of the more detailed nature of this tutorial, the experiment provided has incorporated the necessary changes. The tutorial will walk you through the changes provided in the InLine script.

Goal:

This tutorial will teach you the concepts behind saving active eye tracking data.

Overview of Tasks:

- Open the TETVaryingPositionAOITracking.es2.
- Set the Stimulus Slide Object’s Property Pages.
- Confirm and review the necessary variable declarations.
- Review the script which is used to continuously inspect the current eye gaze data.
- Use the current eye gaze data to interactively highlight Areas of Interest (AOIs).
- Verify the overall experiment structure and run the experiment.

Estimated Time: 10-15 minutes

Task 1: Open the TETVaryingPositionAOITracking.es2 Experiment in E-Studio

Locate the E-Studio icon in the Start > All Programs > E-Prime menu and launch the application by selecting it. Load the TETVaryingPositionAOITracking.es2 sample experiment.

The E-Studio application is installed as part of the typical E-Prime installation. This application is used to create, modify, and test experiments within E-Prime. Open the E-Studio application, navigate to My Experiments\Tobii\Tutorials\TET\TETVaryingPositionAOITracking, and load the TETVaryingPositionAOITracking.es2 sample experiment.

1) ***Click*** on the Windows Start menu, ***select*** **All Programs**, and then ***select*** **E-Prime.** From the menu, ***click*** on **E-Studio** to launch the application.

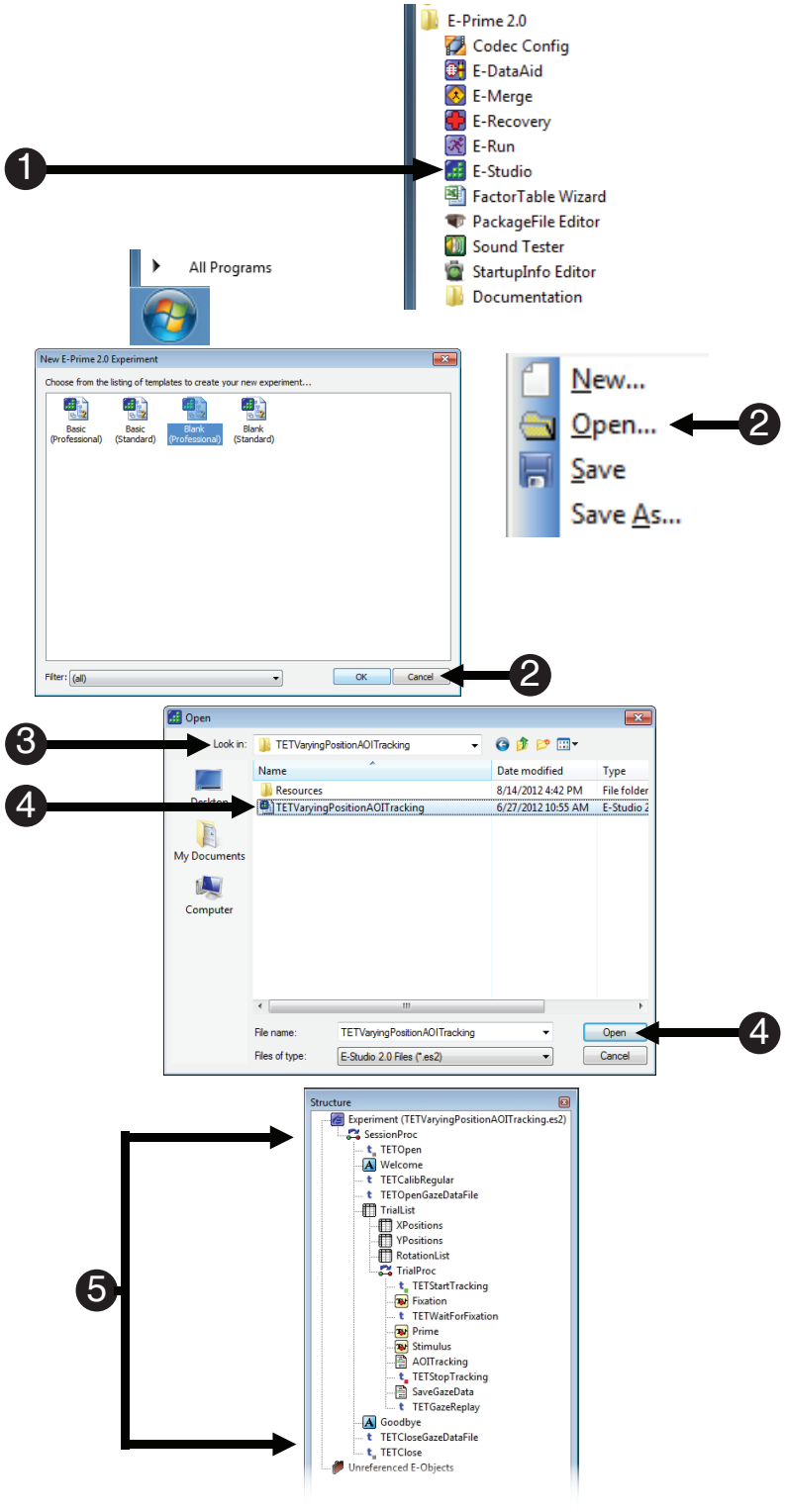
2) ***Click*** the **Cancel** button. ***Select*** **File** > **Open**.

3) ***Navigate*** to the "**…\My Experiments\Tobii\Tutorials\TET\TETVaryingPositionAOITracking**" folder to load the paradigm.

4) ***Select*** the "TETVaryingPositionAOITracking .es2" file and then ***click*** the **Open** button to load the paradigm into E-Studio.

⚠ **NOTE:** *If you cannot find the TETVarying PositionAOITracking .es2 file, you may need to refresh your E-Prime Samples and Tutorials folders. Select Tools\ Options… from the E-Studio menu bar then click "Copy Samples and Tutorials to My Experiments folder…"*

5) ***Compare*** the structure of the experiment you have opened to the one shown on the right.



Task 2: Set Stimulus Slide Object Property Pages

Open the Stimulus Slide Object Property Pages and edit the PreRelease and Duration.

This experiment tracks the participant's gaze on screen in real time. In order to accomplish this there are specific things that need to happen in the experiment. The first thing that needs to be changed is the Stimulus Slide Object. In order to track the participant's gaze in real time we will begin by confirming the PreRelease of the Stimulus Slide Object to be the same as the Duration. Prerelease controls the amount of time released during the processing of the current object to allow for setup of the next object. This will allow resources that are being used for the Stimulus Slide Object be released and made available for the AOITracking InLine. This will give the AOITracking InLine the ability to dynamically alter the border color of the AOI that is being viewed by the participant. For more information about PreRelease refer to the E-Prime documentation.

1) ***Double click*** the **Stimulus Slide** Object to open it in the workspace.

2) ***Click*** the white **Property Pages** button.

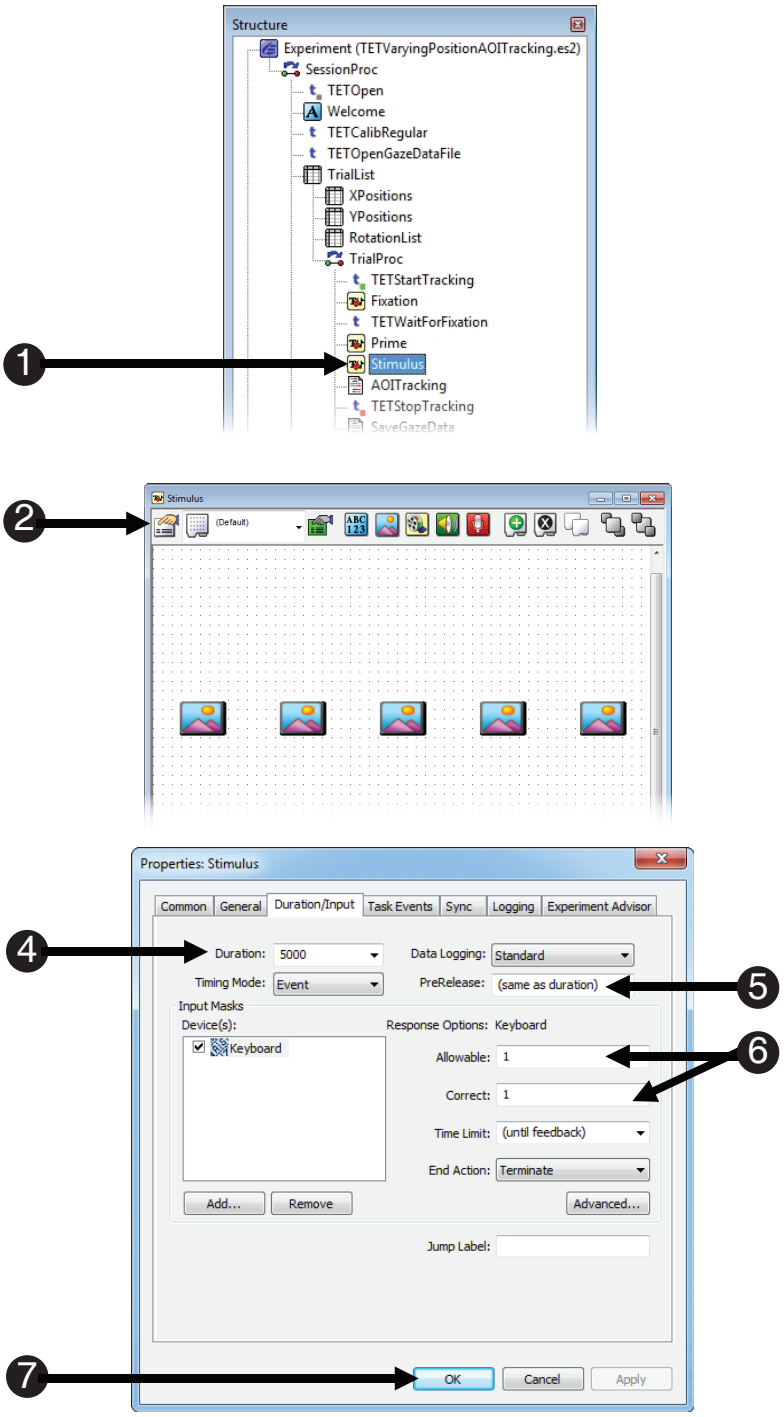
3) ***Select*** the **Duration/Input** tab.

4) ***Confirm*** the **Duration** is set to **5000.**

5) ***Confirm*** the **PreRelease** is set to **5000.**

6) ***Confirm*** the **Allowable** and **Correct** are both **1.**

7) ***Click*** **OK.**



Task 3: Understanding the Variable Declarations

Verify variable declarations.

The declaration in step 2 deals with a special E-Prime data type used to store individual eye gaze observations as reported by the Tobii Eye Tracker.

1) ***Double click*** the **AOITracking InLine** to open it in the workspace.

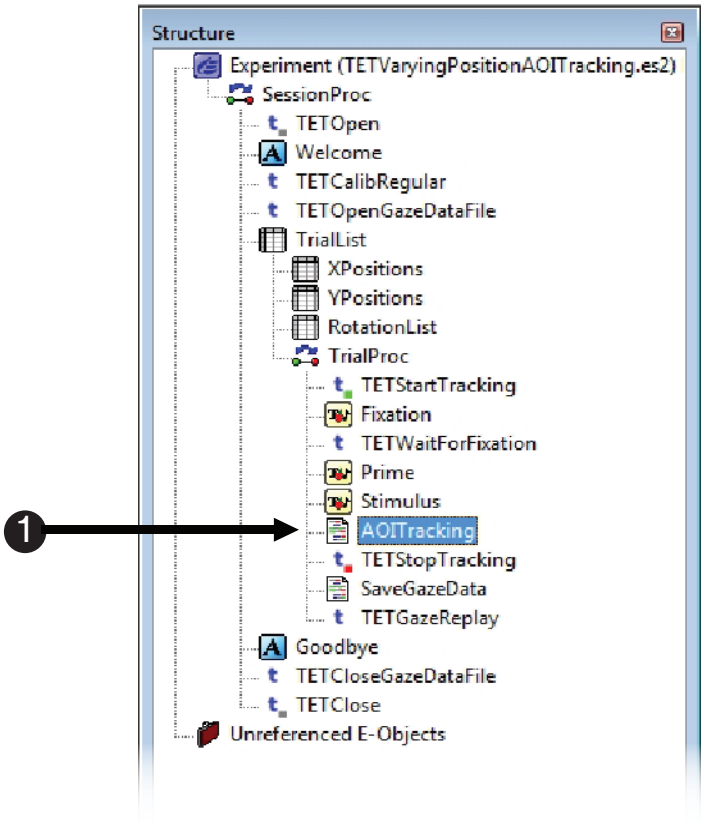
2) ***Confirm*** **Dim theGazeData As TobiiEyeTrackerResponseData**
Declares variable to store the gaze data sent from the Tobii Eye Tracker.

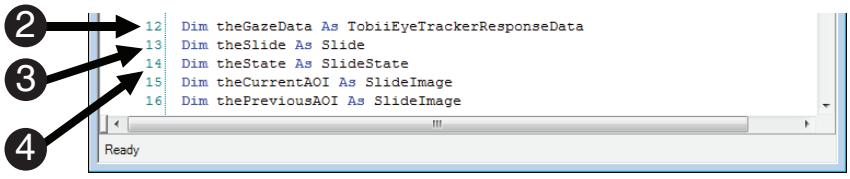
3) ***Confirm*** **Dim theSlide As Slide**
Declares variable for the slide.

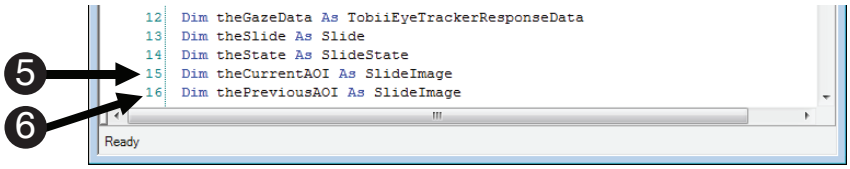
4) ***Confirm*** **Dim theState As SlideState**
Declares variable for the slide state.

5) ***Confirm*** **Dim theCurrentAOI As SlideImage**
Declares variable for the slide image sub-object which represents the current AOI being viewed.

6) ***Confirm*** **Dim thePreviousAOI As SlideImage**
Declares variable for the slide image sub-object which represents the current AOI being viewed.







Task 3 (continued): Understanding the Variable Declarations

The following steps allow you to confirm the declaration of each of the necessary variables.

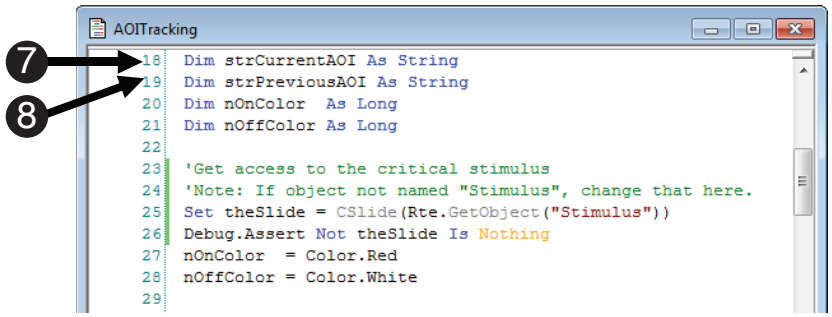
The next set of declarations allows for comparison of the previous and current AOI and the variables that determine border color. E-Prime will use this information to decide what color to make the border.

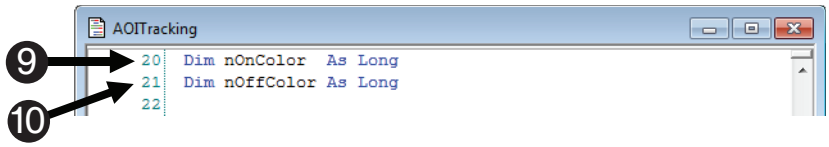
7) ***Confirm*** **Dim strCurrentAOI As String**
This variable will save the current AOI data.

8) ***Confirm*** **Dim strPreviousAOI As String**
This variable will save the previous AOI data.

9) ***Confirm*** **Dim nOnColor As Long**
This variable will set the “on” color for the border.

10) ***Confirm*** **Dim nOffColor As Long**
This variable will set the “off” color for the border.





Task 3 (continued): Understanding the Variable Declarations

The following steps allow you to confirm the declaration of each of the necessary variables.

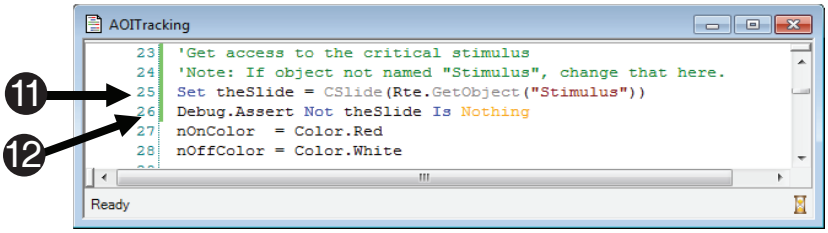
Next, theSlide is initialized, and the border color is set according to it's current state — off or on.

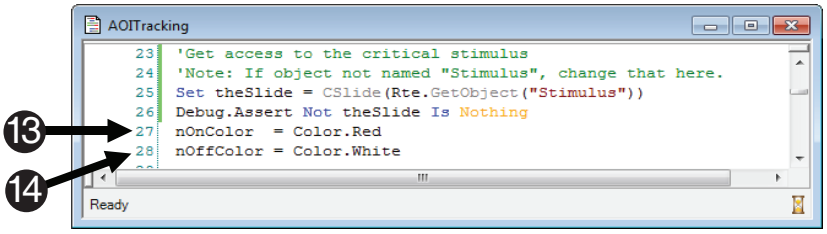
11) ***Confirm*** **Set theSlide = CSlide(Rte.GetObject("Stimulus"))**
This accesses the slide containing the critical stimulus. In this instance it is named Stimulus.

12) **The Debug.Assert Not theSlide Is Nothing**
Checks that the variable theSlide exists.

13) ***Confirm*** **nOnColor = Color.Red**
This variable will change the "on" border to red.

14) ***Confirm*** **nOffColor = Color.White**
This variable will change the "off" border to white.



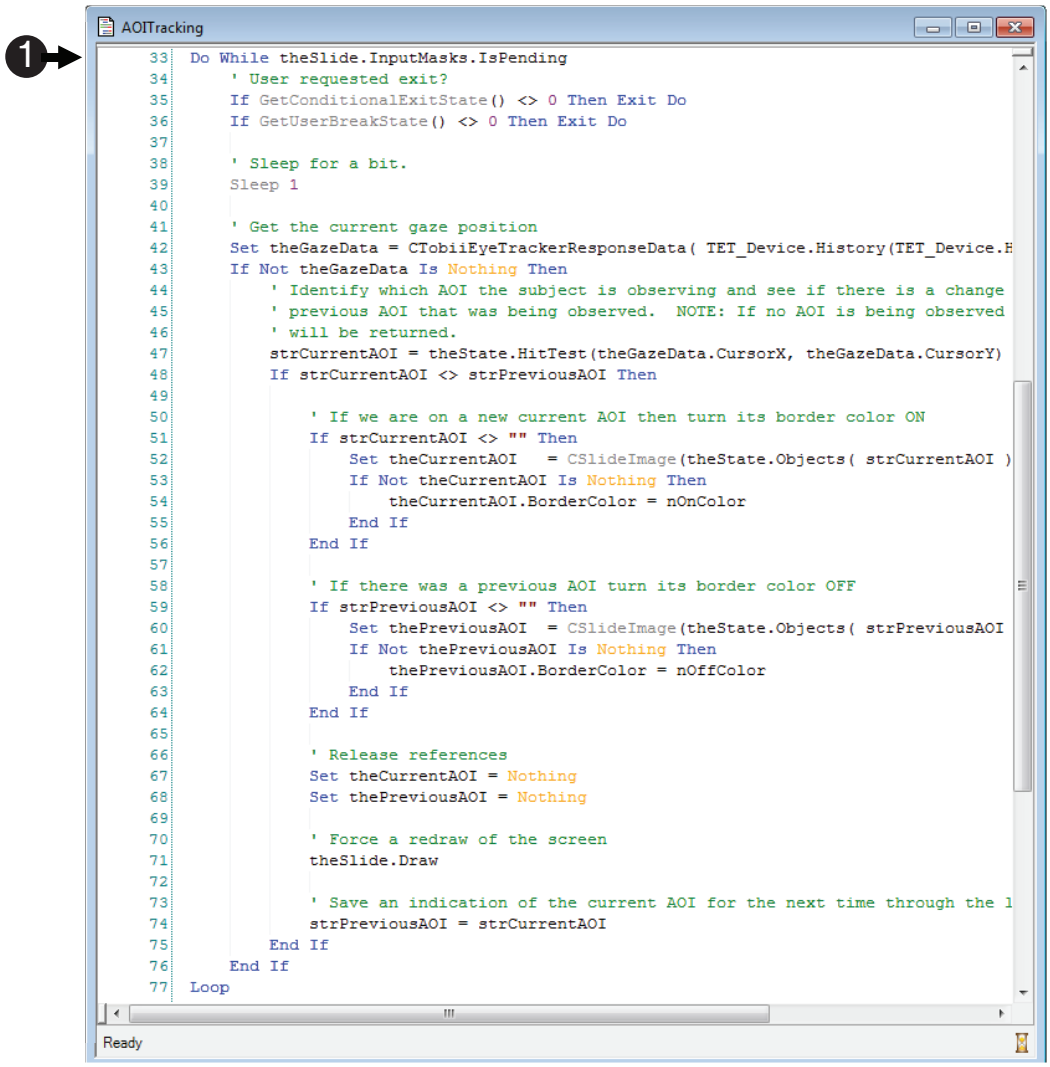


Task 4: The Do…Loop

Understand the basic function of the Do…Loop and how it works in the AOITracking InLine.

A Do…Loop is used in this experiment in conjunction with the IsPending method to continuously evaluate which, if any, AOI is being fixated on until a response is made. This allows E-Prime to analyze the eye gaze data to determine when a fixation moves “off of” and “onto” the available AOIs. Here this information is used to provide feedback to the participant as to what object they are fixating on.

1) Beginning of loop.



Task 5: AOI Tracking

Use the HitTest method to determine which AOI is being viewed and display the appropriate border.

In the previous step, a HitTest was performed using theGazeData.CursorX and .CursorY. We will now use that information to perform a hit test. Then we will determine which AOI is being viewed and alter the appearance of the slide border respectively.

1) ***Perform*** a **HitTest** on the **strCurrentAOI**.
(Line 47)

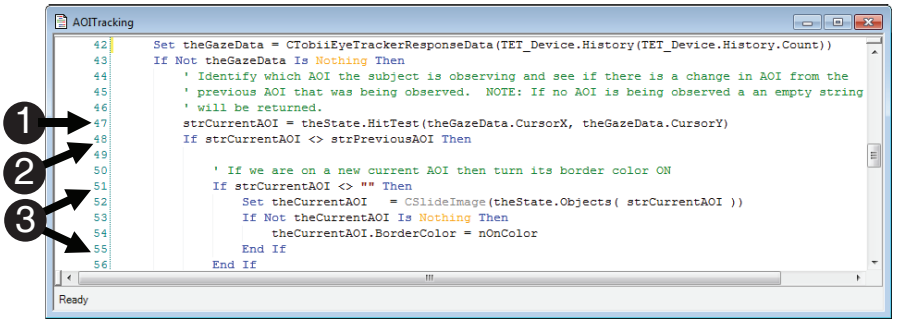
2) **If strCurrentAOI <> strPreviousAOI Then**
Conditional statement to determine if the current AOI is the same as the previous AOI.
(Line 48)

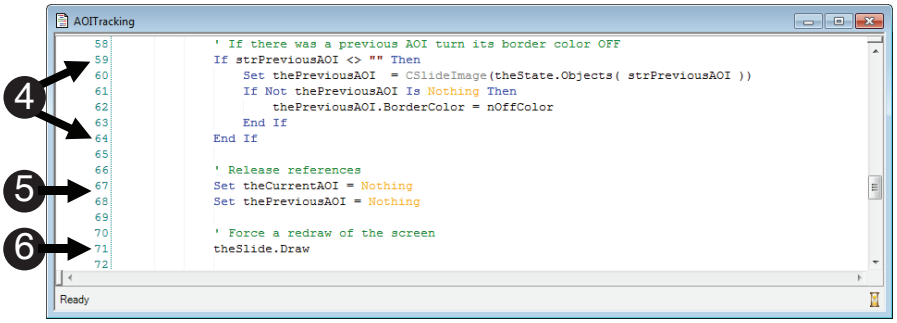
3) ***Use*** a **conditional statement** to ***switch*** **ON** the **BorderColor** if the **AOI** is *currently being viewed*.
(Lines 51-55)

4) ***Use*** a **conditional statement** to ***switch*** **OFF** the **BorderColor** if the **AOI** is *not currently being viewed*.
(Lines 59-64)

5) ***Release*** **theCurrentAOI** and the **PreviousAOI** references.
(Lines 67-68)

6) ***Redraw*** the **screen**.
(Line 71)





Task 6: Set the Values for the Next Trial

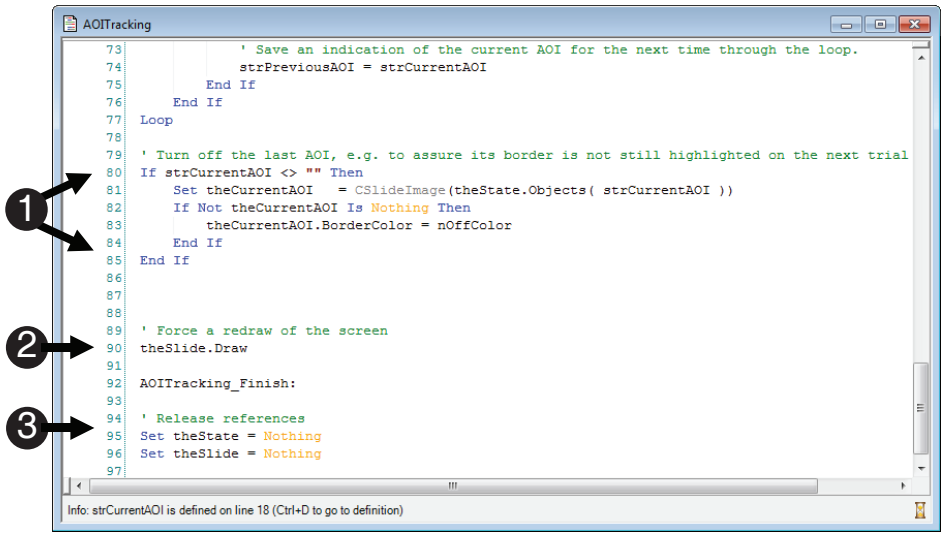
Clear the references and redraw the border for the next trial.

The last step before the next trial is to reset everything prior to starting the next trial. First, reset or turn off the BorderColor, redraw the screen and then release the variable references.

1) ***Turn off*** the **BorderColor**. *This will ensure it is not highlighted at the beginning of the next trial.* (Line 80-85)

2) ***Redraw*** the **screen.** (Line 90)

3) ***Release*** **References**. (Line 94-95)



Task 7: Run the Experiment

Run the experiment to verify that the eye tracker is working and to ensure the .gazedata file is written.

The next steps will walk you through generating the experimental script, starting the experimental paradigm, calibration, and running the experiment.

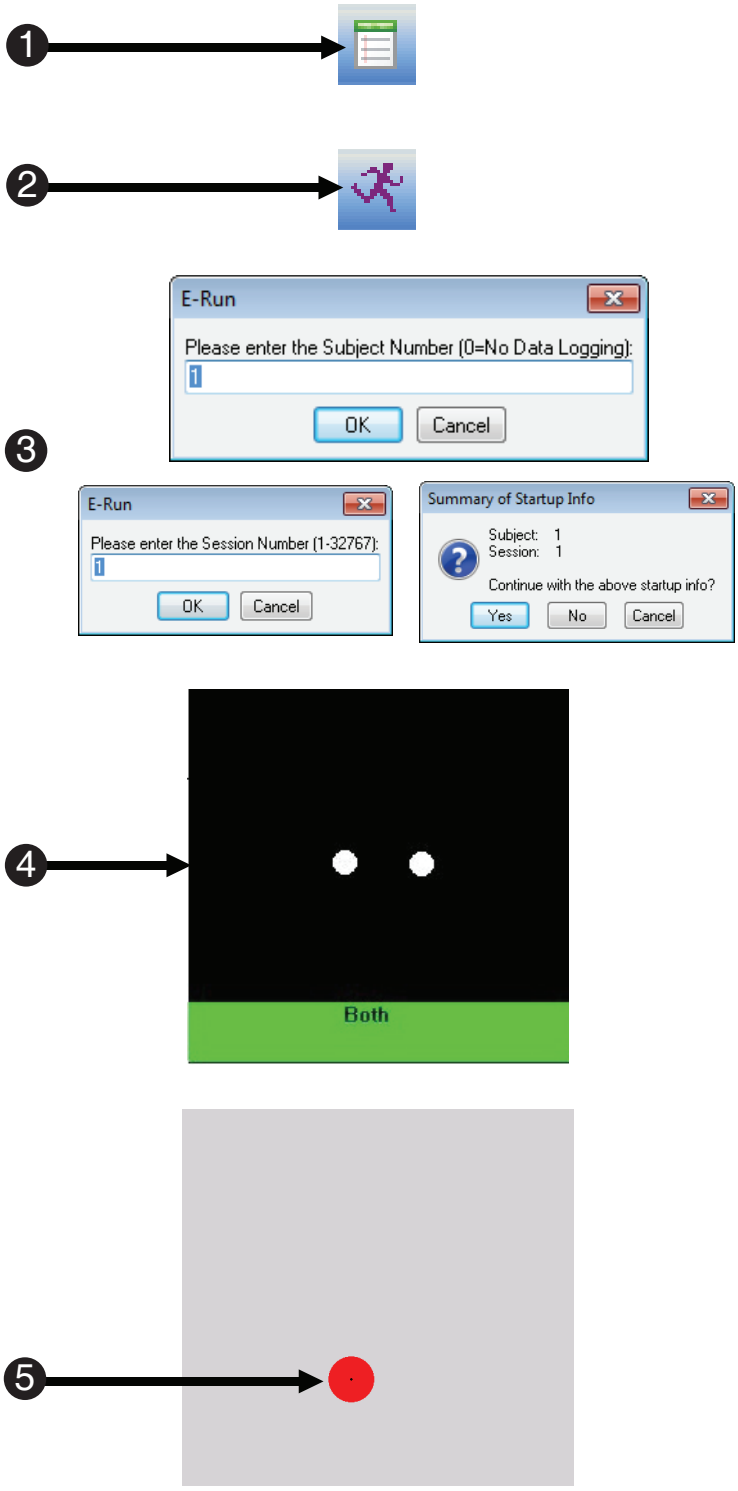
1) ***Press* Ctrl+S** to save your work before continuing. ***Click*** the **generate icon** or ***press* Ctrl+F7** to generate the script and ***check* it** for **errors**.

2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

3) ***Click* OK** to accept the **default values** for **Subject Number**, **Session Number** and **Yes** for **Summary of Startup Info**.

4) ***Look*** at the **screen** to ***verify*** the **eyes** are **stable** in the track status window. ***Ensure*** that there are **two white dots** that appear in the box and that the **bottom bar** is **green**. ***Press* Space** once both eyes are stable.

5) ***Run*** through the **calibration process** by ***following*** the **dots** on the **screen** with your **eyes**.



Task 7 (continued): Run the Experiment

Run the experiment to verify that the eye tracker is working, and to ensure the .gazedata file is written.

Once you have completed calibration, you will be prompted to verify that the data is acceptable.

6) ***Accept*** the **calibration**.

⚠ **NOTE:** *Please refer to* **Tutorial 1, Task 17 (continued): Run the Experiment, (Page 44)** *for additional information about calibration.*

Tutorial 5: **Adding Event Markers to an E-Prime Experiment**

Tutorial 5 shows you how to add ClearView PackageCall objects to an existing experiment in order to perform scene-based analyses in Tobii Studio. Tutorial 5 picks up where Tutorial 4 left off. If you have completed Tutorial 4, then you will begin Tutorial 5 by opening your finished Tutorial 4 experiment. If you have not completed Tutorial 4 then you will begin by opening the paradigm, TETFixedPositionAOI.es2. Details are provided in Step 1.

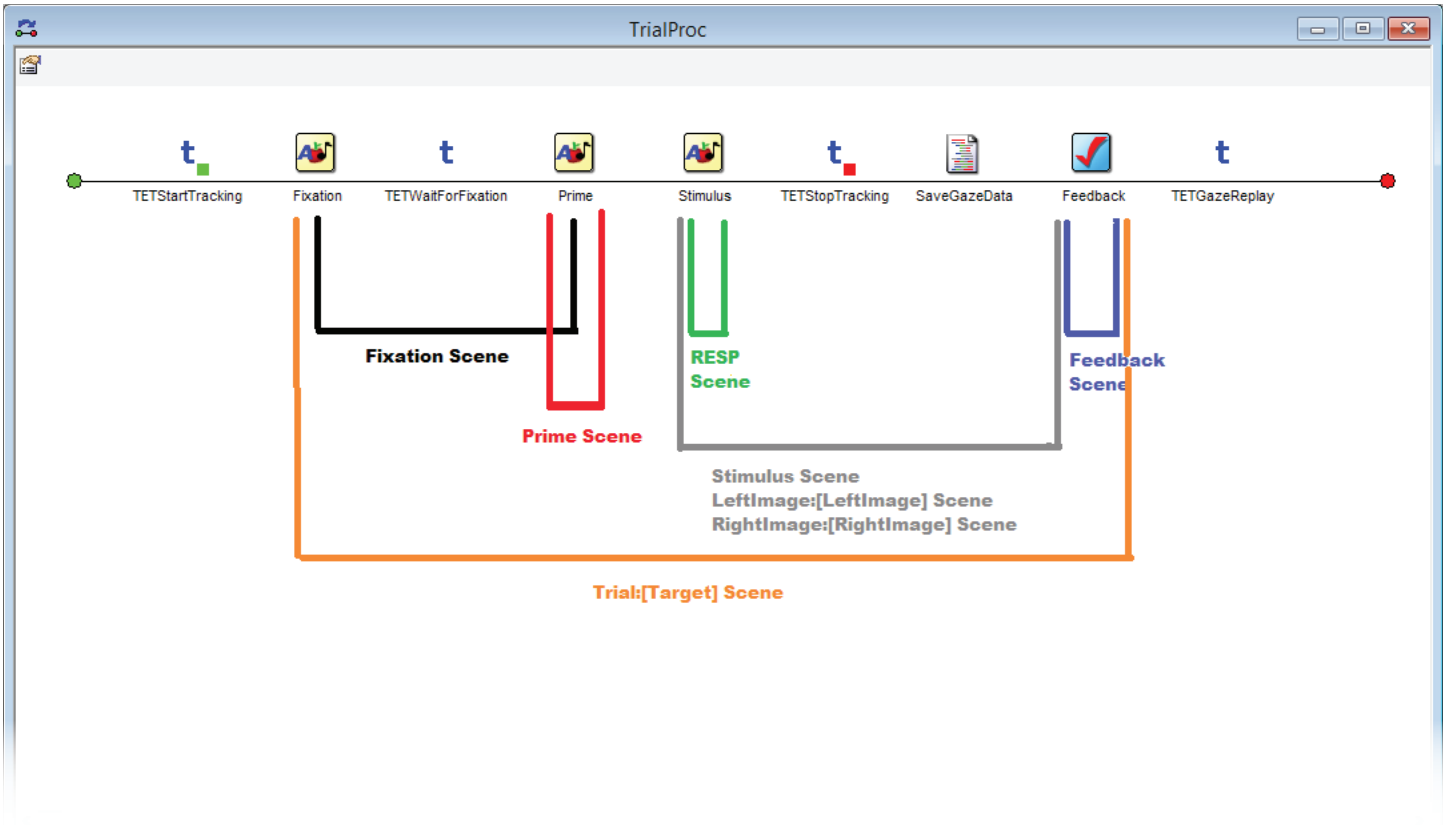
Summary:

Tutorials 1 through 4 focused on the functionality supported by the TET PackageFile, which is one of the two package files that are provided with EET. To review, the TET PackageFile routines support communication with the TET Server. Specifically, these routines support the capture and replay of gaze data at the conclusion of each trial, along with the creation of the .gazedata file that can be examined in Excel after the completion of the experiment. When working with the TET PackageFile, E-Prime is the application that is controlling and coordinating the experiment events. The experiment session begins by launching the experiment from E-Studio or with E-Run. During the experiment, E-Prime requests eye tracking information from the Tobii monitor, E-Prime script is called to present a replay of the gaze data for the trial that was just completed, and E-Prime combines the experiment data it collects (e.g., trial number, response time) with the Tobii eye tracking data and writes the combined information out to the .gazedata file. Lastly, once the experiment session is complete, the .gazedata file can be analyzed in Excel.

Tutorial 5 introduces the functionality supported by the second package file in EET, the ClearView (CV) PackageFile. The CV PackageFile is designed for users who need to perform more visually-sophisticated eye tracking data analysis in Tobii Studio than can be performed in Excel on the .gazedata file. Specifically, Tobii Studio supports analysis by scenes, where a scene is defined as a pair of scene markers that is inserted into the data stream. Each scene has a name, such as “Fixation”, and a Scene Start and Scene Stop marker. The CV PackageFile supports the creation of scenes by providing customized Task Events that send the scene markers to Tobii Studio. By using the CV PackageCalls and E-Prime's Task Events, an E-Prime experiment can be customized to instruct Tobii Studio to start a Scene, end a Scene, and to log important events within the trial. Incorporating this functionality into an existing E-Prime experiment involves adding CV PackageCalls to the E-Prime experiment structure and adding Task Events to the critical E-Prime objects that correspond to the start and end of Scenes. This process is demonstrated in Tutorial 5.

The Task Events feature in E-Prime 2.0 Professional is a powerful tool, which enables several tasks to execute when several events occur. Examples of tasks that can be defined include writing a value to a communication port. Examples of events that can be identified as the triggering event include when an E-Prime display object begins to execute. The Task Events feature is used extensively throughout this tutorial to send SceneStart and SceneStop markers to Tobii Studio for the various scenes (such as Fixation, Trial) that are defined for analysis in Tobii Studio. Users who are not familiar with E-Prime Task Events are urged to review Section 3.1 Task Events of the New Features/Reference Guide prior to completing this tutorial.

The figure on the following page illustrates the scenes that will be defined in this Tutorial. Note that some scenes, such as the Feedback scene, span on the duration of single object, while other scenes, such as Trial, span over multiple objects.



E-Prime and the Tobii Studio software run on different computers, which allow each application to complete their designated function. However, this requires that the computers communicate with one another. In the case of the two-computer setup using T-Series eye tracker, E-Prime and Tobii Studio both communicate with the Tobii Eye Tracker server via a network switch. PackageCalls and/or Task Events are added at the appropriate place in the E-Prime experiment, and the required parameter values are then specified and sent to Tobii Studio. The data from E-Prime is sent with the Ethernet port via the switch to Tobii Studio in under 10 ms. The switch connection allows for the data to be transmitted between the two software programs, but leaves Tobii Studio blind to what is being presented on screen by the E-Prime software.

In order for the scenes and segments displayed by E-Prime to be properly analyzed by Tobii Studio, the Tobii Studio information must first pass through a video capture card. This allows for a video of the experiment to be saved in Tobii Studio, leading to more robust and meaningful Tobii Studio data. Meanwhile, the visual aspects of the E-Prime experiment from are sent to a splitter. The splitter sends the signal to both the monitor and to the video capture card located on the Tobii Studio machine (typical delay is ~40 ms, but is dependent on the specific hardware configuration), and Tobii Studio collects the visual input via an external video device. For more information about the two computer set up see **Appendix B: Hardware Configurations, (Page 204)**.

Tobii Studio converts the E-Prime display into a Tobii Studio video; it can then overlay the gaze information from Tobii monitor onto the video. In order to do this, Tobii Studio must first be started so that the entire E-Prime experiment is included in the Tobii Studio video.

Goal:

By the end of this tutorial, you will have created a basic ClearView-enabled E-Prime paradigm that uses Task Events to send event markers to Tobii Studio for scene-based analysis. This tutorial creates eight scenes. In practice, this is likely to be many more scenes than you would typically define for a particular experiment. However, throughout the tutorial a variety of techniques are illustrated, including how to define multiple Task Events on a single object, performing multiple "tasks" (i.e. defining multiple scenes) within a single Task Event, creating scenes that span multiple display objects, and creating scenes that both start and stop on the same object. Each scene requires both a SceneStart and SceneStop, resulting in sixteen Task Events in this tutorial. Refer to the table below to keep track of the scenes:

Scene Name	Tutorial Task, SceneStart	E-Prime Object/ Event to trigger SceneStart	Tutorial Task, SceneStop	E-Prime Object/ Event to trigger SceneStop
Fixation	Tasks 10-11	Fixation.OnsetTime	Tasks 13-14	Prime.OnsetTime
Trial:[Target]	Task 12	Fixation.OnsetTime	Tasks 29-30	Feedback.OffsetTime
Prime	Tasks 15-16	Prime.OnsetTime	Tasks 17-18	Stimulus.OnsetTime
Stimulus	Tasks 19-20	Stimulus.OnsetTime	Tasks 25-26	Feedback.OnsetTime
LeftImage:[LeftImage]	Tasks 19-20	Stimulus.OnsetTime	Task 27	Feedback.OnsetTime
RightImage:[RightImage]	Tasks 19-20	Stimulus.OnsetTIme	Task 27	Feedback.OnsetTime
RESP	Tasks 19-20	Stimulus.OnsetTime	Tasks 21-22	Stimulus.Keyboard(1). MaxCountReached
			Tasks 23-24	Stimulus.Keyboard(1).TimeLimitReached
Feedback	Task 27	Feedback.OnsetTIme	Tasks 28-29	Feedback.OffsetTime

Overview of Tasks:

- Rename TETFixedPositionAOI.es2 as TETFixedPositionAOIMarkers.es2.
- Add the CV package file.
- Add the CV device to the Experiment Property Pages and edit the devices properties to enable communication between E-Prime and Tobii Studio.
- Add the CV PackageCall to initialize the CV package file.
- Add the CV PackageCalls to start/stop the communication between E-Prime and ClearView.
- Add the CV PackageCalls to start/stop recording the experiment to be viewed in Tobii Studio.
- Add and Edit the Target Attribute to the TrialList.
- Add the CV Task Events to mark the start of the scene(s) that correspond to the Trial Object, Fixation Object, Prime Object, Stimulus Object as well as LeftImage, RightImage, and Response. Use the table above as a guide to adding the aforementioned scene markers.

Estimated Time: 10-15 minutes

Task 1: Open the TETFixedPositionAOI.es2 Experiment in E-Studio

Locate the E-Studio icon in the Start > Programs > E-Prime menu and launch the application by double clicking on it. Load the TETFixedPositionAOI.es2 sample experiment.

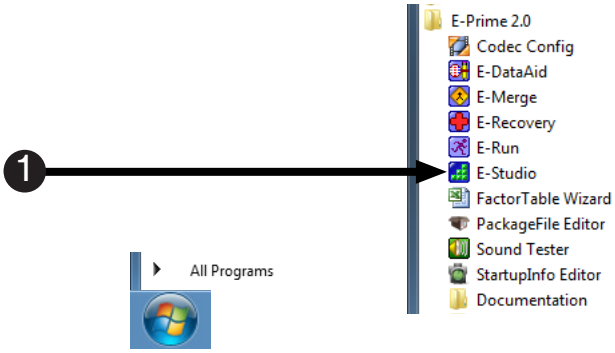
Open the E-Studio application, navigate to …My Experiments\Tobii\Tutorials\CV\ TETFixedPositionAOIMarkers, and load the TETFixedPositionAOI.es2 experiment. This experiment will act as a template on which we will place the ClearView PackageCallObjects and Task Events.

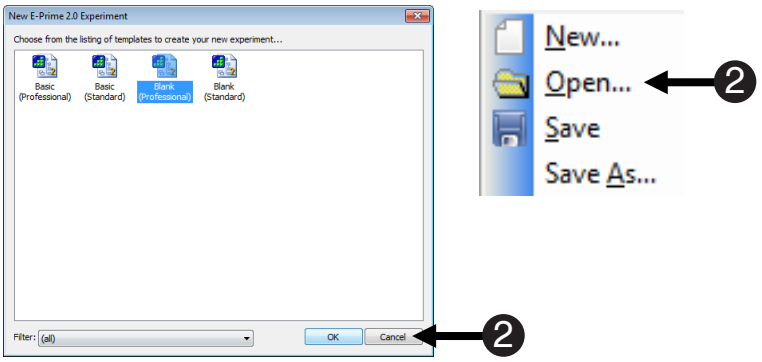
1) ***Click*** on the Windows **Start** menu, ***select*** **All Programs**, and then ***select*** **E-Prime**. From the menu, ***click*** on **E-Studio** to launch the application.

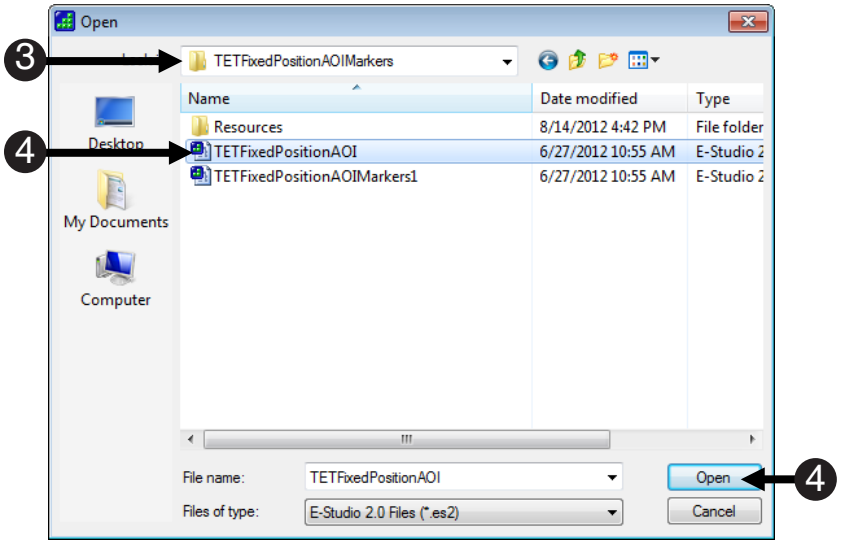
2) ***Click*** the **Cancel** button. ***Select*** **File** > **Open**.

3) ***Navigate*** to the "…\My Experiments\Tobii\Tutorials\CV\ TETFixedPositionAOIMarkers" folder.

4) ***Select*** the **TETFixedPositionAOI.es2** file and then ***click*** the **Open** button to ***load*** the **paradigm** into E-Studio.







Task 2: Save the Experiment Under a New Name

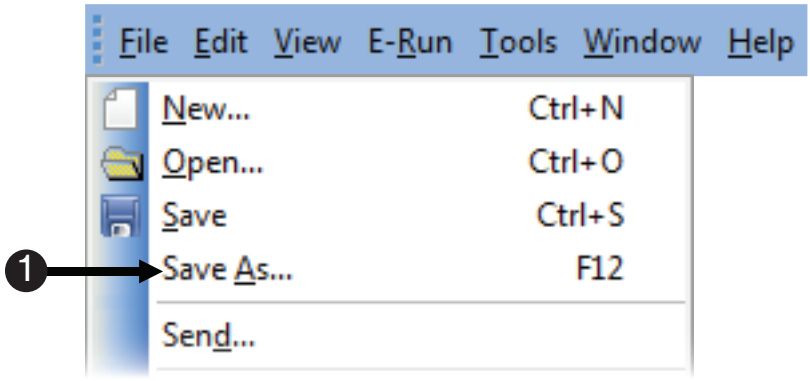
Save the TETFixedPositionAOI.es2 experiment in the same folder under the new name "TETFixedPositionAOIMarkers.es2."

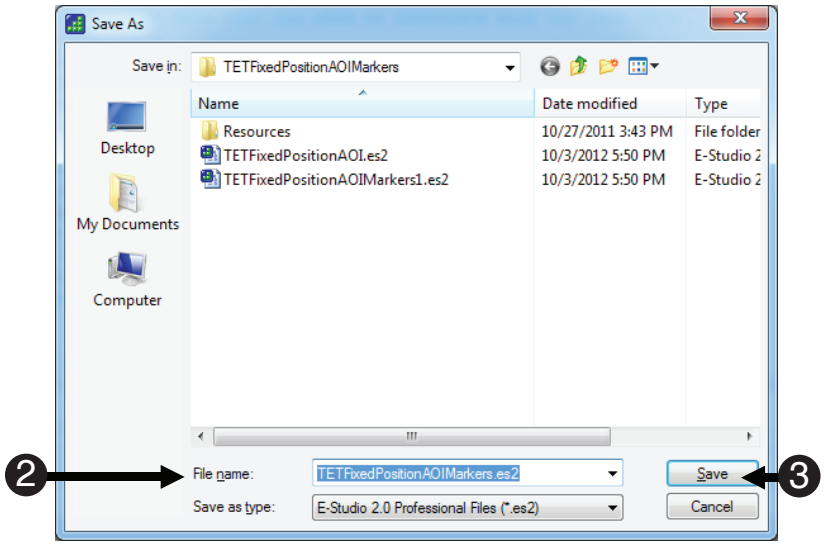
Rename the experiment, but be sure to save it in the same folder (…\My Experiments\Tobii\Tutorials\CV\TETFixedPositionAOIMarkers") so that any resources within the experiment will remain valid and can be reused.

1) ***Select*** **File** > **Save As**… from the application menu bar.

2) ***Type*** **TETFixedPositionAOIMarkers.es2** as the new name in the **File name** field.

3) ***Click*** the **Save** button.





Task 3: Add an Attribute to the TrialList and name it Target

Add an Attribute named "Target" to the TrialList.

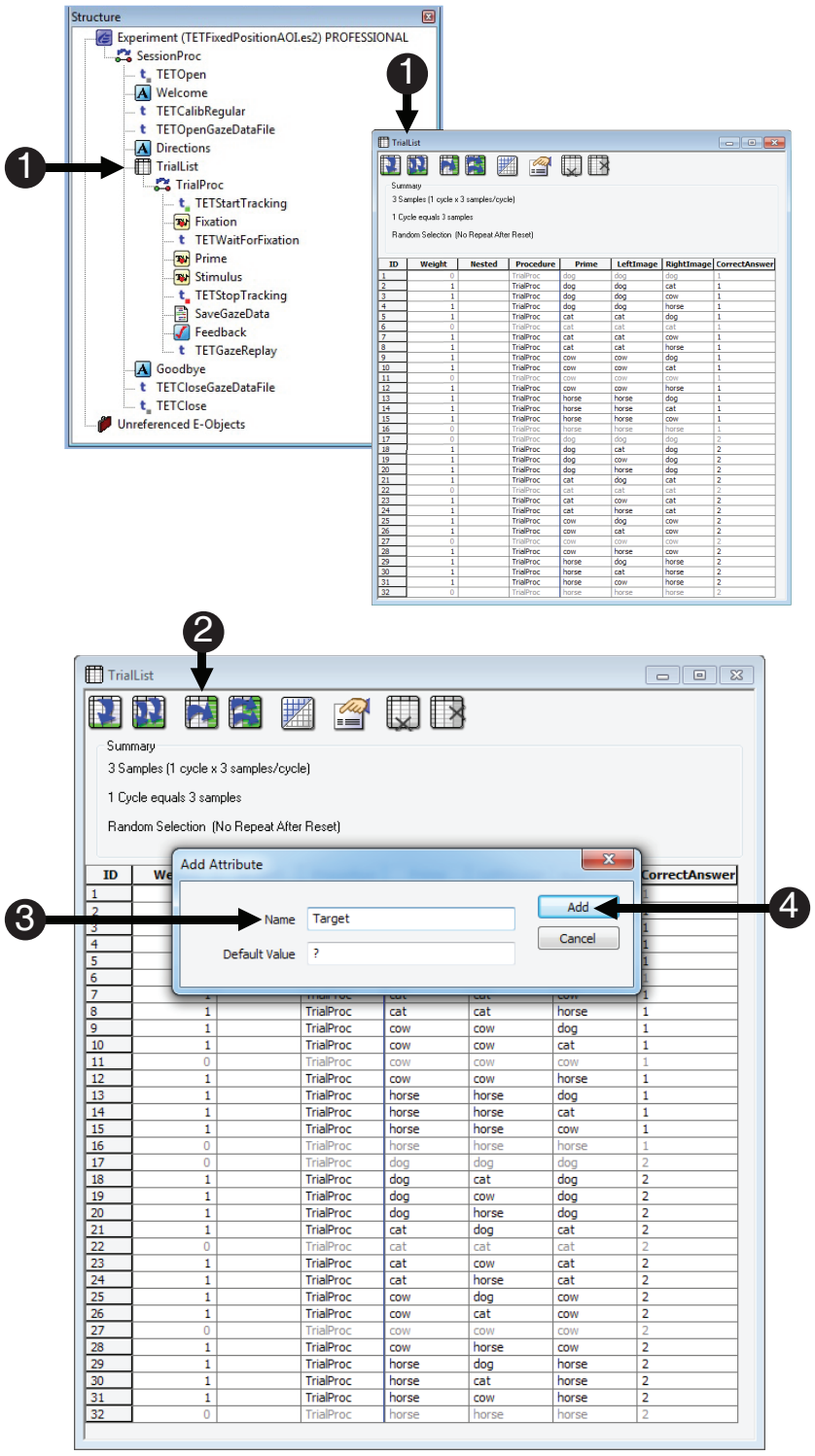
Currently, the TETFixedPositionAOIMarkers.es2 experiment does not explicitly encode the AOI for each trial (left or right). This information would be useful to have when analyzing the gaze data in Tobii Studio. In order to send this information to Tobii Studio, it must first be added to the experiment. In this step, we add an attribute to the TrialList and name it Target. The following task assigns a value to the attribute for each exemplar. Later in this tutorial, we send the value of the Target attribute to Tobii Studio.

1) ***Double click*** the **TrialList** to open it in the workspace.

2) ***Click*** the **Add Attribute** button.

3) ***Name*** the **Attribute** "**Target**."

4) ***Click*** the **Add** button to ***accept*** the changes.



Task 4: Edit the Target Attribute

Edit the Target Attribute to identify the critical AOI.

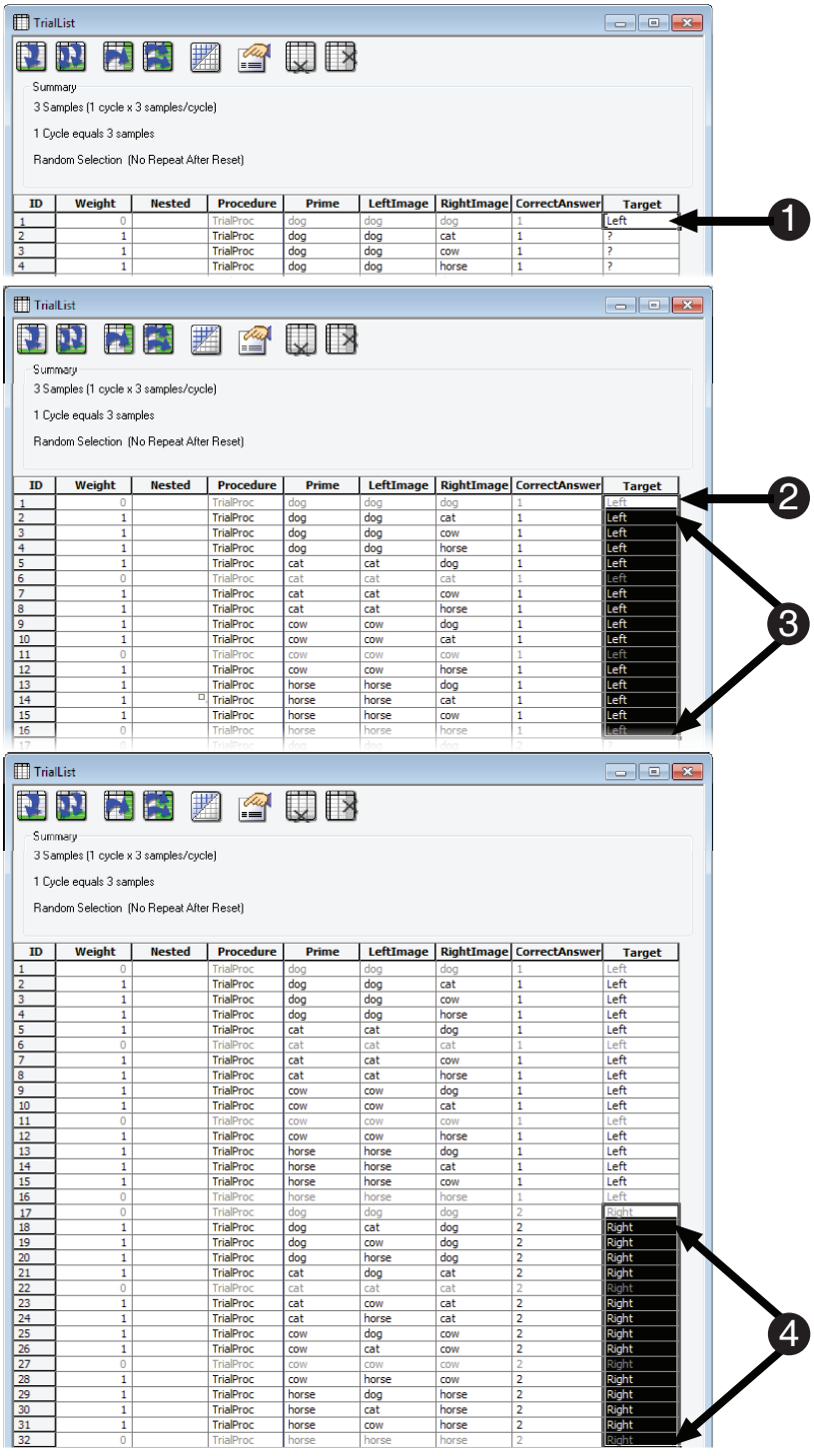
The Target Attribute will be used in Tobii Studio to identify where the target image was displayed relative to the fixation point. The default value assigned to the Target Attribute in Task 3 ("?") is changed in this task to either "Left" or "Right". Although the CorrectAnswer attribute could be used to infer the critical AOI, it does not encode this information in a descriptive manner. It is more useful to have the location of the target picture sent to Tobii Studio ("Left" or "Right") in order to group each condition into similar Scenes rather than sending the CorrectAnswer Attribute of "1" or "2".

1) ***Click*** in the **first cell** under "**Target**" to ***highlight*** the cell with a **question mark**. ***Type*** "**Left**" and ***hit*** **Enter** to accept the change.

2) ***Put*** the **cursor** in the **bottom right corner** of that **cell**, and ***move*** the **mouse** until the **cursor** ***becomes*** a **black cross hair**.

3) ***Click*** and ***drag*** the **black cross hair** ***down*** the **column** until it reaches the **number two** in the **CorrectAnswer** column (rows 1-16).

4) ***Use*** the **above steps** (1-3) to ***rename*** the **remaining cells**, "**Right**" (rows 17-32).



Task 5: Add the ClearView package file to the Experiment

Open the Property Pages for the Experiment Object and use the Packages tab to add the ClearView package file to the experiment.

As was illustrated in Tutorials 1 - 4, Package Files in E-Prime are cohesive sets of E-Basic routines that are grouped together into a single file that can be maintained externally. The routines that are used to communicate with the Tobii Studio software are contained in the ClearView (CV) package file. In order to gain access to these routines, you must first add the package file to the experiment, just as you added the TET package file in Tutorial 1.

⚠ **NOTE:** *As was noted in the introduction to Tutorial 5, this experiment is launched in parallel with a project that was created in Tobii Studio. Every project in Tobii Studio requires that calibration be performed to start the external video object. Since the E-Prime experiment already includes calls to the TET PackageFile Routine that performs calibration, you will end up performing calibration twice. If you wish to avoid the calibration step in E-Prime, highlight the TETCalibRegular PackageCall and press Delete. This will move the PackageCall to Unreferenced E-Objects, and allow the experiment to be run without first performing the extra calibration.*

1) ***Double click*** the **Experiment Object** at the top of the tree in the **Structure** view.

2) ***Click*** on the **Packages tab** of the **Experiment Object** Property Pages.

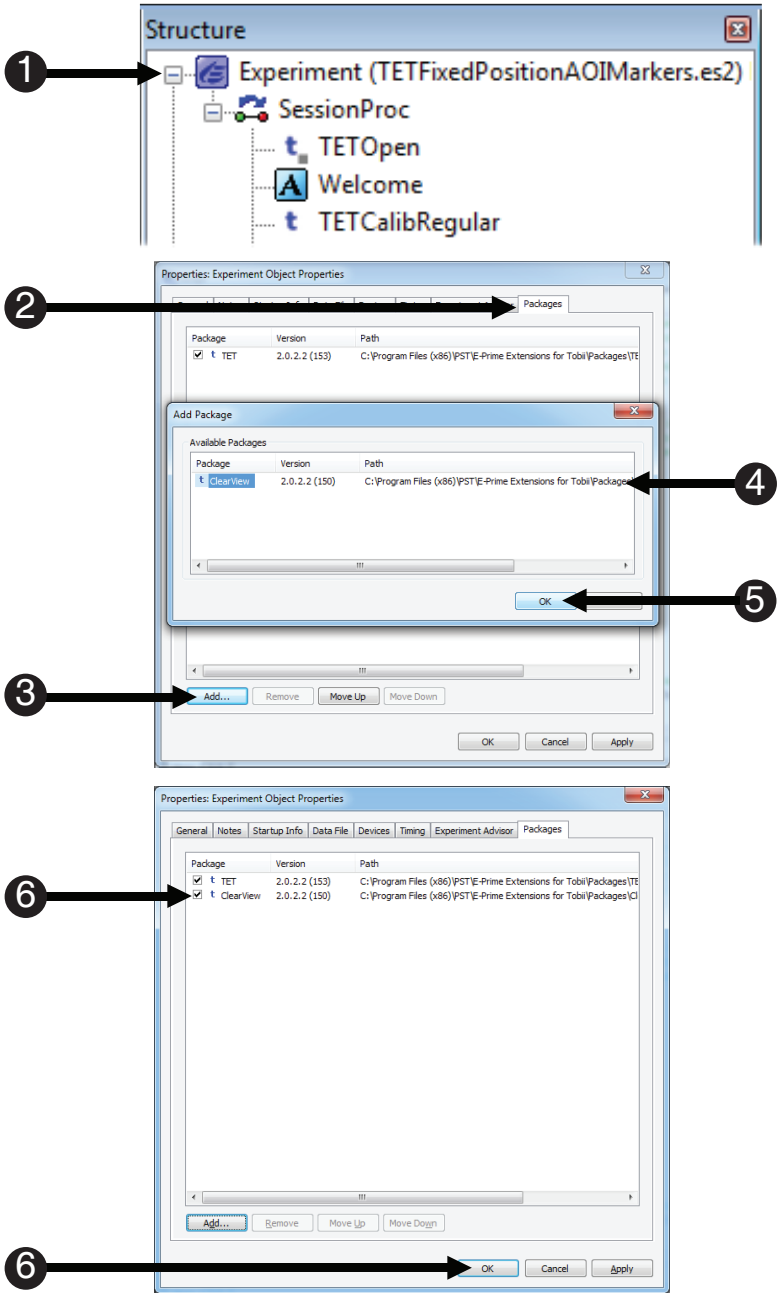
3) ***Click*** the **Add…** button.

4) ***Select*** the **ClearView Package File** in the Add Package dialog.

⚠ **NOTE:** *The version numbers may differ.*

5) ***Click*** the **OK** button to ***dismiss*** the **Add Package** dialog.

6) ***Verify*** the **ClearView package file** is listed under the **Package** column and is checked. Then ***click*** the **OK** button to **dismiss** the **Property Pages**.



Task 6: Add the ClearView Device to the Experiment Properties

Open the Property Pages for the Experiment Object and select the Devices tab to add the ClearView Device to the experiment.

Select the Devices Tab of the Experiment Object Property Pages to add the ClearView Device to the list of devices, and verify it is the last device shown.

1) ***Double click*** the **Experiment Object Properties** at the top of the tree in the **Structure** window.

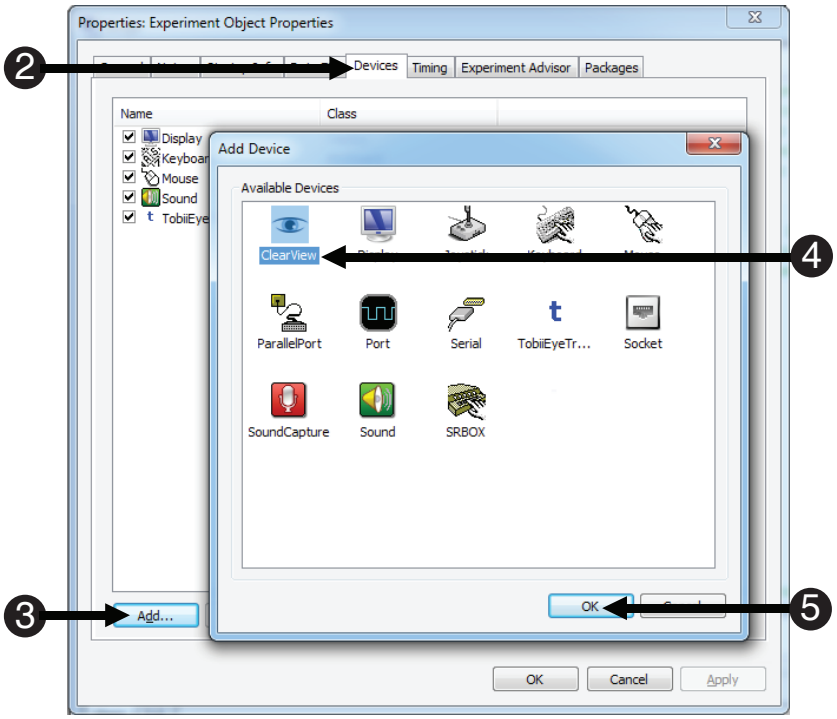
2) ***Click*** on the **Devices** tab of the **Experiment Object** Property Pages.

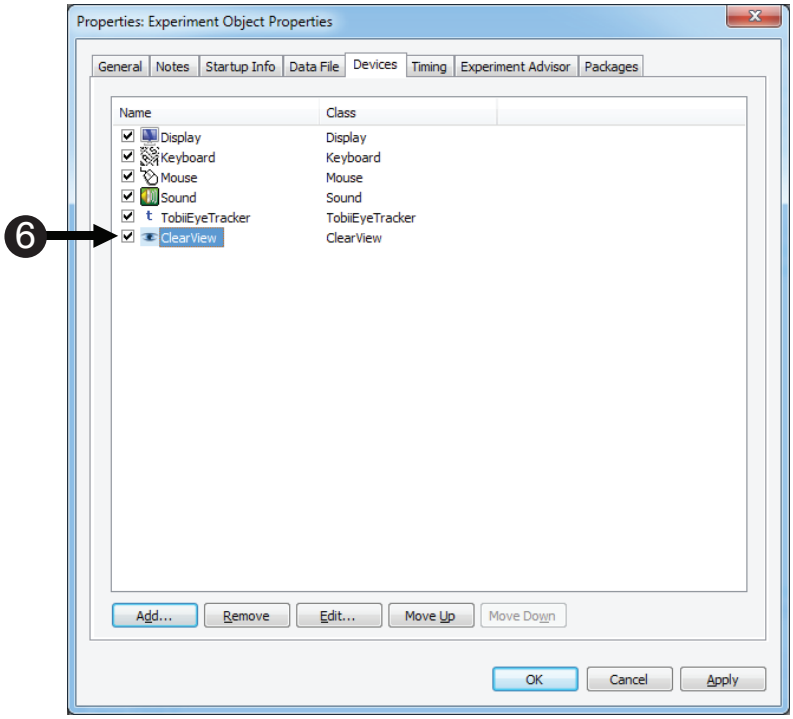
3) ***Click*** the **Add**… button.

4) ***Select*** the **ClearView Device** in the **Add Device dialog**.

5) ***Click*** the **OK** button to dismiss the **Add Device dialog**.

6) ***Verify*** the **ClearView device** is listed last under the **Name** column and is checked.





Task 7: Edit the ClearView Device Property Pages

Open the ClearView Device Property Pages, specify the Tobii Studio machine IP Address, and confirm the TCP/IP Port number.

Now that you have added the ClearView device, you must specify the IP address of the Tobii Studio machine so that E-Prime will be able to send Scene information to Tobii Studio.

1) ***Click*** the **ClearView** device to highlight it.

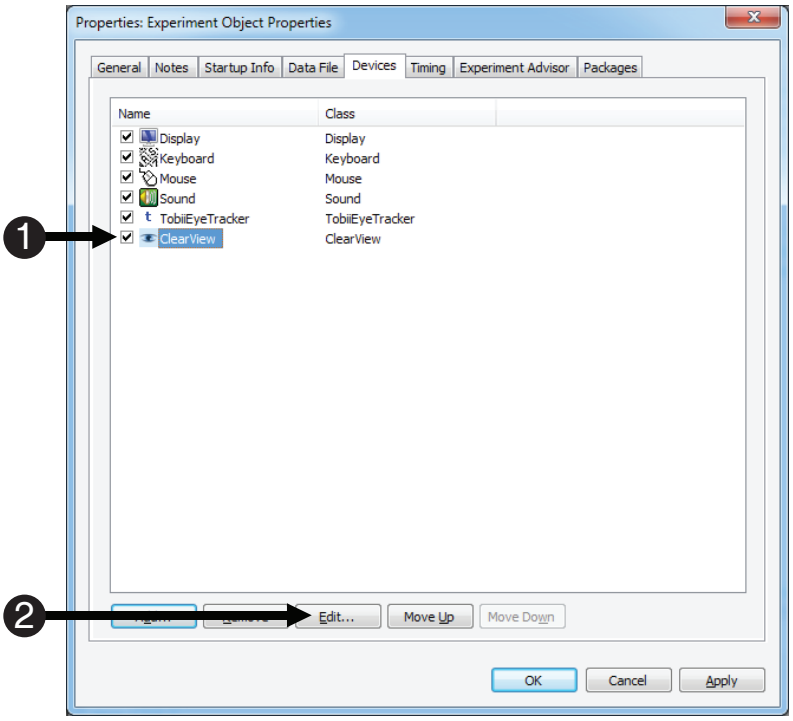
2) ***Click*** **Edit**.

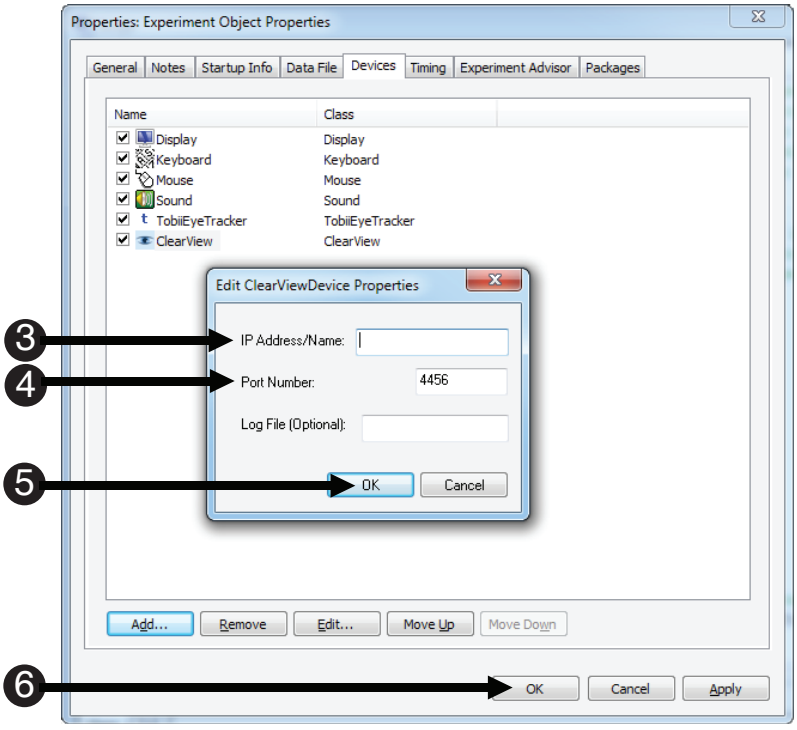
3) ***Type*** the **IP address** of the **Tobii Studio** machine (located on the back of the Eye Tracker).

4) ***Review*** the **TCP/IP Port Number**.

5) ***Click*** **OK** to accept **changes**.

6) ***Click*** the **OK** button to ***dismiss*** the **Add Device** dialog.





Task 8: Add the CVInit PackageCall to initialize the ClearView Package File

Add a PackageCall at the beginning of the SessionProc to initialize the system for use with ClearView Package File. Name the object "CVInit."

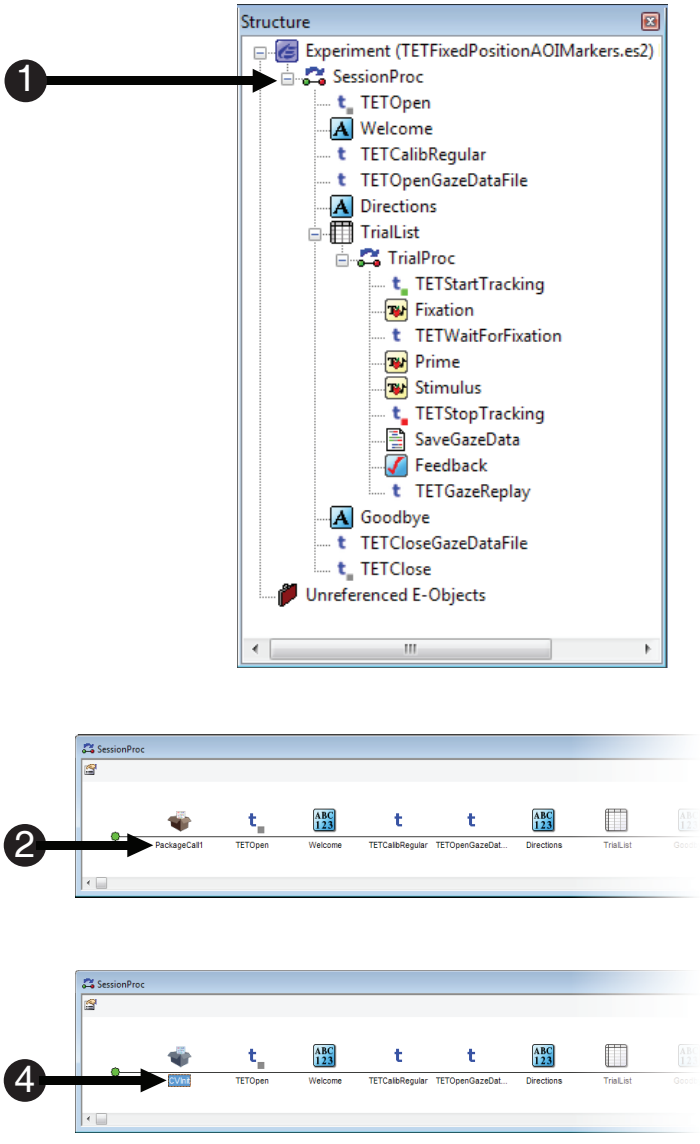
The ClearView Package File contains the routines that allow E-Prime to communicate with Tobii Studio, and must be initialized at the start of the experiment by making a call to the CV_Init routine. To call a routine in a Package File you can make the call directly using E-Basic script within an InLine object. However, the more preferable method is to drag a PackageCall from the E-Studio Toolbox, drop it at the desired location within the experiment, and edit its properties. When working with PackageCalls, we strongly recommend renaming the object to reflect the specific Package File and routine that is being referenced within the object. A common naming convention is to combine an identifier for the Package File (e.g., "CV"), with the name of the routine ("Init"). For example, the object used in this task to call the "CV_Init" routine is named "CVInit".

1) ***Double click*** the **SessionProc** to open it in the workspace.

2) ***Drag*** a new **PackageCall** from the **E-Studio Toolbox** and ***drop*** it as the **first** object on the **SessionProc**, right before **TETOpen**. The object will be given a default name of **PackageCall1**.

3) ***Click*** on the **PackageCall1** to ***select*** it and then ***press*** **F2** to rename the object.

4) ***Type*** "**CVInit**". ***Press*** **Enter** to accept the change.



Task 8 (continued): Add the CVInit PackageCall to initialize the ClearView Package File

Configure the CVInit PackageCall object to call the Init routine of the ClearView Package File.

The Properties dialog of the PackageCall is used to specify which Package File and Routine are to be called in each instance. After a Package is selected within the interface, the Routine dropdown list will be populated with all of the routines contained within the package. After you select a Routine, the Parameters field will be set to the default parameters and the Details field will be filled with the text that the Package File author included for the selected routine. You can refer to the Details field for information about each parameter in the list (any parameter in double quotes indicates string data). The CVInit PackageCall includes a parameter that allows you to turn communications with Tobii Studio on/off. The default is on.

5) ***Double click*** the **CVInit** PackageCall to ***display*** its Property Pages.

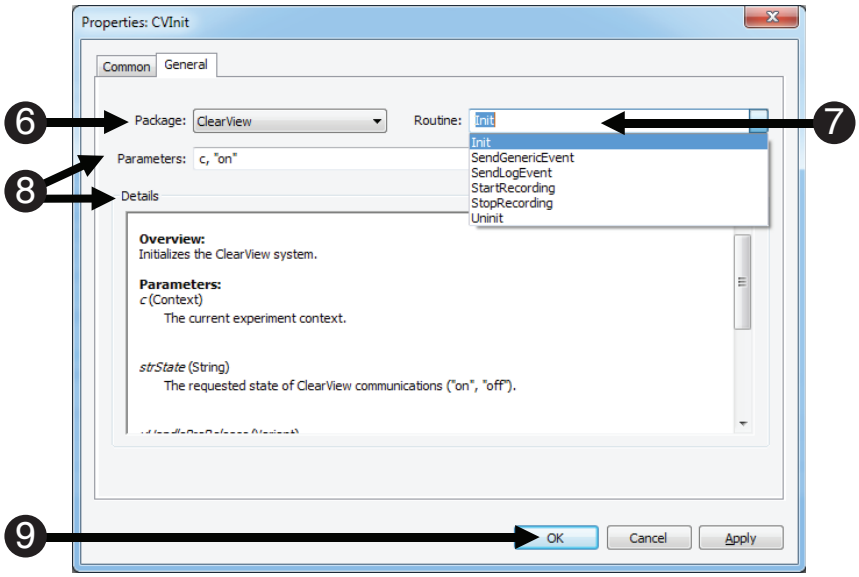
6) ***Select*** **ClearView** from the **Package** dropdown list.

7) ***Select*** **Init** from the **Routine** dropdown list.

8) ***Review*** the **CVInit** parameters listed in the **Parameters** and **Details** fields.

9) ***Click*** the **OK** button to ***accept*** the changes and ***dismiss*** the **Property Pages**.

10) ***Confirm*** your **SessionProc** is identical to the example shown.



Task 9: Add the CVStartRecording PackageCall

Add a PackageCall before the Welcome Object to initialize the system for use with ClearView Package File. Name the object CVStartRecording.

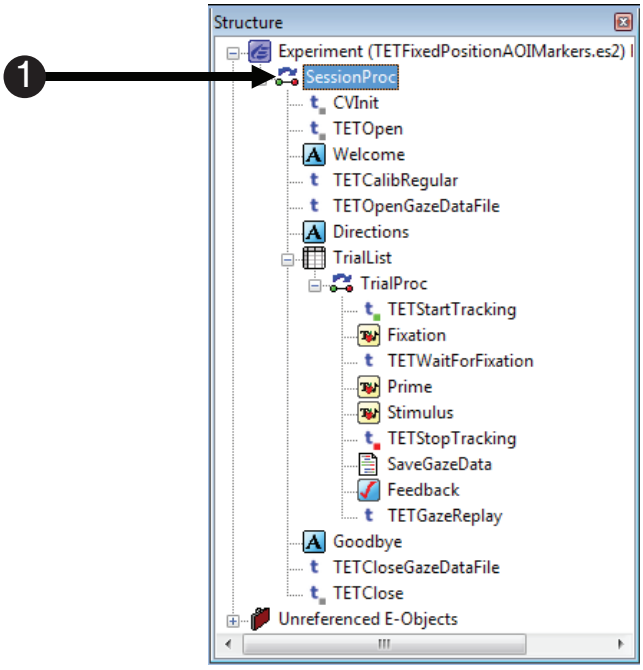
The CVStartRecording PackageCall tells the external video object in Tobii Studio to record the video signal that is sent from E-Prime to the capture card. This PackageCall should be placed at the location in the experiment where it is appropriate to begin recording data. In this experiment, the appropriate location is immediately before the Directions Text Object.

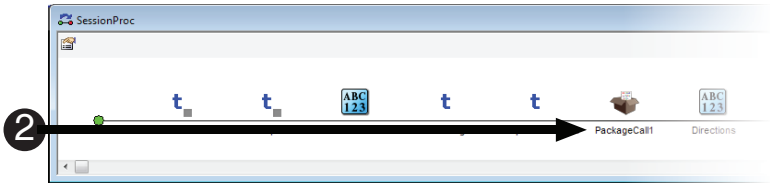
1) If the **SessionProc** is not already open, ***double click*** the **SessionProc** to **open** it in the workspace.

2) ***Drag*** a new ***PackageCall*** from the **E-Studio Toolbox** and ***drop*** it immediately ***before*** the **Directions object**. The **Object** will be given a default name of **PackageCall1**.

⚠ **NOTE:** *In order to include the Directions in the recording, the CVStartRecording PackageCall needs to be placed before the Direction object.*

3) ***Click*** on the **PackageCall1** to ***select*** it and then ***press*** **F2** to ***rename*** the **object**. ***Type*** "**CVStartRecording**" and then ***press*** **Enter** to ***accept*** the **change**.





Task 9 (continued): Add the CVStartRecording PackageCall

Configure the CVStartRecording PackageCall to the StartRecording routine of the ClearView Package File and accept the default Parameters list.

The CVStartRecording PackageCall only has one parameter, c. This parameter sets the current experiment context. Accept the default parameter and close the Property Pages dialog.

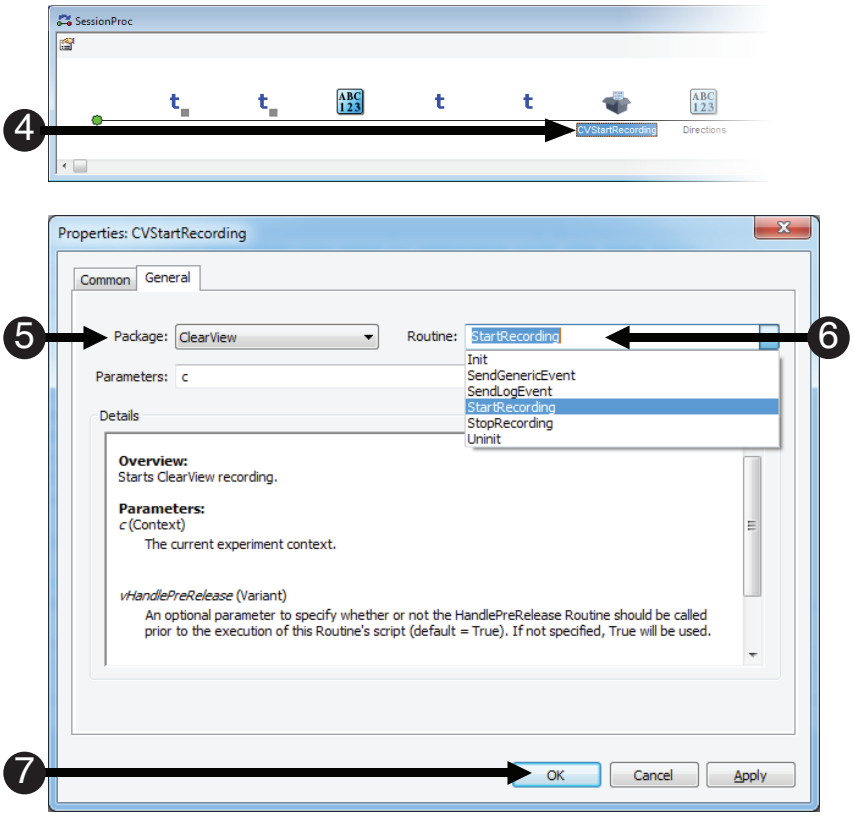
4) ***Double click*** the **CVStartRecording PackageCall** to ***display*** its **Property Pages**.

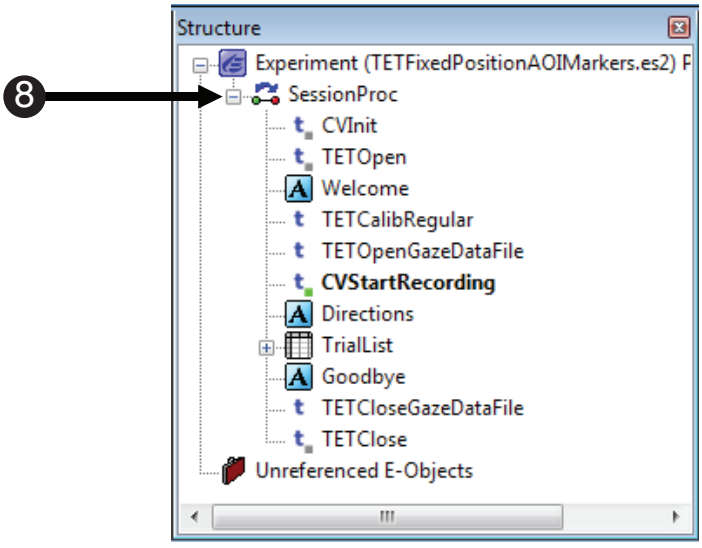
5) ***Select*** **ClearView** from the **Package** dropdown list.

6) ***Select*** **StartRecording** from the **Routine** dropdown list.

7) ***Click*** the **OK** button to ***accept*** the **changes** and ***dismiss*** the **Property Pages**.

8) ***Confirm*** your **SessionProc** is **identical** to the **example** shown here.





Task 10: Add a Task Events to the Fixation object

Define the "Event" component of the Task Event by specifying the Fixation.OnsetTime as the triggering event.

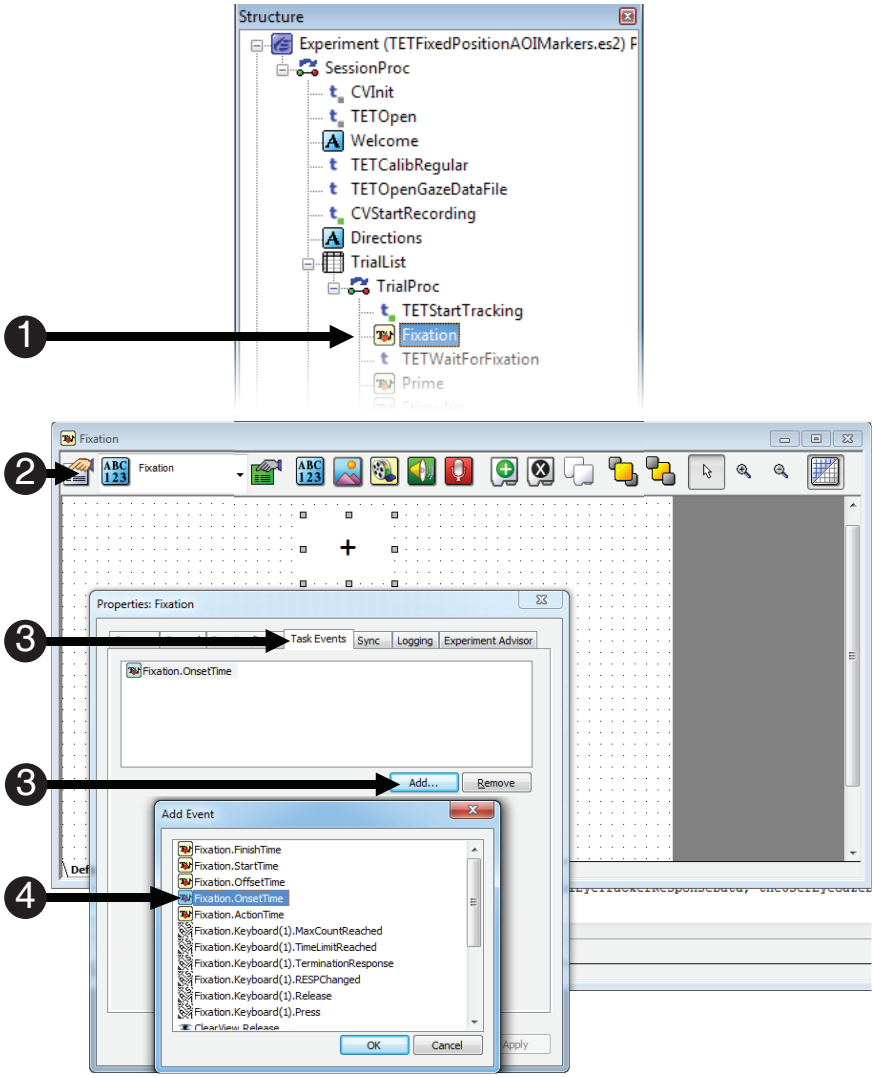
As described in the introduction to this section, Tutorial 5 uses E-Prime 2.0 Professional's Task Events feature to send Scene markers to Tobii Studio. Tobii Studio requires both a Scene Start and Scene Stop marker in order to perform Scene-based analyses of gaze data. Therefore, each Scene requires two related Task Events, one for Scene Start and another for Scene Stop. The first scene to be defined is the Fixation Scene. The Fixation Object's Onset Time, or the time when the Fixation object begins its action of being displayed, is used as the triggering event to mark the Scene Start. (As you will see later, this same event is used to mark the start of other scenes as well.) The trigger event to mark the stop of the Fixation Scene is the Prime Object's Onset Time. Using the Prime Object's Onset Time, rather than the Fixation Object's Offset Time, provides a more precise time stamp of when the Fixation Scene ends, because the Prime Object overwrites the information that is currently being displayed with the Fixation Object. The offset time for Fixation does not indicate when the Fixation object is no longer visible; it identifies when the Fixation Object is no longer executing. For more details about when information on the screen is updated, see **Section 4.5.3: Stimulus Presentation Solutions** in the *E-Prime 2.0 User's Guide*. Note that the CVSendLogEvent PackageCall could also have been used to send Scene marker information to Tobii Studio. However, we recommend using the Task Events method because it intrinsically accounts for issues that would otherwise need to be handled with the coordination of PackageCall placement and other timing parameters.

1) ***Double click*** the **Fixation object** to open it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Fixation.OnsetTime**.



Task 11: Edit the Fixation.StartOnsetTime Task Event

Define the "Task" component of the Task Event by specifying the Fixation SceneStart as the information to be sent to Tobii Studio.

The next step is to configure the Task Event to send the Scene Start information to Tobii Studio. In this case, we want to send a string that identifies the start of the Fixation Scene when the Fixation object begins to display.

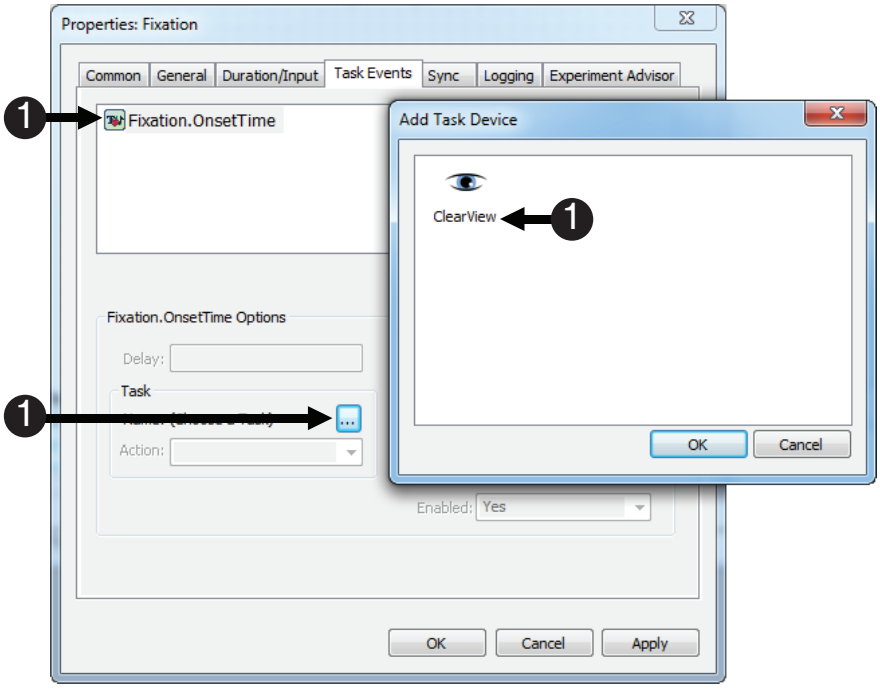
1) ***Select*** the **Fixation.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

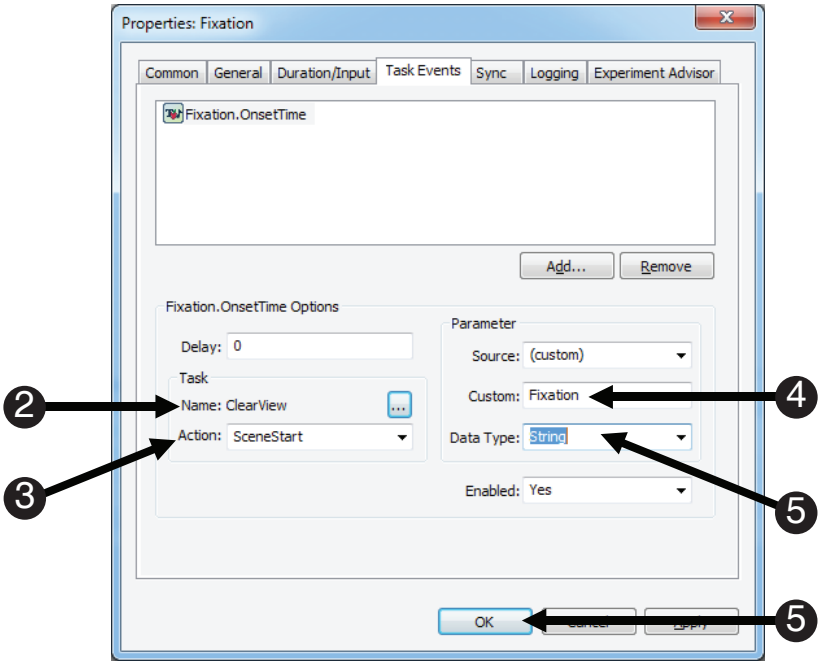
2) ***Note*** the **Name** field ***reads*** **ClearView**, indicating that the **ClearView device** will ***receive*** the **Task information**.

3) ***Select*** **SceneStart** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **Fixation**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 12: Add another Task Event to the Fixation object

Define another Task Event based on the Fixation Object's OnsetTime; this will mark the start of the Trial Scene.

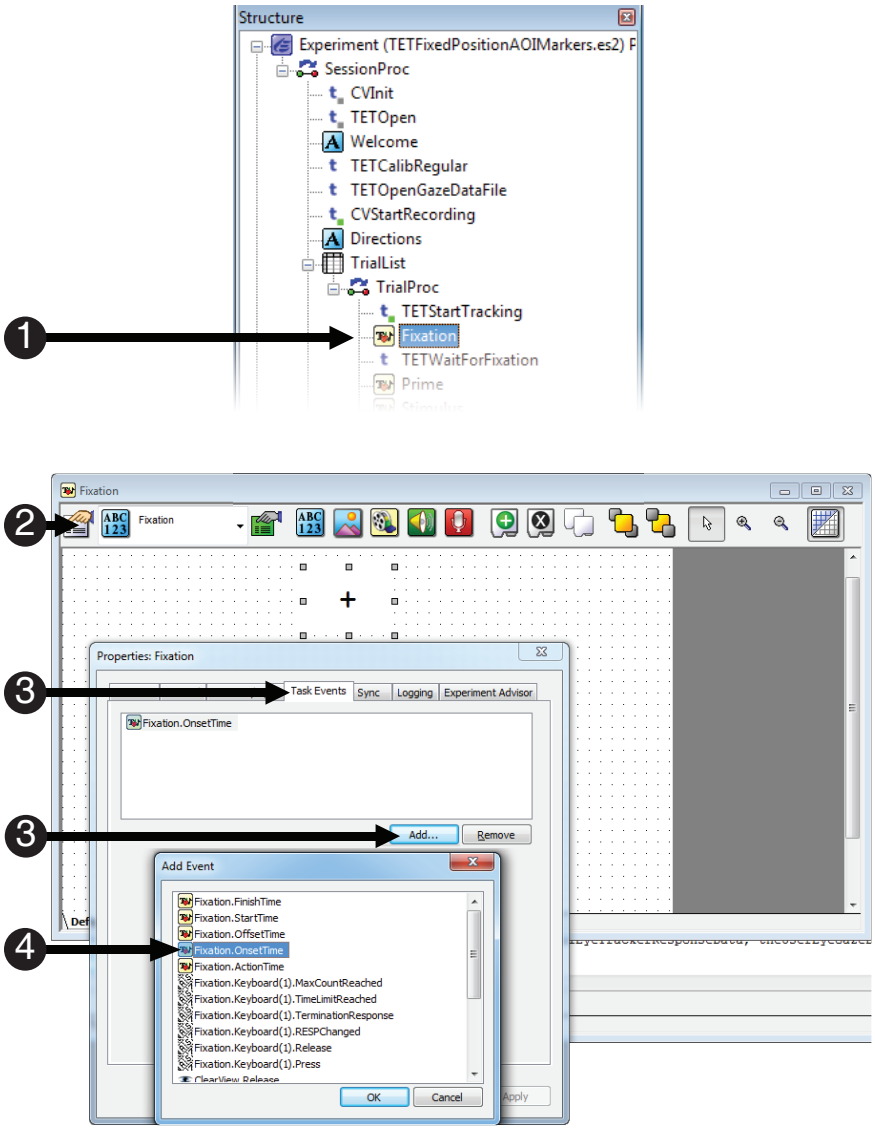
A second Task Event, which also uses the Fixation.OnsetTime as the triggering event, is added to the Fixation Object. This second Task Event is used to identify the start of the Trial Scene. Here we create a trial scene for each target type ("Left" or "Right"), using the Target Attribute that was created in Task 3 of this tutorial. Therefore, we have two scenes that start when the Fixation object beings to display. These scenes will end, however, with different events on different objects, as you will see shortly.

1) ***Double click*** the **Fixation object** to open it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Fixation.OnsetTime**.



Task 12 (continued): Add another Task Event to the Fixation

Define another Task Event based on the Fixation Object's OnsetTime; this will mark the start of the Trial scene.

The Target attribute becomes part of the Scene marker that is sent to Tobii Studio. When the Target attribute equals "Left", then the Scene marker will equal "Trial:Left"; when the Target attribute equals "Right", then the Scene marker will equal "Trial:Right".

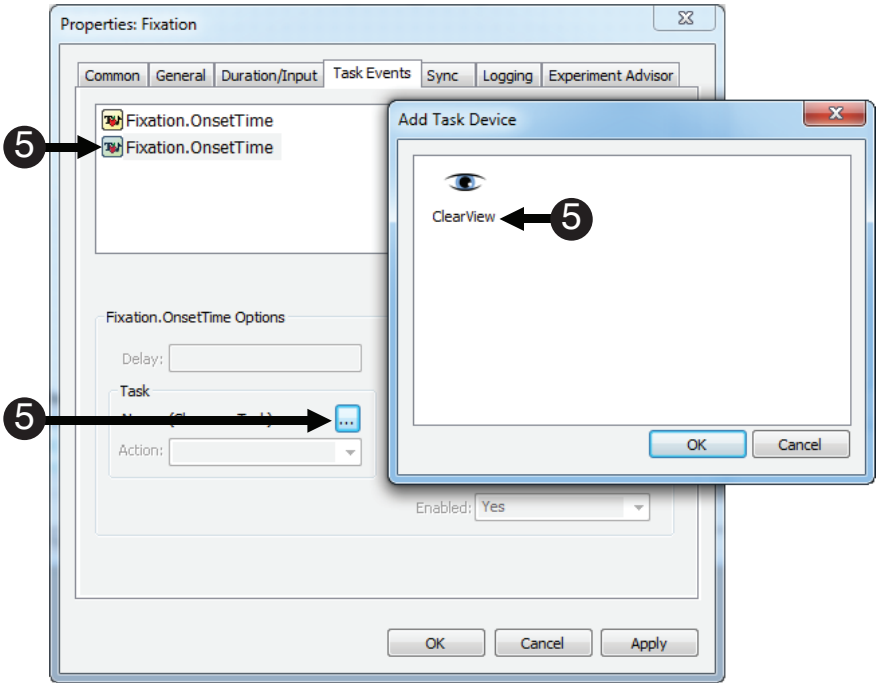
5) ***Select*** the second **Fixation.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon to ***select*** **ClearView** as the **Task** device.

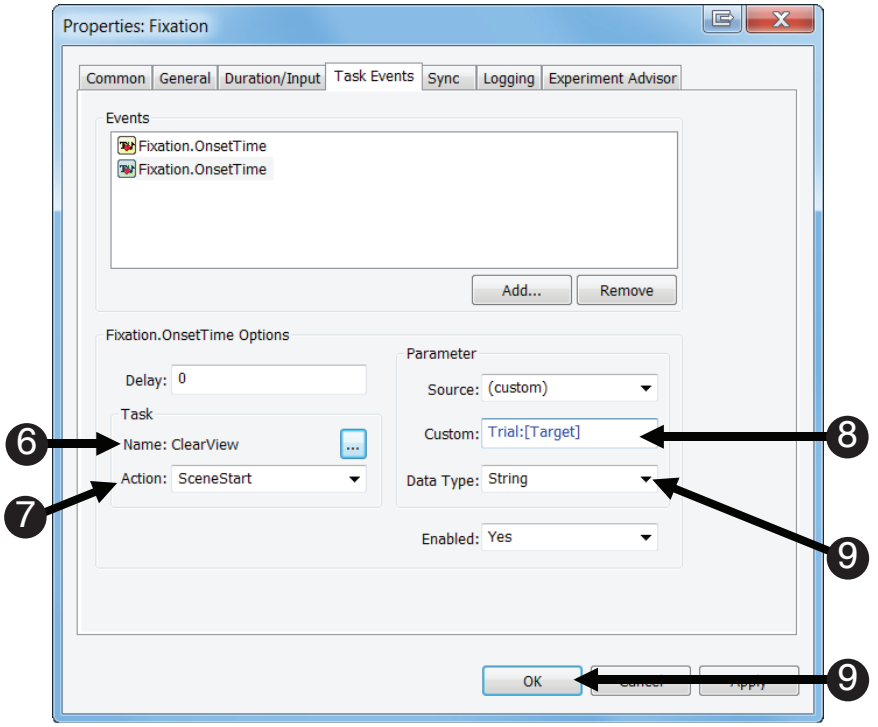
6) ***Note*** the **Name** field ***reads*** **ClearView**.

7) ***Select*** **SceneStart** from the **Action** field.

8) ***Edit*** the **Custom** file to ***read***: **Trial:[Target]**.

9) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 13: Add a Fixation SceneStop Task Event to the Prime Object

Define the "Event" component of the Task Event by specifying the Prime.OnsetTime as the triggering event.

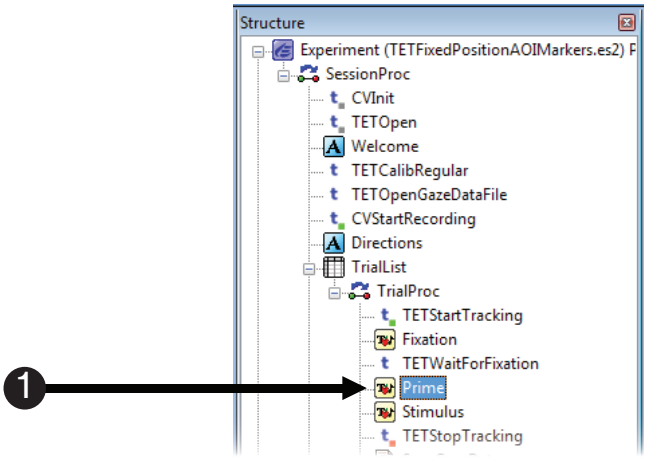
Here we define the end of the Fixation scene. The Prime Object's OnsetTime is the triggering event to mark the end of the Fixation scene that was started in Task 11 of this Tutorial.

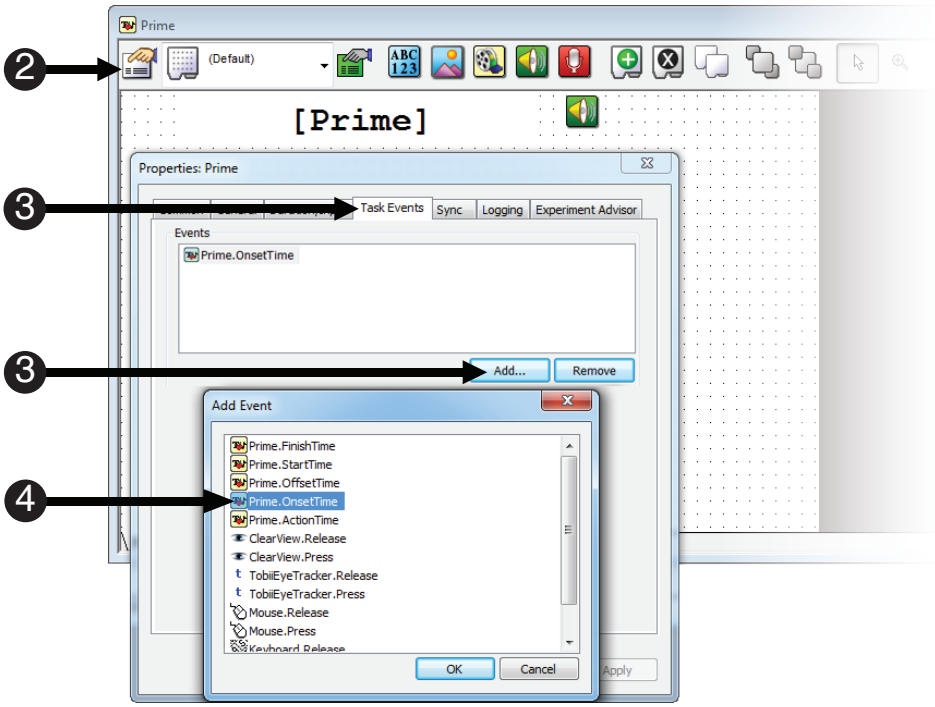
1) ***Double click*** the **Prime Object** to ***open*** it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Prime.OnsetTime**.





Task 14: Edit the Fixation SceneStop Task Event

Define the "Task" component of the PrimeObject Task Event by specifying the Fixation SceneStop marker as the information to be sent to Tobii Studio.

This Task Event defines the end of the Fixation scene. Therefore, we send the SceneStop marker for the Fixation scene to Tobii Studio when the Prime Object begins to execute.

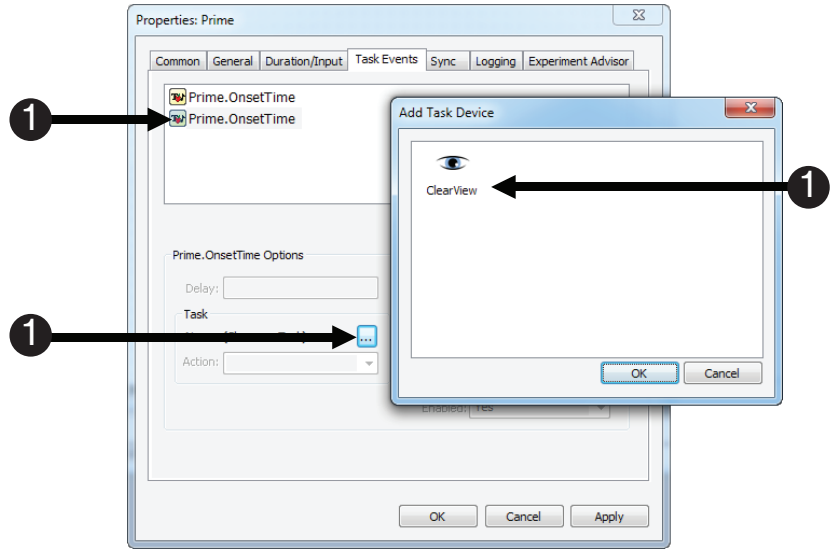
1) ***Select*** the second **Prime.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

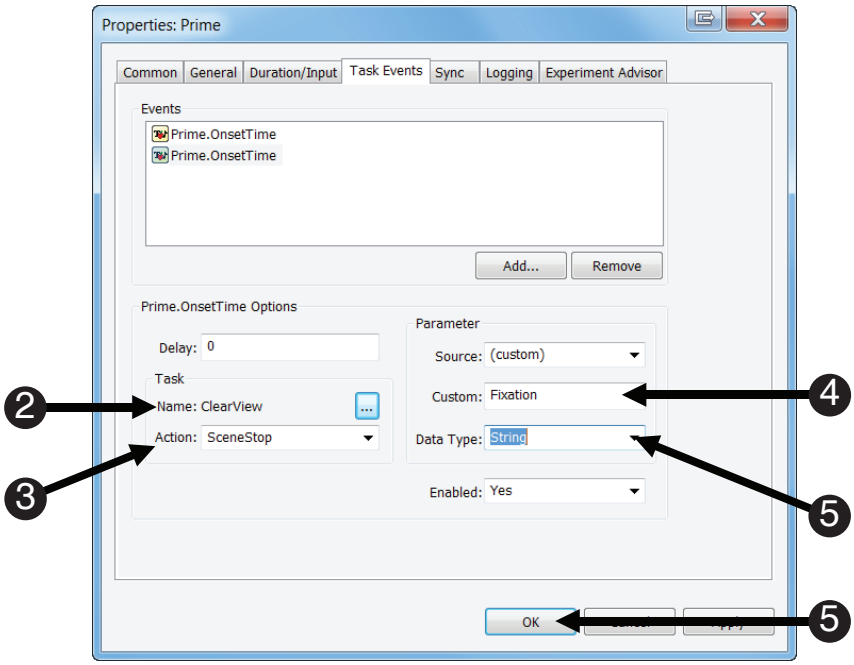
2) ***Note*** the **Name** field ***reads*** **ClearView.**

3) ***Select*** **SceneStop** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **Fixation**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 15: Create the Prime Scene by adding another Task Event to the Prime Object

Define the "Event" component of a new Task Event for the Prime Object by specifying the Prime.OnsetTime as the triggering event for the Prime scene.

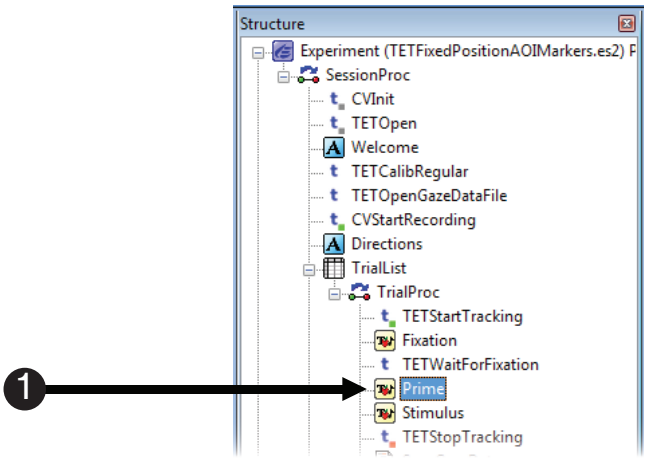
This step will add another .OnsetTime task event to the Prime Object. This task event will send the onset time of the Prime Object from E-Prime to Tobii Studio. This Task Event sends the SceneStart marker for the Prime scene when the Prime Object begins to execute. As you will see later, the SceneStop marker for the Prime scene will be sent when the Stimulus Object begins to execute.

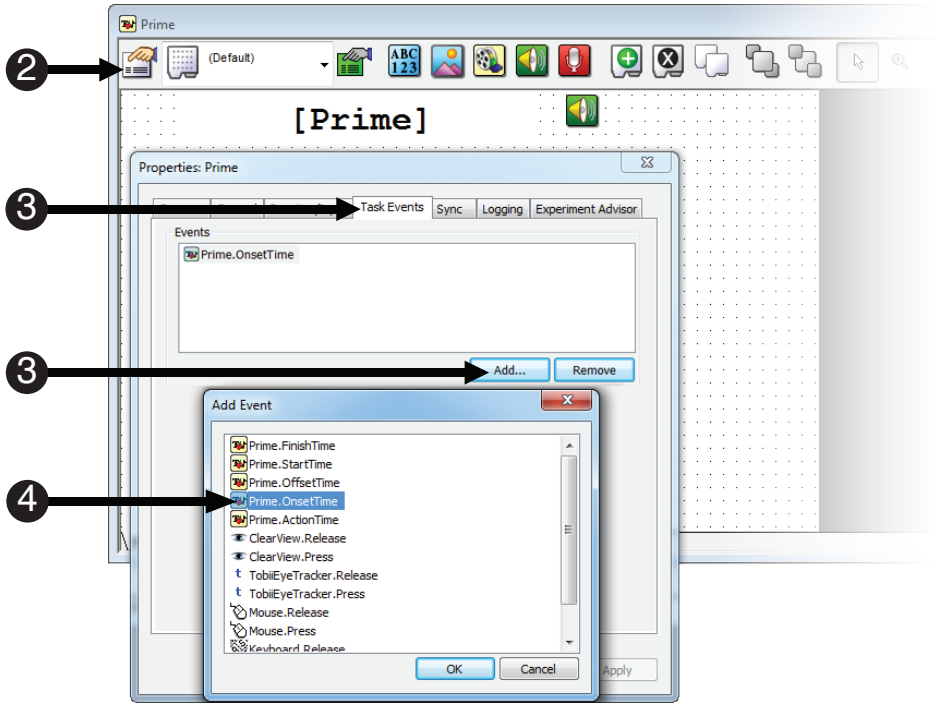
1) If necessary, ***double click*** the **Prime** to ***open*** it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Prime.OnsetTime**.





Task 16: Edit the Prime Scene Task Event

Define the "Task "component of the Task Event by specifying the Prime SceneStart marker as the information to be sent to Tobii Studio.

The next step is to edit the Prime.OnsetTime Task Event so E-Prime knows what information to send to Tobii Studio. In this case we want to send an Attribute identifying the Prime, so we will edit the Task Event parameters to do so. Recall that the Prime attribute is assigned to the Prime word that appears on each trial. Therefore, sending the Prime attribute will result in scenes that are labeled "dog", "cat", "cow", or "horse".

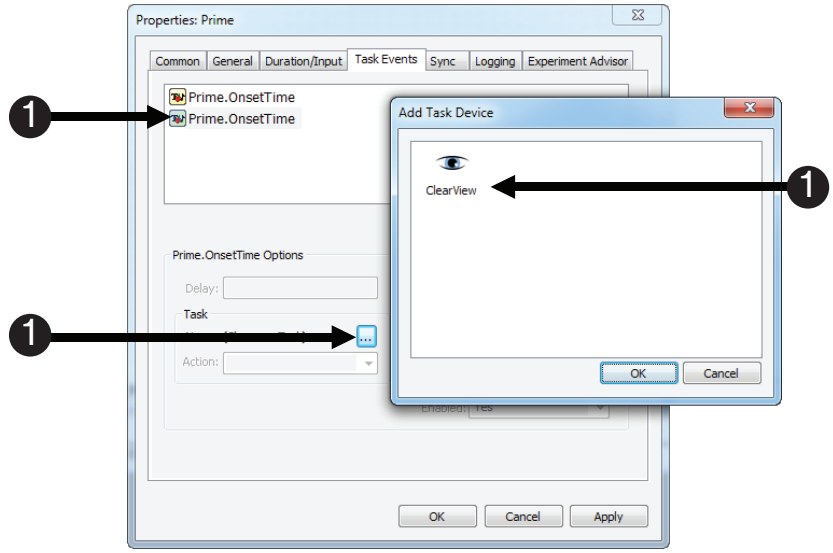
1) ***Select*** the second **Prime.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

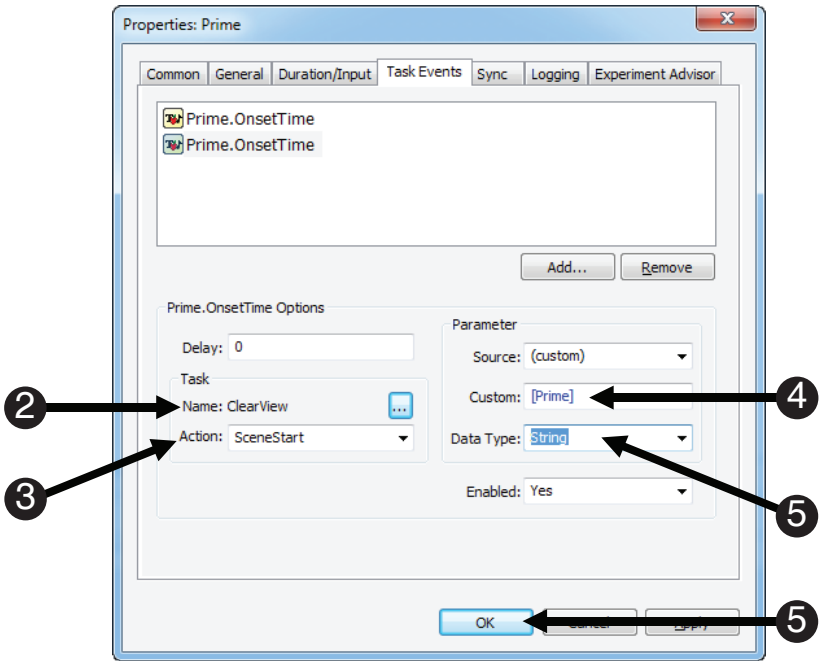
2) ***Note*** the **Name** field ***reads*** **ClearView.**

3) ***Select*** **SceneStart** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **[Prime]**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 17: Add the Prime SceneStop Task Event to the Stimulus Object

Define the "Event" component of a new Task Event by specifying the Stimulus.OnsetTime the triggering event.

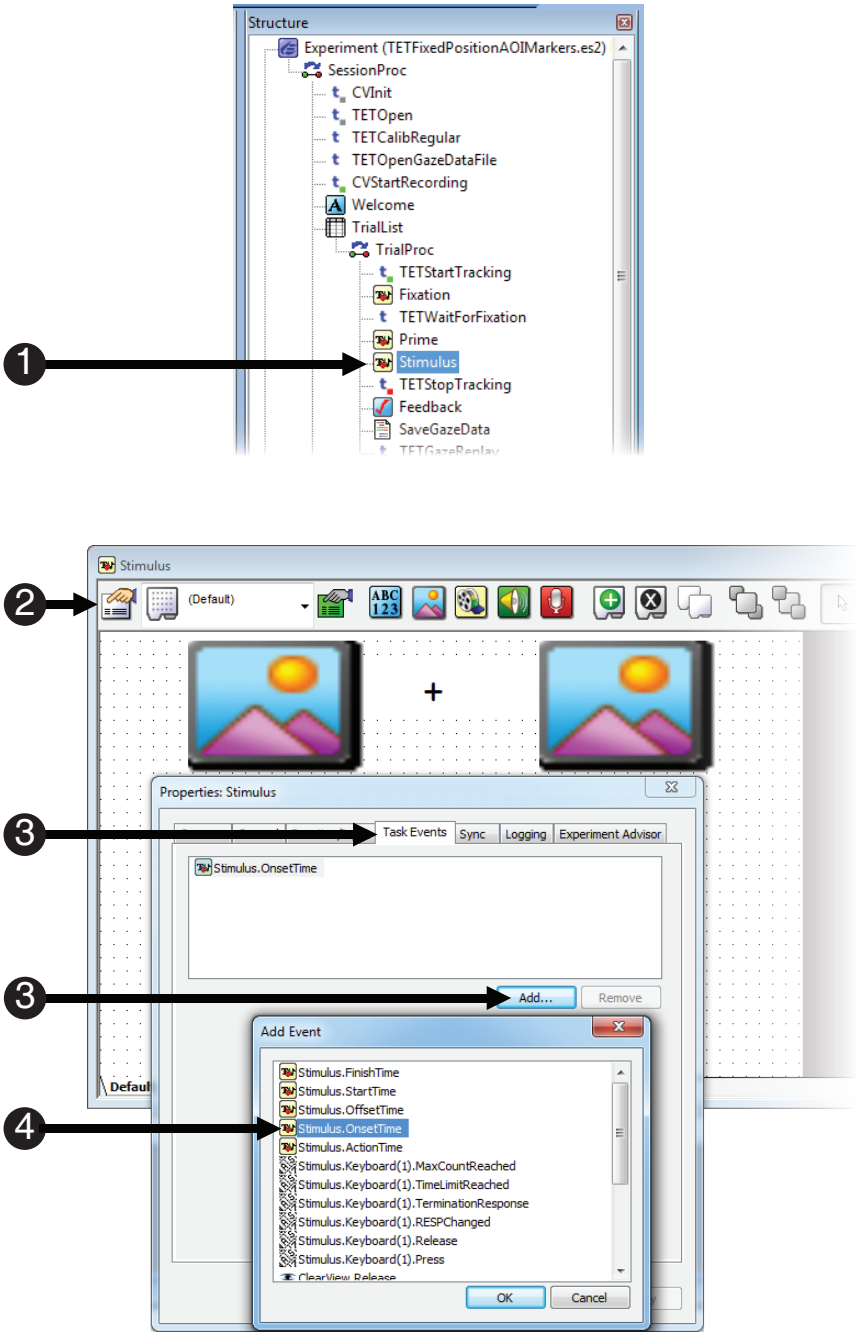
This Task Event sends the Prime SceneStop signal at the onset time of the Stimulus Object to Tobii Studio. This ensures that the Prime scene in Tobii Studio has a starting and ending point. Like the last SceneStop that we sent for the Fixation scene, the SceneStop is triggered during the onset of the next display object in the trial procedure (Stimulus) instead of the offset of the object, which marked the start of the scene (Prime).

1) ***Double click*** the **Stimulus** to ***open*** it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Stimulus.OnsetTime**.



Task 18: Edit the Stimulus.OnsetTime Task Event

Define the "Task" component of the Task Event by specifying the Prime SceneStop marker as the information to be sent to Tobii Studio.

This Task Event defines the end of the Prime scene. Therefore, we send the value of the [Prime] attribute to Tobii Studio. As was done earlier with the Fixation scene, sending both a SceneStart and SceneStop marker is required in order for Tobii Studio to support a scene-based analysis.

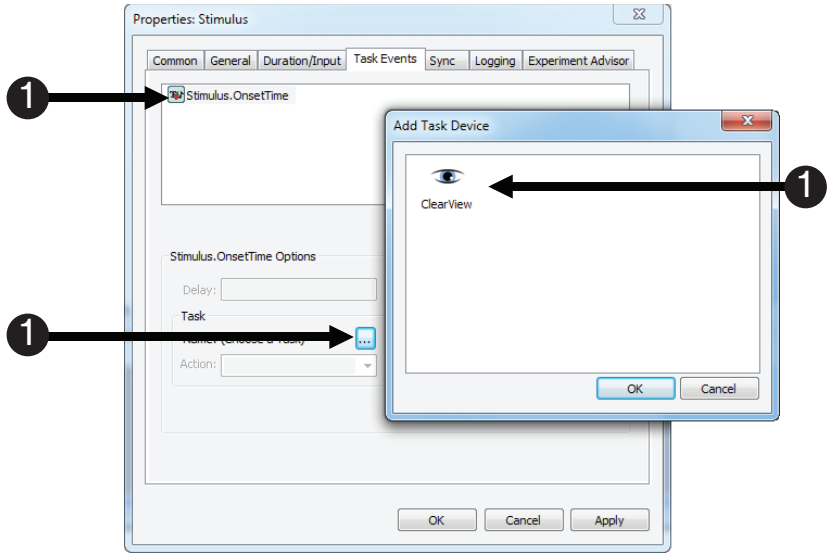
1) ***Select*** the **Stimulus.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

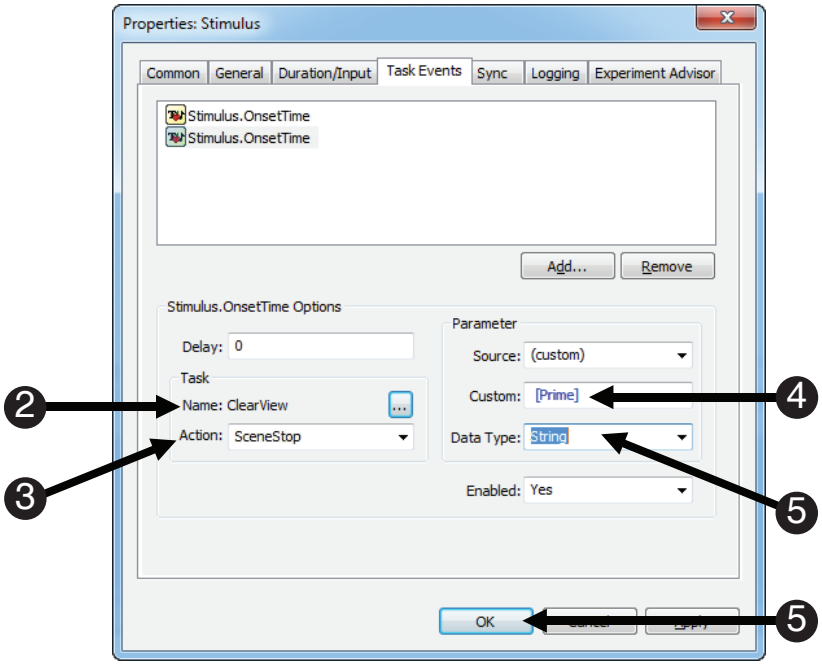
2) ***Note*** the **Name** field ***reads*** **ClearView.**

3) ***Select*** **SceneStop** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **[Prime]**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 19: Create multiple new SceneStarts with one Task Event on the Stimulus Object

Define the "Event" component of a new Task Event that will be used to create multiple scenes.

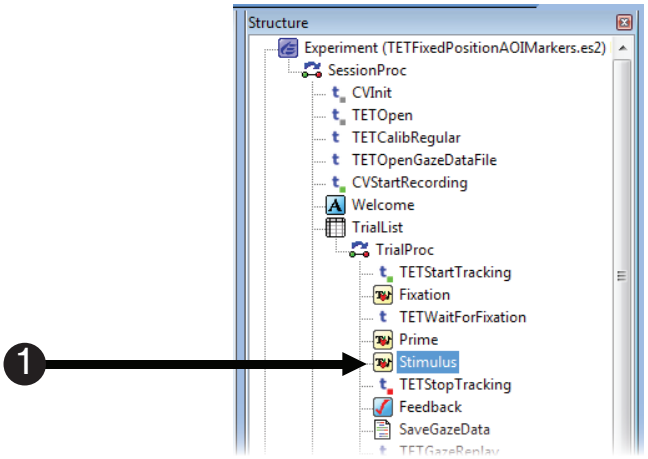
Unlike the last few Task Events that have been created, the Task Event created below utilizes a special feature of E-Prime Extensions for Tobii, which enables the start of multiple scenes to occur within a single Task Event.

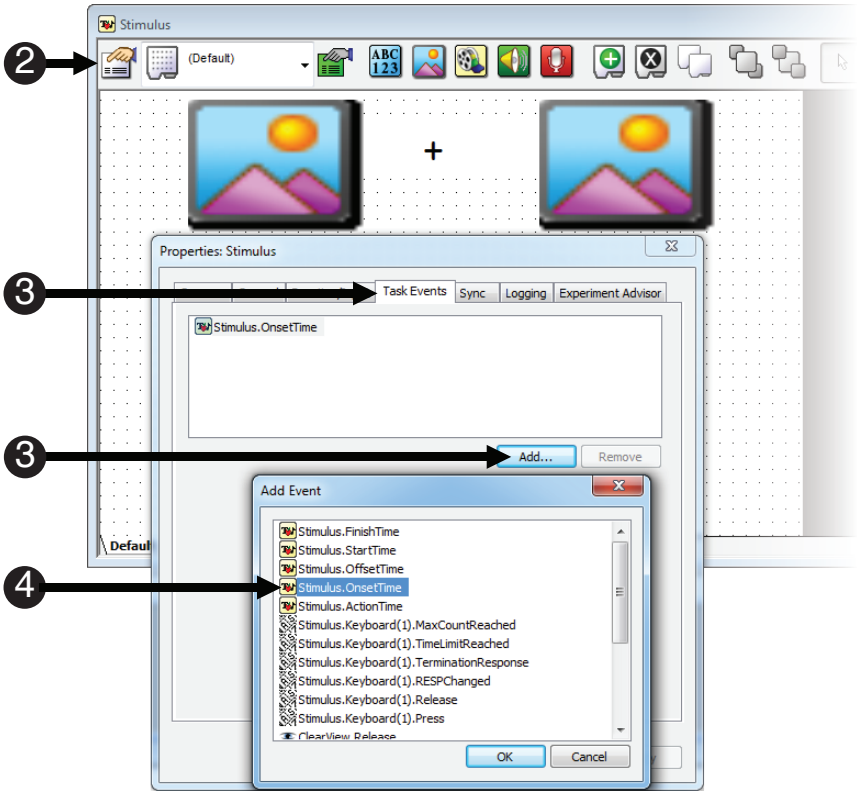
1) If necessary, ***double click*** the **Stimulus** to ***open*** it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Stimulus.OnsetTime**.





Task 20: Edit the Stimulus.OnsetTime Task Event

Define the "Task" component of the event as sending multiple SceneStart markers to Tobii Studio.

The next step is to define the Task component for the Stimulus Object Event that was created in the previous task. In this case, we want to perform multiple Tasks, i.e. send multiple SceneStart markers, for a single Event. A special feature of E-Prime Extensions for Tobii is the use of the pipe character (|) as a delimiter to specify multiple values in the Custom field. Separating the names of each Scene marker enables E-Prime to send multiple SceneStart triggers to Tobii Studio with only one Task Event. The first scene marker to be sent is the Stimulus Scene. This will allow you to conduct scene-based analyses on the entire Stimulus Slide object. The next two scenes describe the right and left images, allowing scenes with similar stimuli to be analyzed together. Finally, we create a scene called RESP that focuses on the period of time just before the participant makes a response.

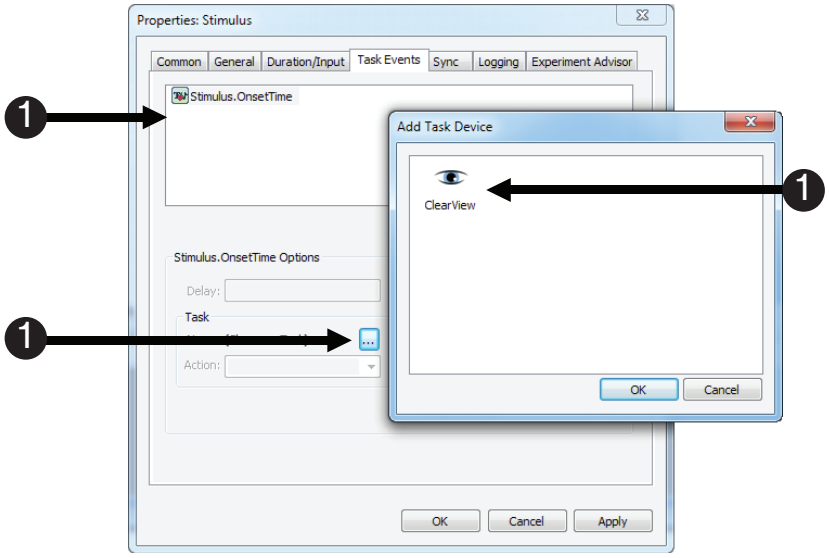
1) ***Select*** the second **Stimulus.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

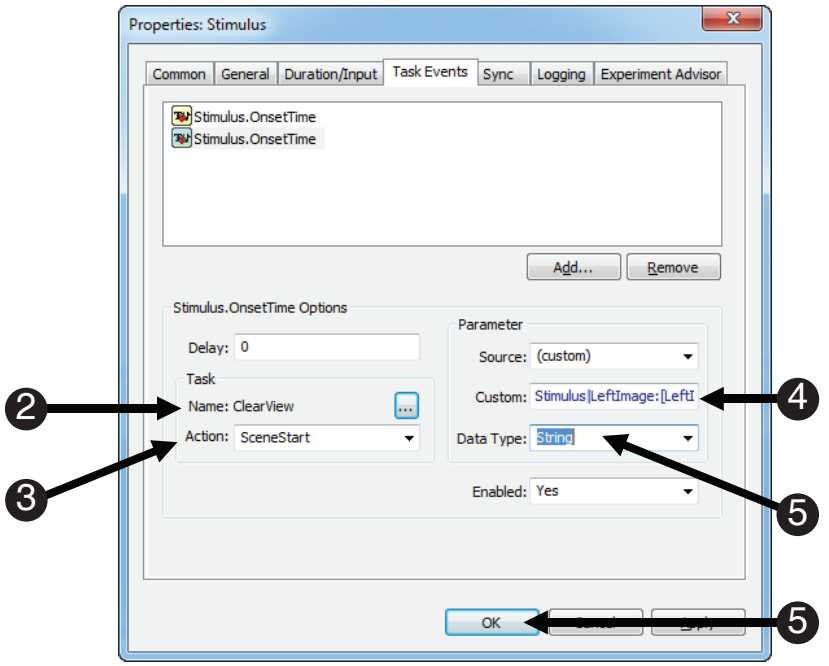
2) ***Note*** the **Name** field ***reads*** **ClearView.**

3) ***Select*** **SceneStart** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **Stimulus|LeftImage:[LeftImage]|RightImage:[RightImage]|RESP**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 21: Add another Task Event to the Stimulus Object

Define the "Event" component as the Stimulus.Keyboard(1).MaxCountReached; this event will trigger several important data-related scenes.

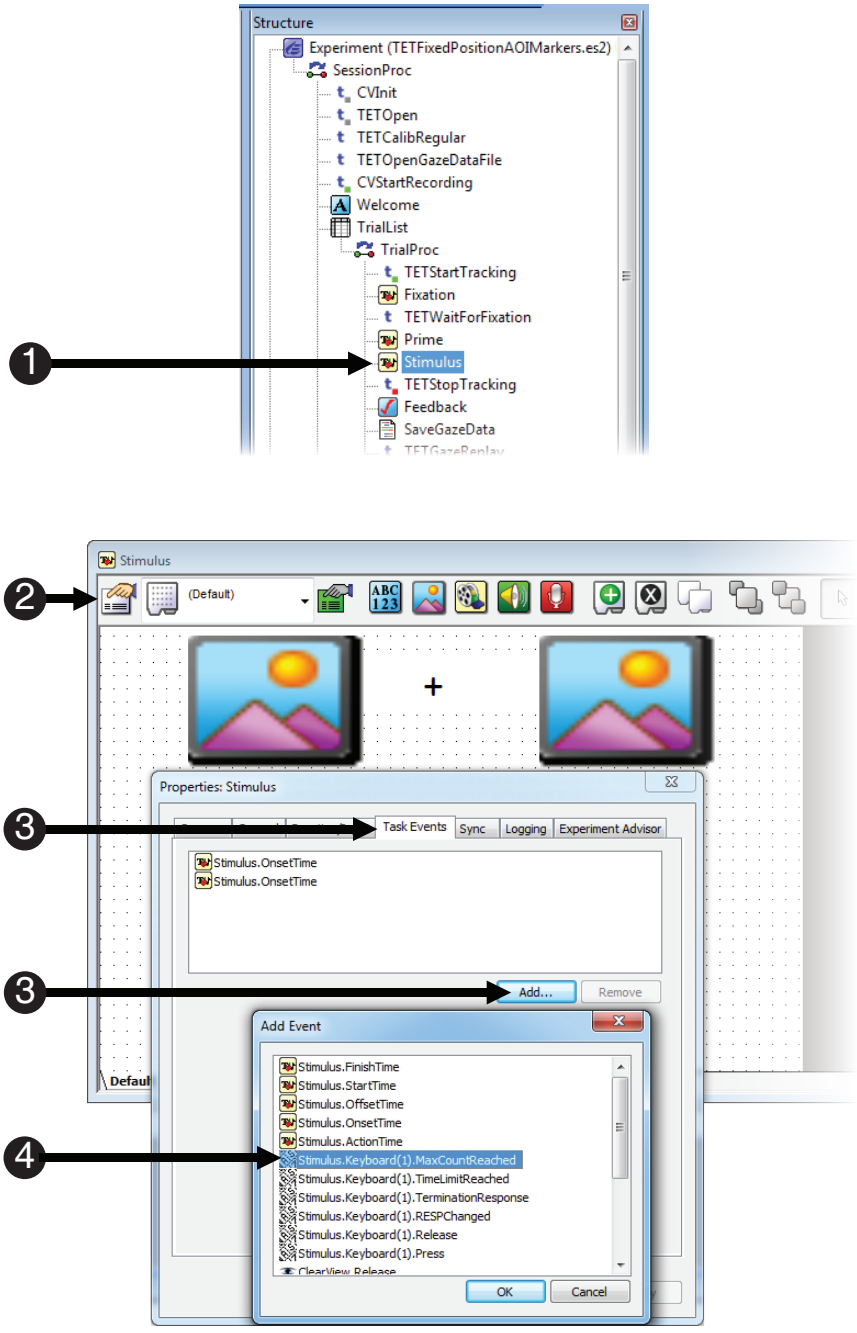
The purpose of this Task Event is to define the end of the RESP scene in Tobii Studio. We want to end the RESP scene when the Input Mask is terminated for the Stimulus Object. This can occur in one of two ways – either a response is made, or the object's display duration is reached. This Task Event identifies the former, when the maximum number of responses has been made to the Stimulus Object. (In this case, the maximum number of responses that can be made to the Stimulus is one; e.g., a single keypress terminates the InputMask). This event is triggered with Stimulus.Keyboard(1). MaxCountReached. The trigger that comes across when the max count is reached will accurately reflect when a response was made.

1) If necessary, ***double click*** the **Stimulus** to ***open*** it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Stimulus. Keyboard(1).MaxCountReached**.



Task 22: Edit the Task associated with the Stimulus.Keyboard(1).MaxCountReached Event

Define the "Task" component of the Task Event as sending the RESP SceneStop marker to Tobii Studio.

This Task Event will ensure that the RESP scene has a SceneStop marker when the participant responds. We will also add a Task Event to the experiment that handles the instances when a participant does not respond in the next two Tutorial Tasks.

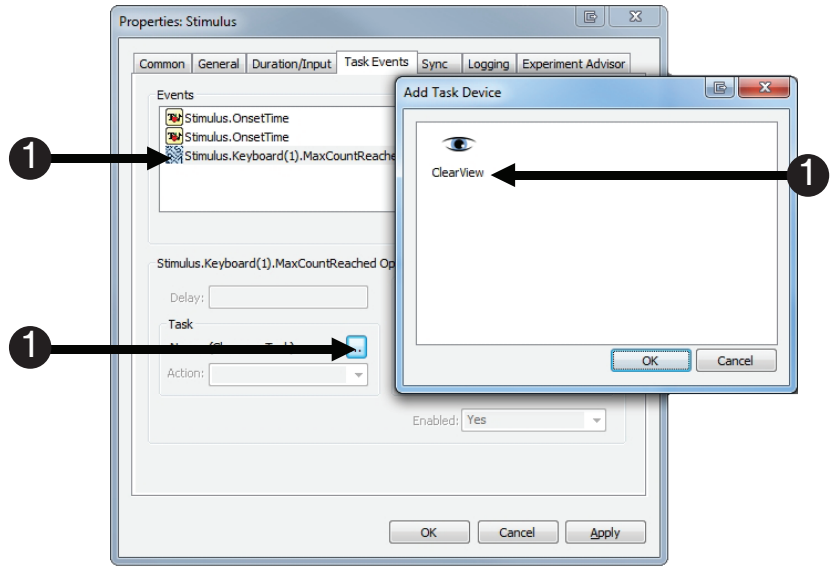
1) ***Select*** the **Stimulus.Keyboard(1).MaxCountReached** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

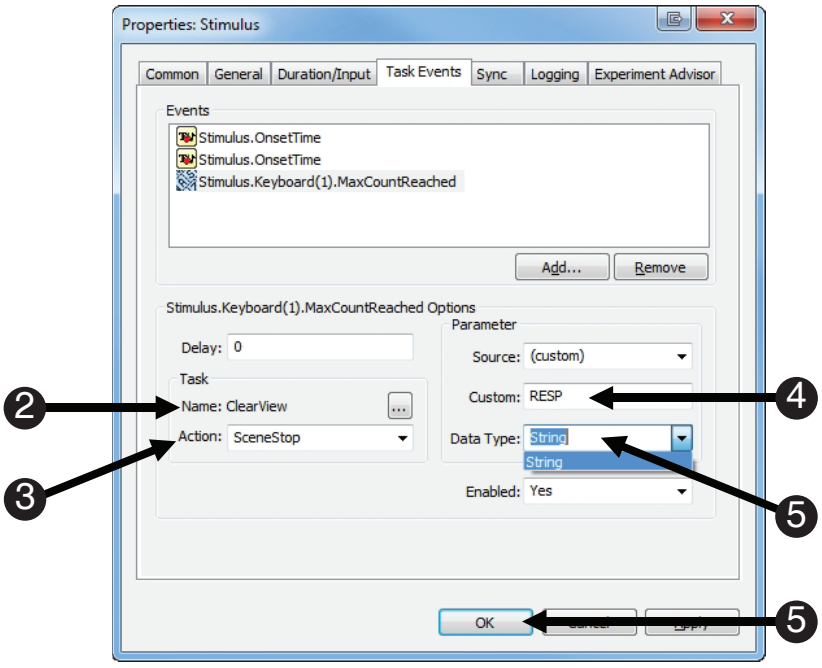
2) ***Note*** the **Name** field ***reads*** **ClearView.**

3) ***Select*** **SceneStop** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **RESP**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 23: Add another Task Event to the Stimulus Object

Define the "Event" component as Stimulus.Keyboard(1).TimeLimitReached; this occurs when no response has been made.

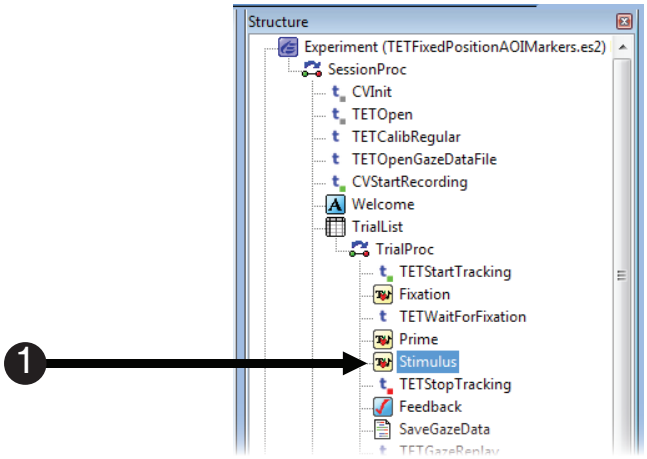
This Task Event will ensure that the RESP scene has a SceneStop marker when no response is made, which is identified when the time limit is reached for the Stimulus Object's InputMask.

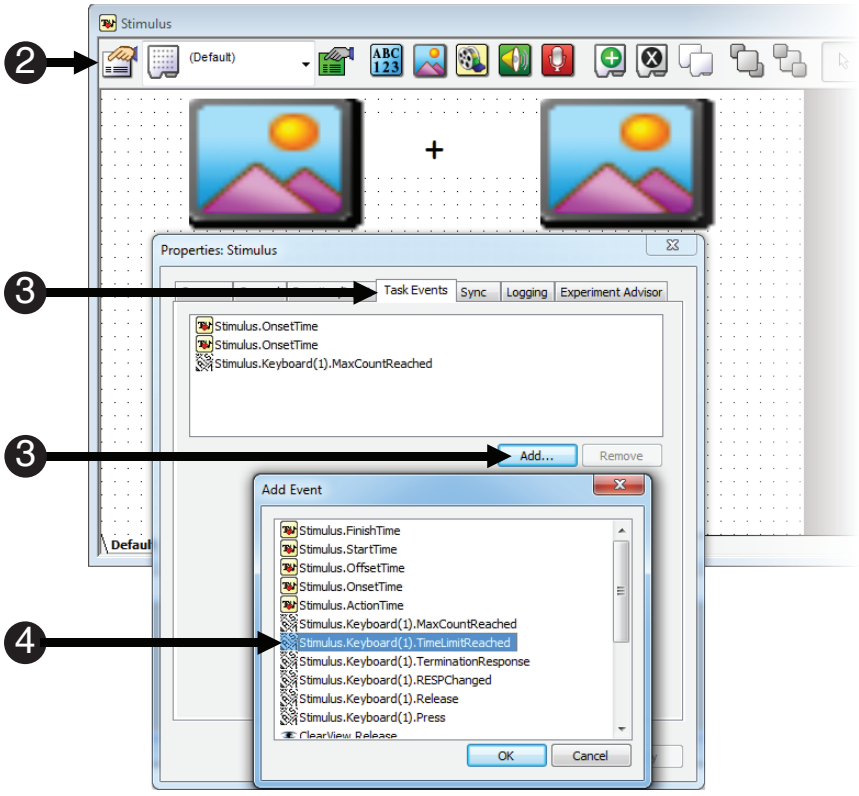
1) If necessary, ***double click*** the **Stimulus** to ***open*** it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Stimulus.Keyboard(1).TimeLimitReached**.





Task 24: Edit the Task associated with the Stimulus.Keyboard(1).TimeLimitReached Event

Define the "Task" component of the Task Event as sending the RESP SceneStop marker to Tobii Studio.

This Task Event will ensure that the RESP scene has a SceneStop marker when the participant does not respond.

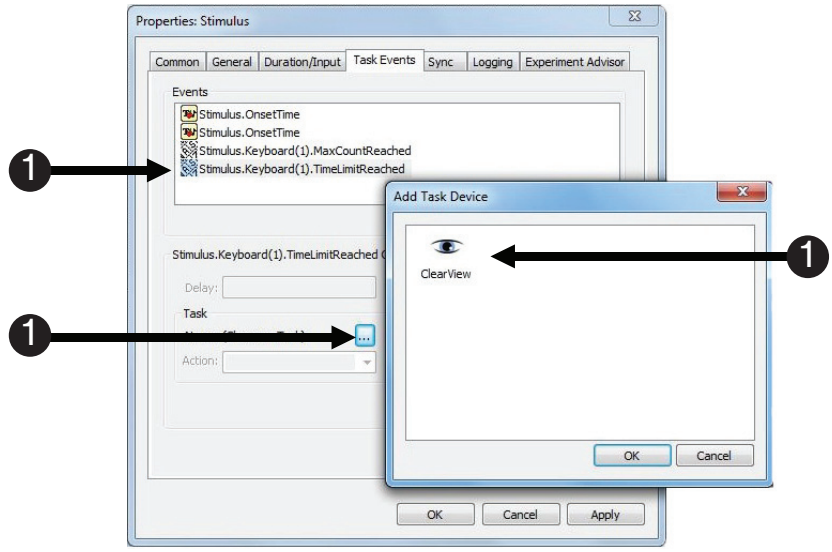
1) ***Select*** the **Stimulus.Keyboard(1).TimeLimitReached** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

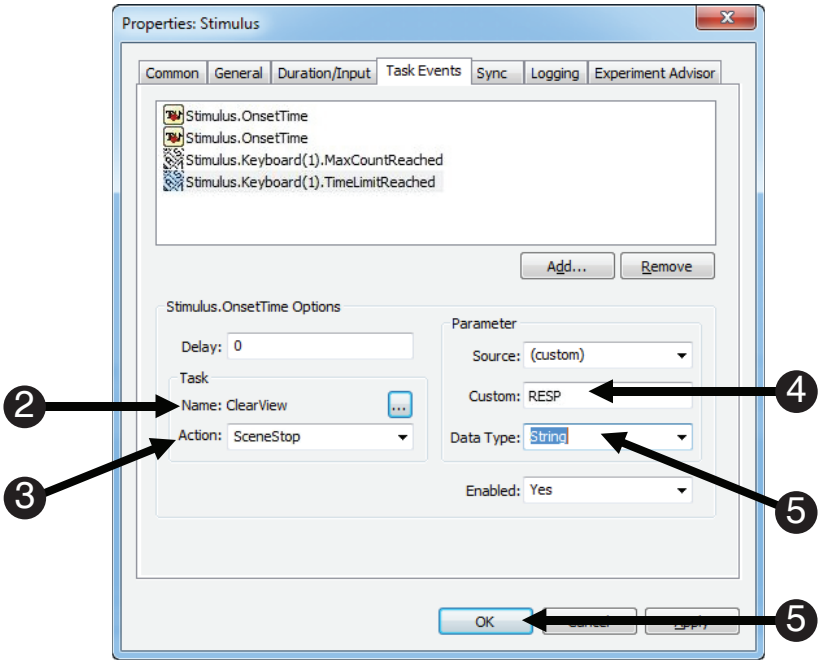
2) ***Note*** the **Name** field ***reads*** **ClearView.**

3) ***Select*** **SceneStop** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **RESP**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 25: Add a Task Event to the Feedback Object

Add the Feedback.OnsetTime Task Event to the Feedback Object.

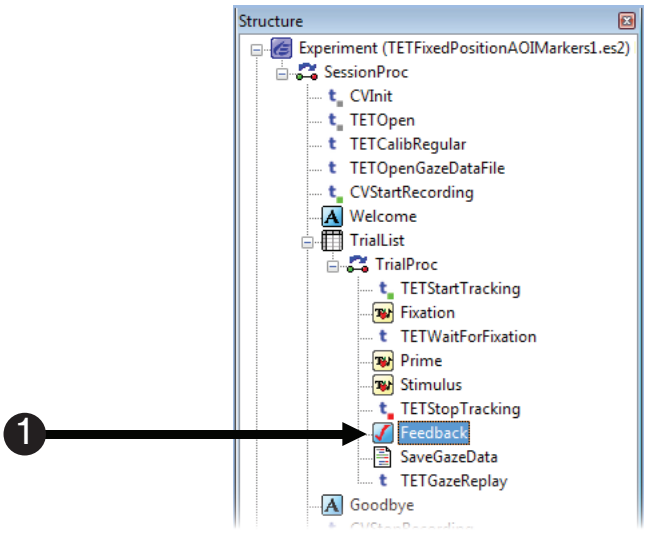
This step will add the .OnsetTime task event to the Feedback Object. This task event will send the end of scene marker for the Stimulus scene to Tobii Studio.

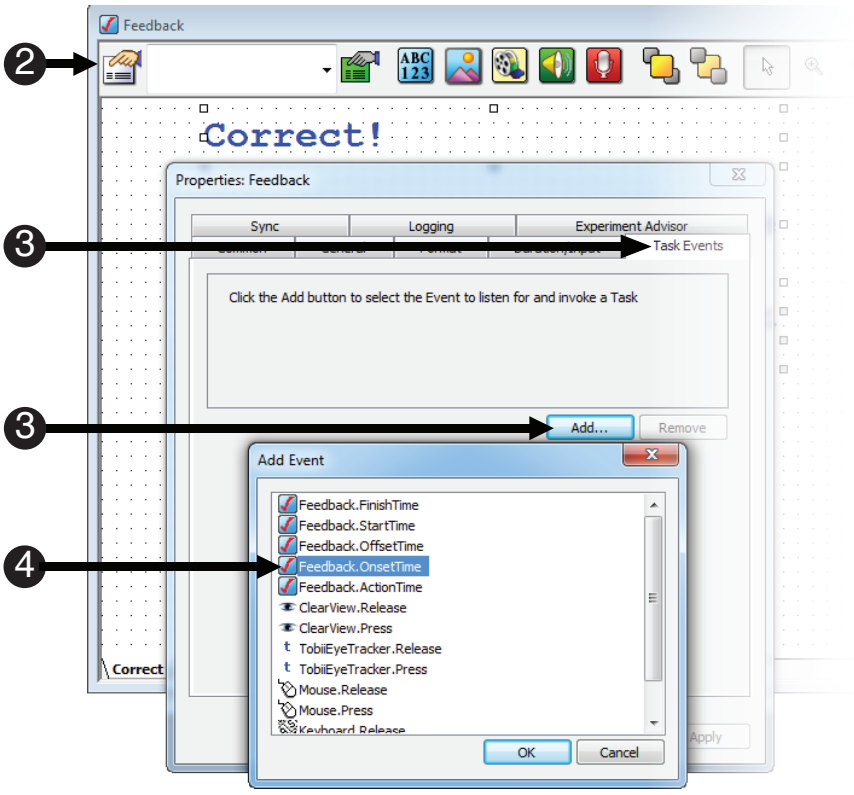
1) ***Double click*** the **Feedback** to ***open*** it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Feedback.OnsetTime**.





Task 26: Edit the Feedback.OnsetTime Task Event

Define the "Event" component as Feedback.OnsetTime.

Define the "Task" component of the Task Event by sending the Stimulus SceneStop marker to Tobii Studio.

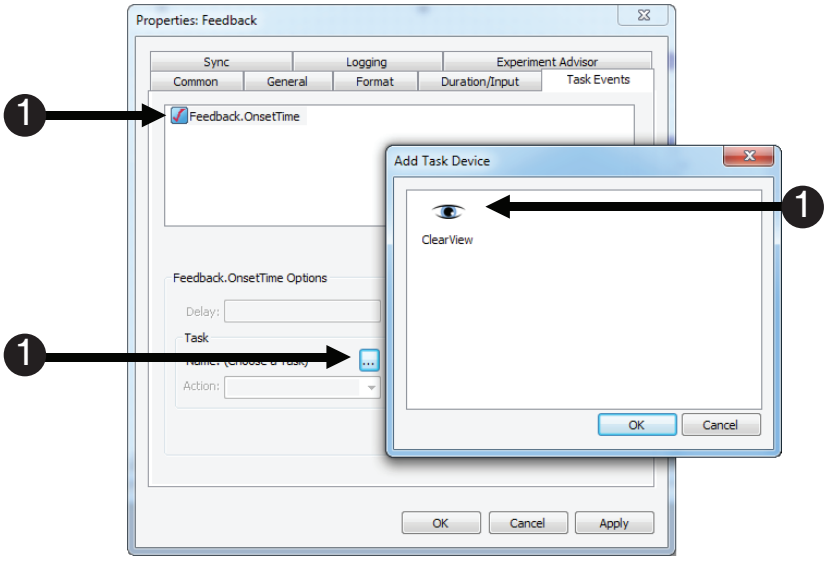
1) ***Select*** the **Feedback.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

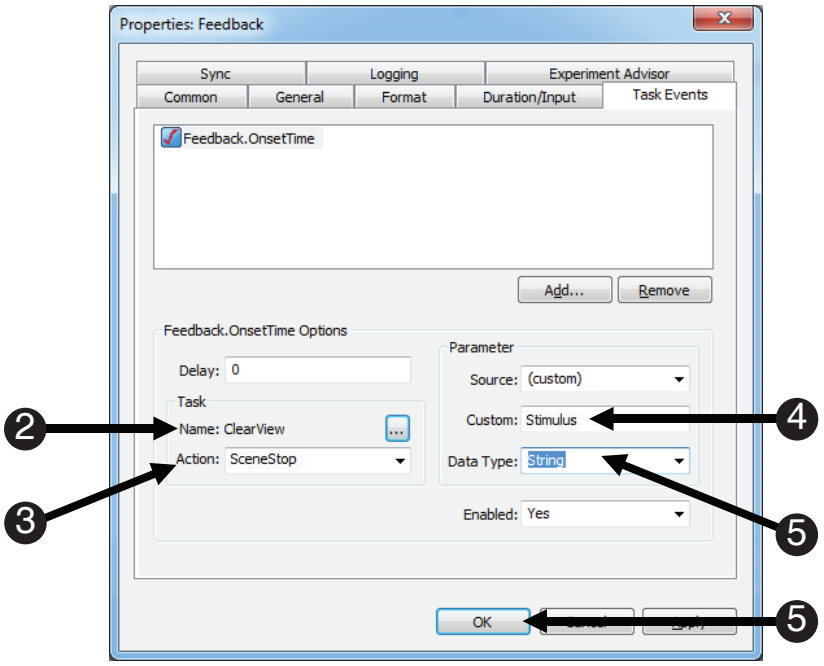
2) ***Note*** the **Name** field ***reads*** **ClearView.**

3) ***Select*** **SceneStop** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **Stimulus**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 27: Add three more Task Events to the Feedback Object

Use the Feedback.OnsetTime event as the trigger for several scenes.

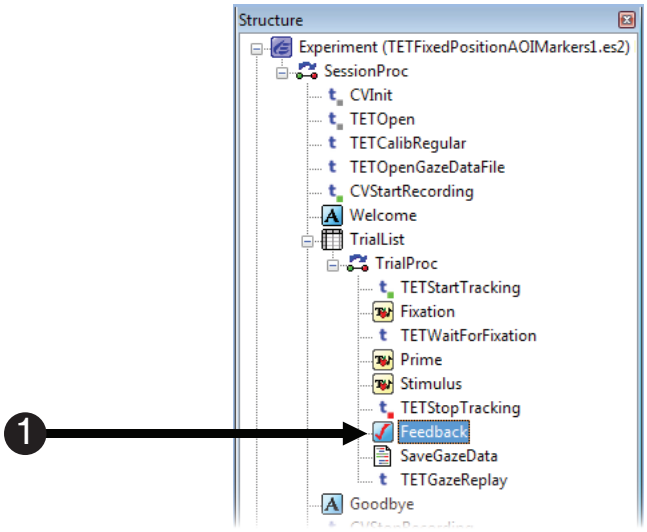
The next step is to add three more Task Events to the Feedback Object, all triggered from the Feedback.OnsetTime event. We use this event to mark the end of several scenes that were started earlier in this tutorial on other objects. We also create a Feedback scene. First, we end the RightImage scene.

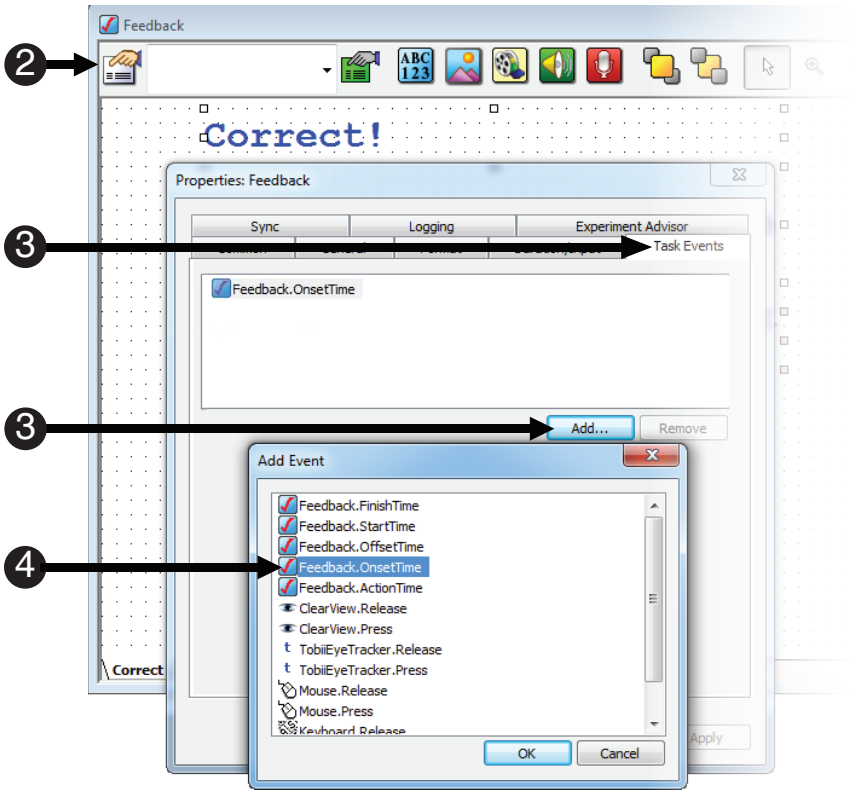
1) If necessary, ***double click*** the **Feedback Object** to ***open*** it in the workspace.

2) ***Click*** the white **Property Pages** button.

3) ***Select*** the **Task Events** tab, and then ***click*** the **Add...** button.

4) ***Double click*** **Feedback.OnsetTime**.





Task 27 (continued): Add three more Task Events to the Feedback Object

Use the Feedback.OnsetTime event as the trigger for several scenes.

We now define the "Task" component of this second Task Event as sending the SceneStop marker for the RightImage scene to Tobii Studio.

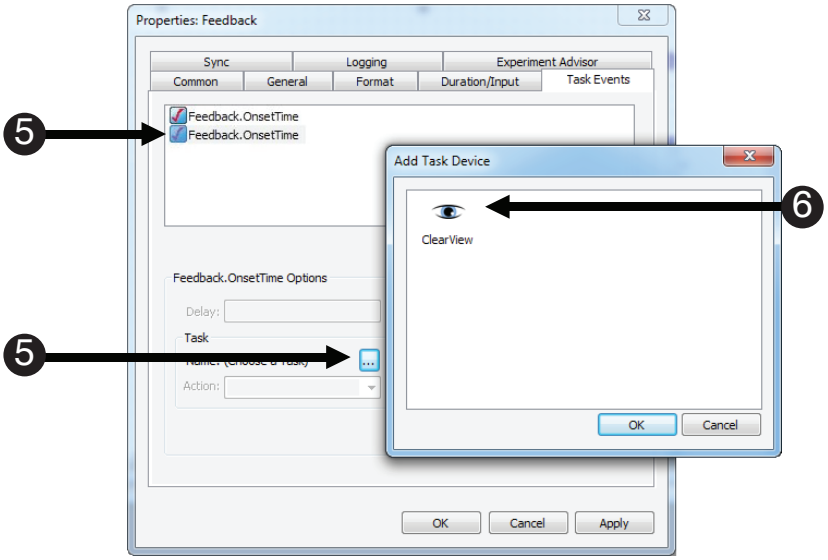
5) ***Select*** the second **Feedback.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

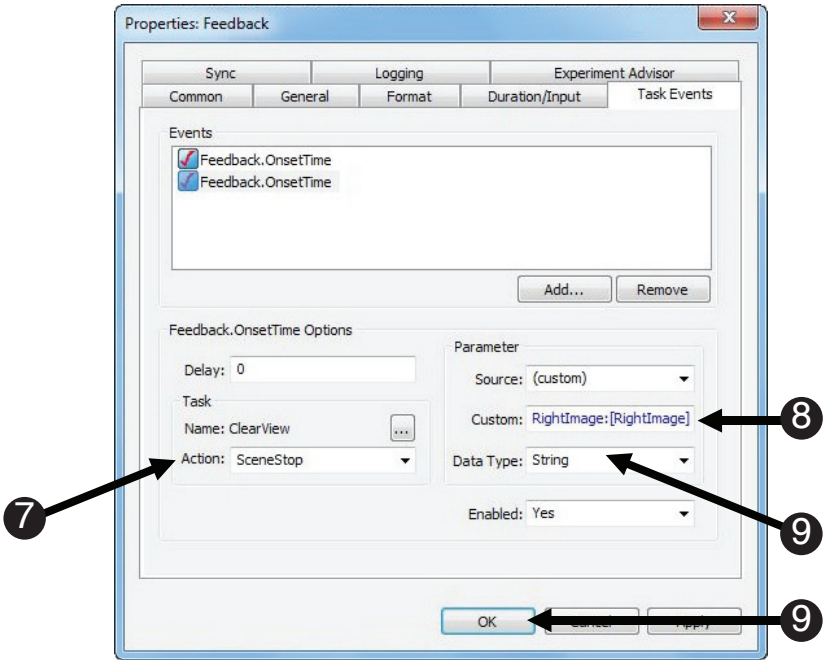
6) ***Note*** the **Name** field ***reads*** **ClearView.**

7) ***Select*** **SceneStop** from the **Action** field.

8) ***Edit*** the **Custom** field to ***read*** **RightImage:[RightImage]**.

9) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 27 (continued): Add three more Task Events to the Feedback Object

Use the Feedback.OnsetTime event as the trigger for several scenes.

We now end the LeftImage scene. Staying with the Feedback Object and still using the Feedback.OnsetTime event, we also create the start of a new scene named Feedback.

10) ***Repeat*** steps **2** through **4** from earlier in this task to create a third **Feedback.OnsetTime** event.

11) ***Select*** **SceneStop** from the **Action** field.

12) ***Select*** **SceneStop** from the **Action** field.

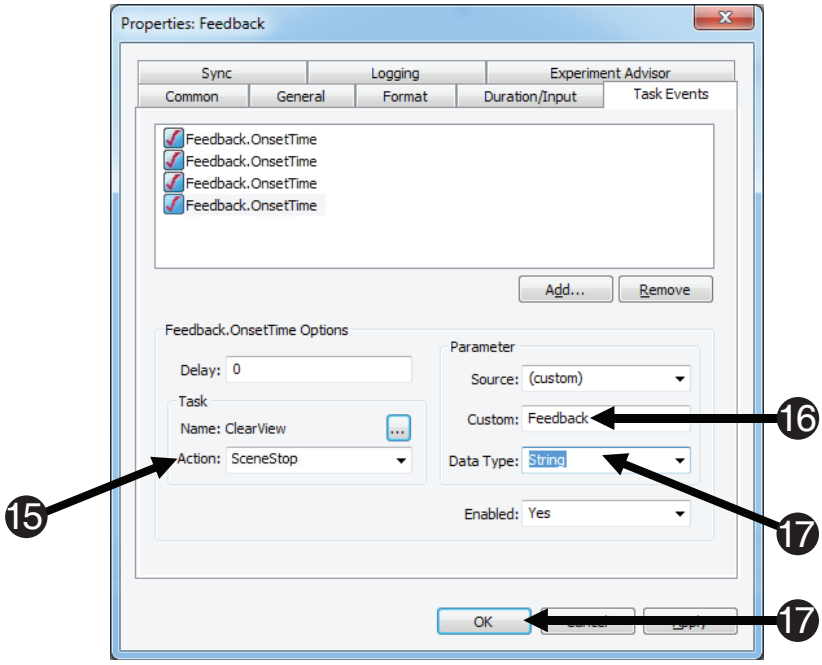
13) ***Select*** **String** as the **Data** type, and ***click*** **OK**.

14) ***Repeat*** steps **2** through **4** from earlier in this task to create a fourth **Feedback.OnsetTime** event.

15) ***Select*** **SceneStop** from the **Action** field.

16) ***Edit*** the **Custom** field to ***read*** **Feedback**.

17) ***Select*** **String** as the **Data** type, and ***click*** **OK**.



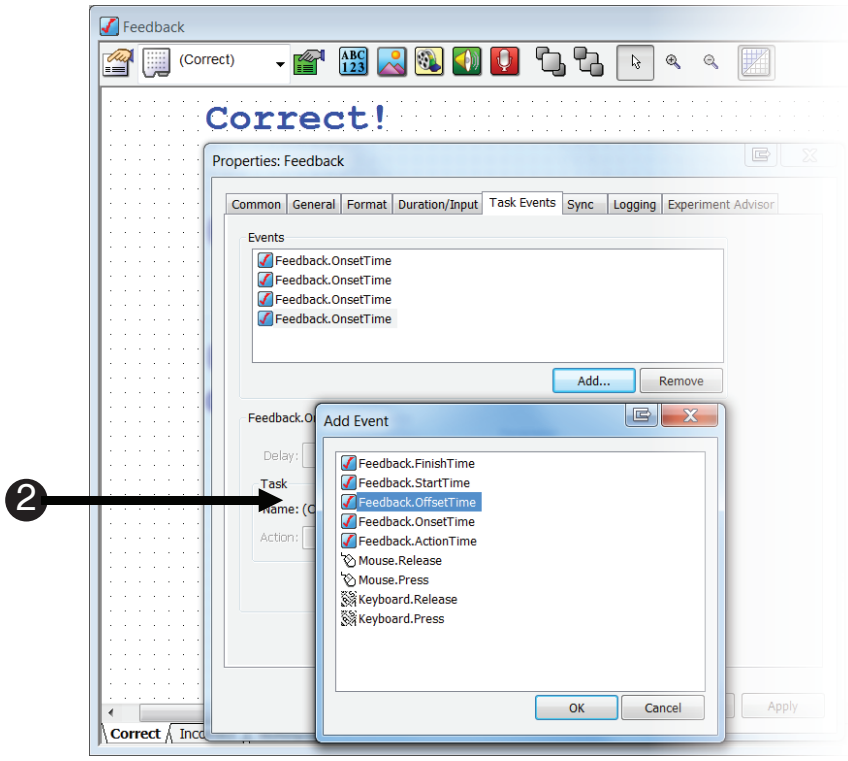
Task 28: Add another Task Event to the Feedback Object

Define the "Event" component of the Task Event with the Feedback.Offset as the trigger.

This step adds the Feedback.OffsetTime event as the triggering event. This event serves as the end of the Feedback scene.

1) ***Repeat*** steps **2** through **4** from **Task 27** to add another Task Event to **Feedback Object**.

2) ***Double click*** **Feedback.OffsetTime**.



Task 29: Complete the Task Event for the Feedback Object

Define the "Task" component of the Task Event as sending the Feedback SceneStop marker to Tobii Studio.

This step sends the SceneStop marker for the Feedback Object.

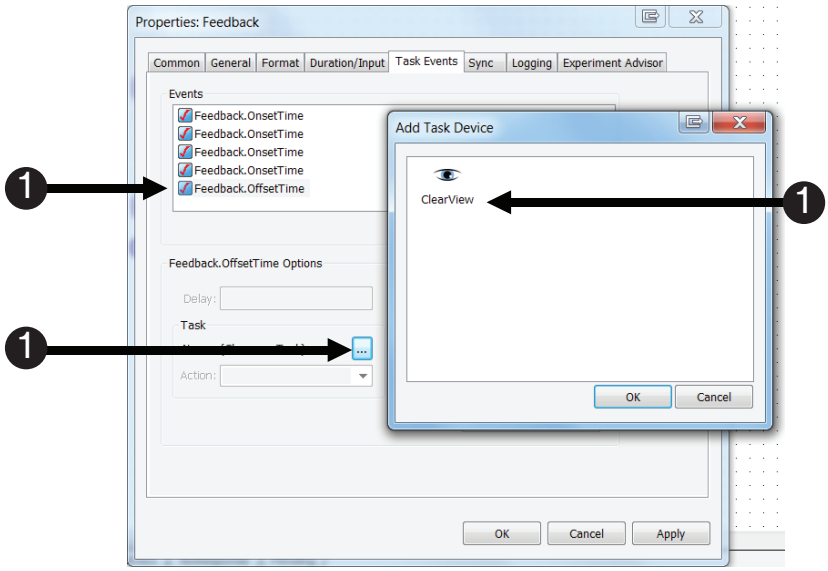
1) ***Select*** the **Feedback.OffsetTime** task event that you created in the previous task.
   Click the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

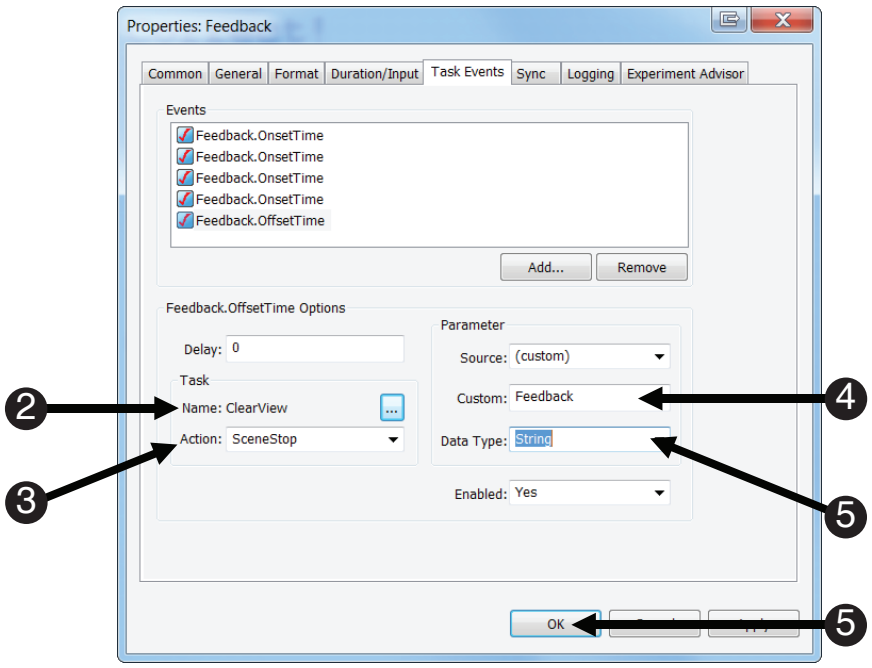
2) ***Note*** the **Name** field ***reads*** **ClearView.**

3) ***Select*** **SceneStop** from the **Action** field.

4) ***Edit*** the **Custom** field to ***read*** **Feedback**.

5) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 30: Create the final Task Event to end the Trial:[Target] scene

Configure one more Task Event on the Feedback Object to end the Trial scene.

This step sends the SceneStop marker for the Trial:[Target] scene. This is the last SceneStop marker that needs to be defined.

1) ***Repeat*** steps **2** through **4** from **Task 27** to add another Task Event to **Feedback Object**.

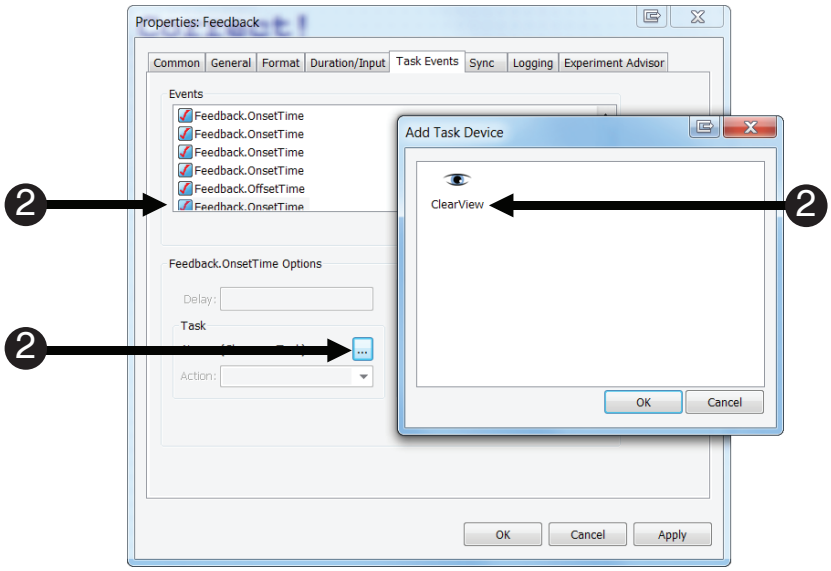
2) ***Select*** the **Feedback.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **ClearView** icon.

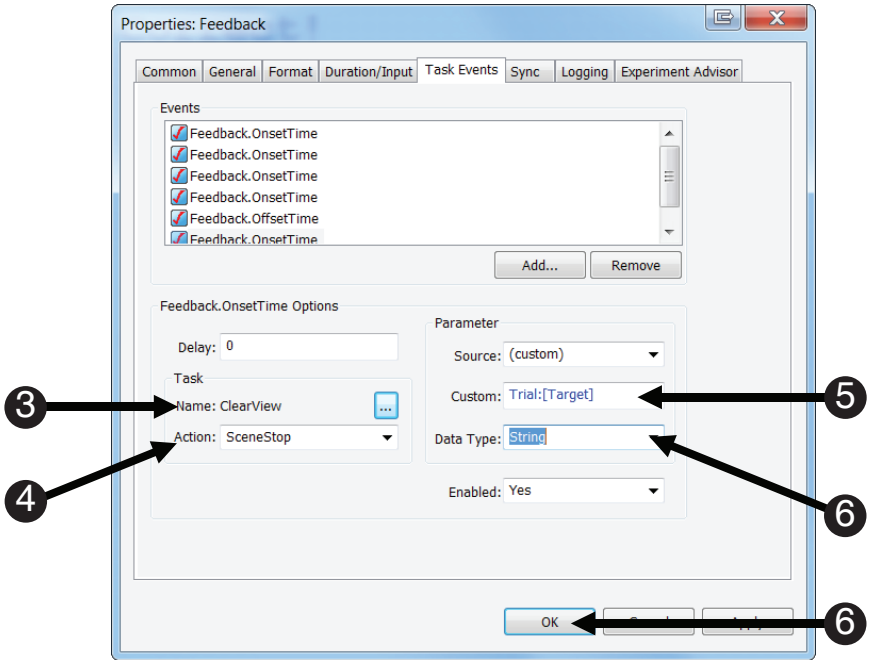
3) ***Note*** the **Name** field ***reads*** **ClearView.**

4) ***Select*** **SceneStop** from the **Action** field.

5) ***Edit*** the **Custom** field to ***read*** **Trial:[Target]**.

6) ***Select*** **String** as the **Data** type, and ***click*** **OK**.





Task 31: Add the CVStopRecording PackageCall

Add a PackageCall to the SessionProc and name the PackageCall "CVStopRecording," and accept the default parameters.

The CVStopRecording PackageCall stops ClearView from recording. This PackageCall should be placed at the location in the procedure where you want to stop communication with Tobii Studio. Accept the default parameters.

1) ***Double click*** the **SessionProc** to open it. ***Drag*** and ***drop*** a new **PackageCall** from the **Toolbox** ***before*** the **TETCloseDataFile PackageCall** and ***after*** the **Goodbye Object**. ***Rename*** the **PackageCall** to "**CVStopRecording**" and ***press*** **Enter** to accept the change.

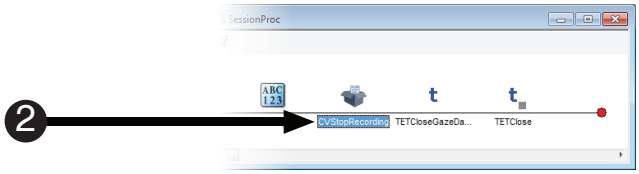
2) ***Double click*** the **CVStopRecording Object** on the **SessionProc** to display its **Property Pages**.

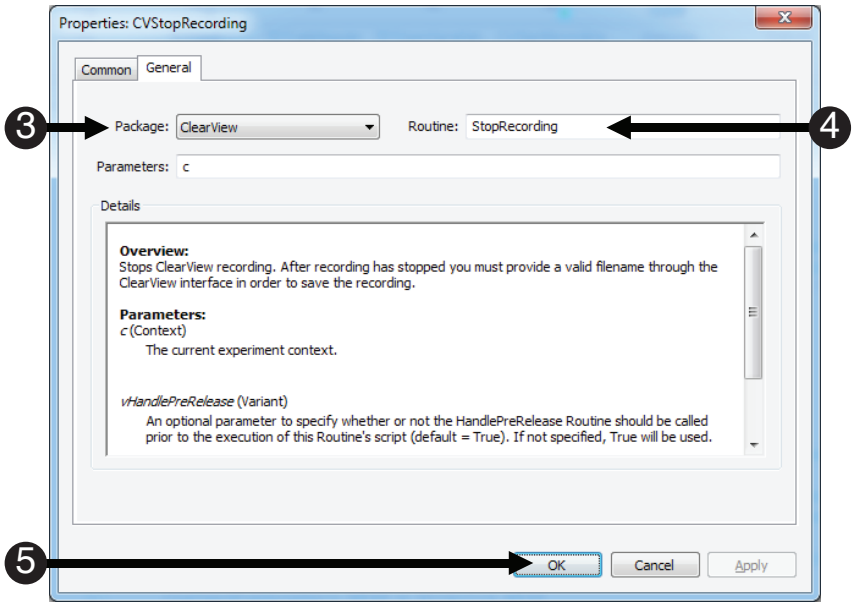
3) ***Select*** **ClearView** from the **Package** dropdown list.

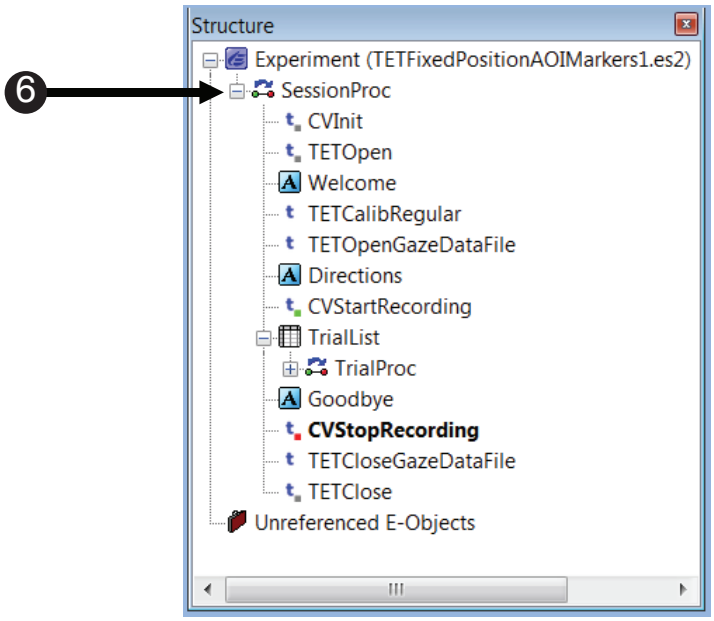
4) ***Select*** **StopRecording** from the **Routine** dropdown list.

5) ***Click*** the **OK** button to accept the default parameters.

6) ***Confirm*** your **SessionProc** is identical to the example shown.







Task 32: Add the CVUnInit PackageCall to uninitialize the ClearView Packages and accept the default parameters

Add a PackageCall to the SessionProc and name the PackageCall "CVUnInit."

The CVUnInit PackageCall uninitializes the ClearView package file. Add the PackageCall to the SessionProc and Accept the default parameters.

1) ***Drag*** and ***drop*** a new **PackageCall** from the **Toolbox** after the **TETClose PackageCall**. ***Rename*** the **PackageCall** to "**CVUnInit**" and ***press*** **Enter** to accept the change.

2) ***Double click*** the **CVUnInit Object** on the **SessionProc** to display its **Property Pages**.

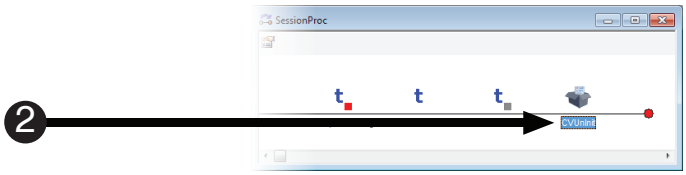
3) ***Select*** **ClearView** from the **Package** dropdown list.

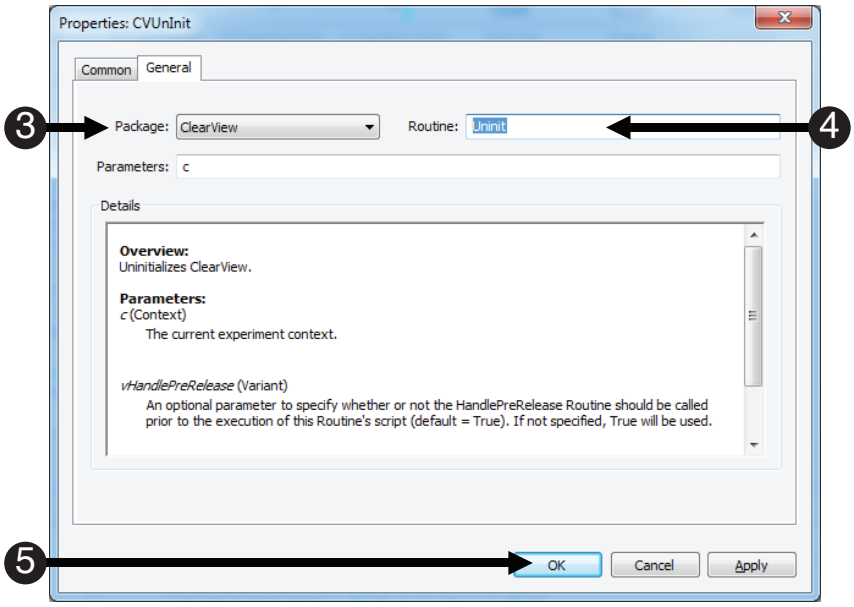
4) ***Select*** **UnInit** from the **Routine** dropdown list.

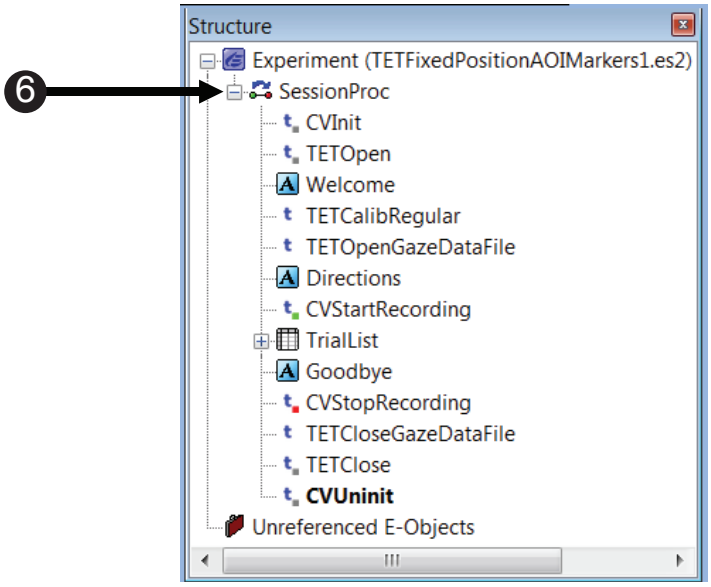
5) ***Click*** the **OK** button to accept the default parameters.

6) ***Confirm*** your **SessionProc** is identical to the example shown.

⚠ **NOTE:** *You will need to set up an external video object in Tobii Studio to collect data from this experiment. Please refer to Tobii Studio documentation on how to do this.*







Task 33: Run the Experiment

Run the experiment to verify that the SceneStart and SceneStop markers are being sent by E-Prime and received by Tobii Studio as expected.

⚠ **NOTE:** *As was noted in the introduction to Tutorial 5, Tobii Studio requires that calibration be performed to start the external video object. Since this experiment already includes calls to the TET PackageFile Routine that performs calibration, you will end up performing calibration twice. If you wish to avoid the calibration step in E-Prime, highlight the TETCalibRegular PackageCall and press Delete. This will move the PackageCall to Unreferenced E-Objects, and allow the experiment to be run without first performing the extra calibration.*

The next steps will walk you through generating the experimental script, starting the experimental paradigm, calibration, and running the experiment.

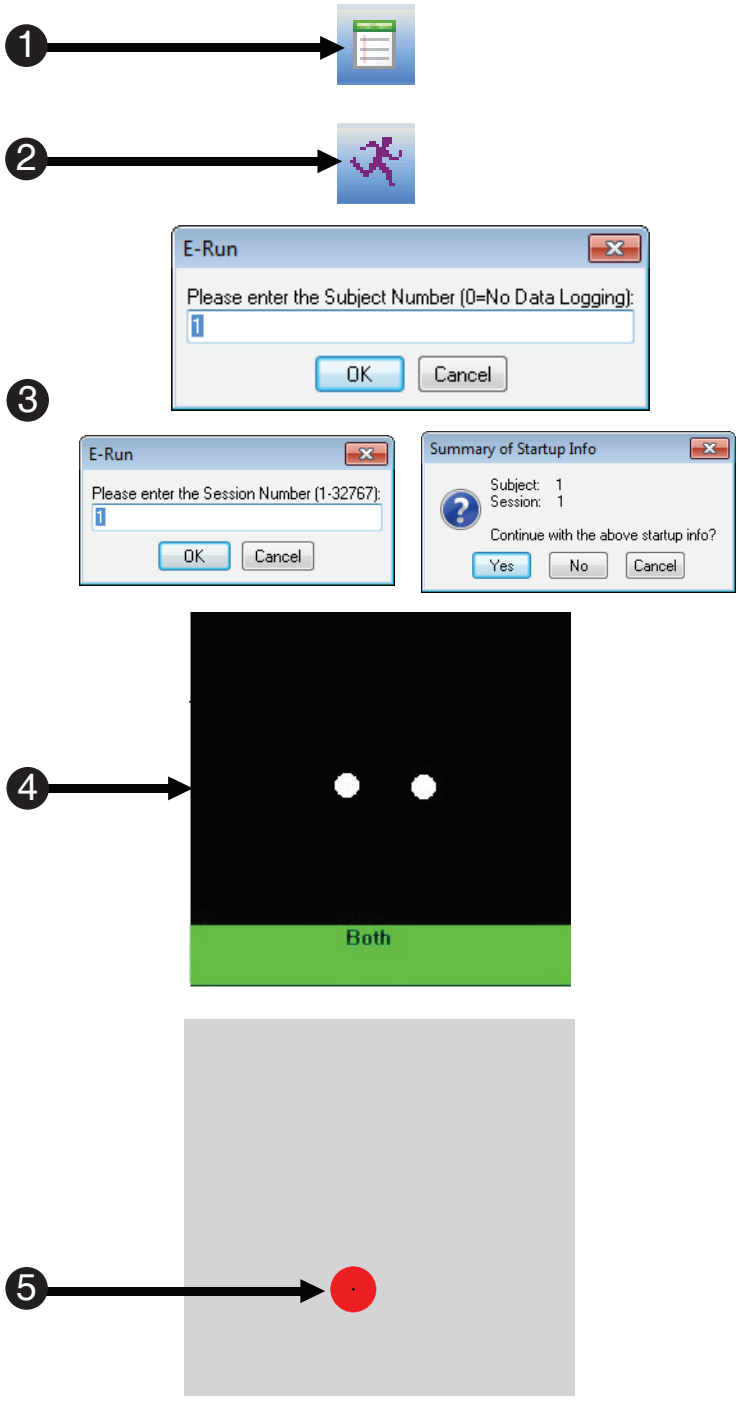
1) ***Press* Ctrl+S** to save your work before continuing. ***Click*** the **generate icon** or ***press* Ctrl+F7** to generate the script and ***check* it** for **errors**.

2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

3) ***Click* OK** to accept the **default values** for **Subject Number**, **Session Number** and **Yes** for **Summary of Startup Info**.

4) ***Look*** at the **screen** to ***verify*** the **eyes** are **stable** in the track status window. ***Ensure*** that there are **two white dots** that appear in the box and that the **bottom bar** is **green**. ***Press* Space** once both eyes are stable.

5) ***Run*** through the **calibration process** by ***following*** the **dots** on the **screen** with your **eyes**.



Task 33 (continued): Run the Experiment

Run the experiment to verify that the eye tracker is working, and to ensure the .gazedata file is written.

Once you have completed calibration, you will be prompted to verify that the data is acceptable.

6) ***Accept*** the **calibration**.

⚠ **NOTE:** *Please refer to* **Tutorial 1, Task 17 (continued): Run the Experiment, (Page 44)** *for additional information about calibration.*

Scene-based Analysis of E-Prime Experiments in Tobii Studio

Tobii Studio supports segment-based and scene-based analysis of data sent from E-Prime. For more information on segment-based analysis, see the KB 5624: INFO: Supporting Segment-based Analyses in E-Prime Extensions for Tobii. This section focuses on Scene-based analysis. The advantage of a scene-based analysis in Tobii Studio is that it provides more robust data analysis based on E-Prime scenes. Scene-based analysis affords the opportunity to receive more descriptive scene data analyses sent over by E-Prime during a trial run using the Task Events feature.

During this tutorial, you created an E-Prime experiment that defined multiple scenes. The purpose of these scenes is to send information about the experiment trial sequence to Tobii Studio. Tobii Studio, in turn, can utilize the scene information to perform complex analysis, including the creation and display of gaze visualization and heat maps.

Once you have successfully run your experiment in E-Prime, please switch to the Tobii computer. You should no longer see the same “record” screen that you did before the run of the experiment. This means that the run has been completed and that there should be data.

This section describes the steps that we took to perform a meaningful analysis on the data that is created from the completed Tutorial 5 experiment in Tobii Studio. With this data, we are interested how long participants looked at both the Distractor and Stimulus images during the entire experiment. This will allow us to determine if the participant was spending more time fixating on the distractor image in each task or the target stimulus for each task. To get this information, we will look at two scenes that we created in Tutorial 5. While the other scenes are useful, these are the only two that we will focus on in this chapter. Unlike the previous sections of this Tutorial, we do not present a step-by-step guide to using Tobii Studio here. Instead, this section is intended to provide a very high-level overview of analyzing gaze data in Tobii Studio. The steps listed below are meant only as a guide and in some instances do not reflect the order in which events need to be carried out. This section demonstrates how to pull meaningful data from the scenes that were created in E-Prime as well as give a brief overview of the many data analysis options available for the gaze data. For specific information on data analysis in Tobii Studio, please refer to the Tobii Studio User Manual.

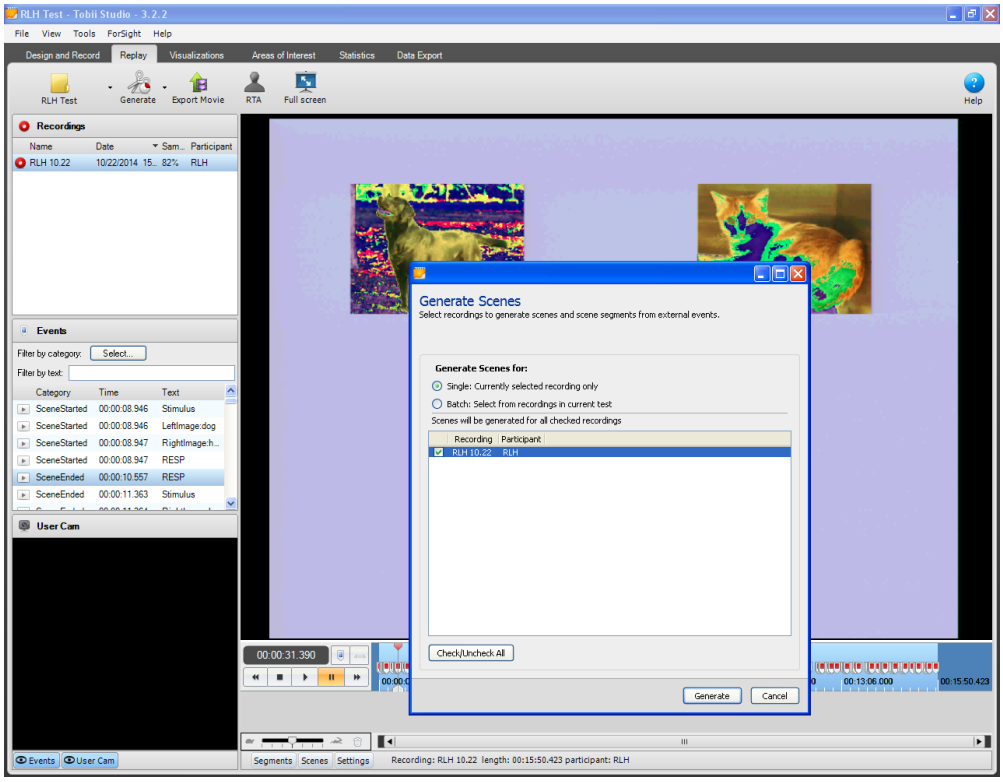
1) Compile and Generate Scene Data

Selecting the Replay Tab allows the user to access basic statistics about the participant’s data file. Minimally, there will be a full listing of all of the pre-defined scenes in the Events window on the left-hand side of the window. At the bottom of the screen, you will see an Event Viewer. This contains a timeline of the entire experiment with a red flag in the timeline that represents the SceneStart and SceneStop markers that were sent to Tobii during the recording.

In order to perform the aforementioned analysis on our scene data, we have to generate our scenes in Tobii Studio. The purpose of this is to pair each SceneStart marker with every SceneStop marker of the same name to create a cohesive scene.

Please note that every SceneStart marker needs a SceneStop marker in order to create a Scene of any particular event. ⚠**NOTE:** *If a scene is missing either a SceneStart or SceneStop marker, Tobii Studio will not create a scene of that event.*

To generate the scenes in Tobii Studio, select Generate under the Replay tab, select Scenes, and then select the recordings for which you would like to analyze scenes:



2) Review Visualizations (Optional)

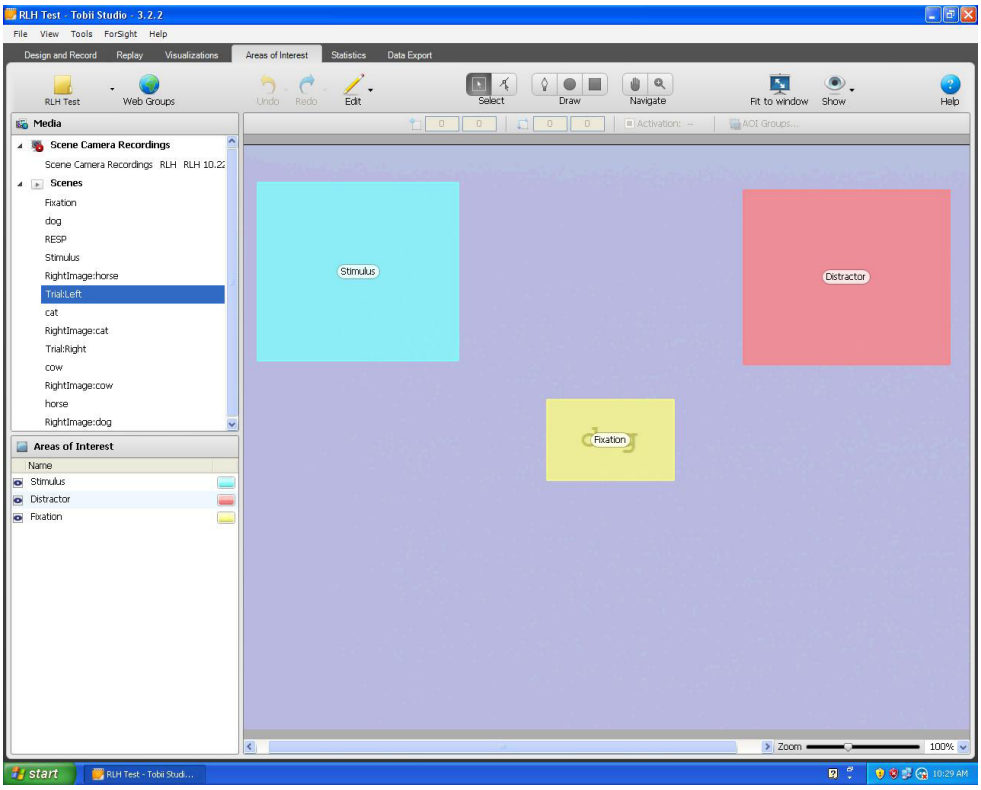
The Visualization Tab provides a variety of visual representations of the gaze data for each scene. Gaze Plot, Heat Map and Gaze Cluster analyses are supported. While this data is of interest to many researcher studies, it is not relevant to the analysis that is described here and therefore we will not be working within this tab.

3) Create Areas of Interest (AOIs)

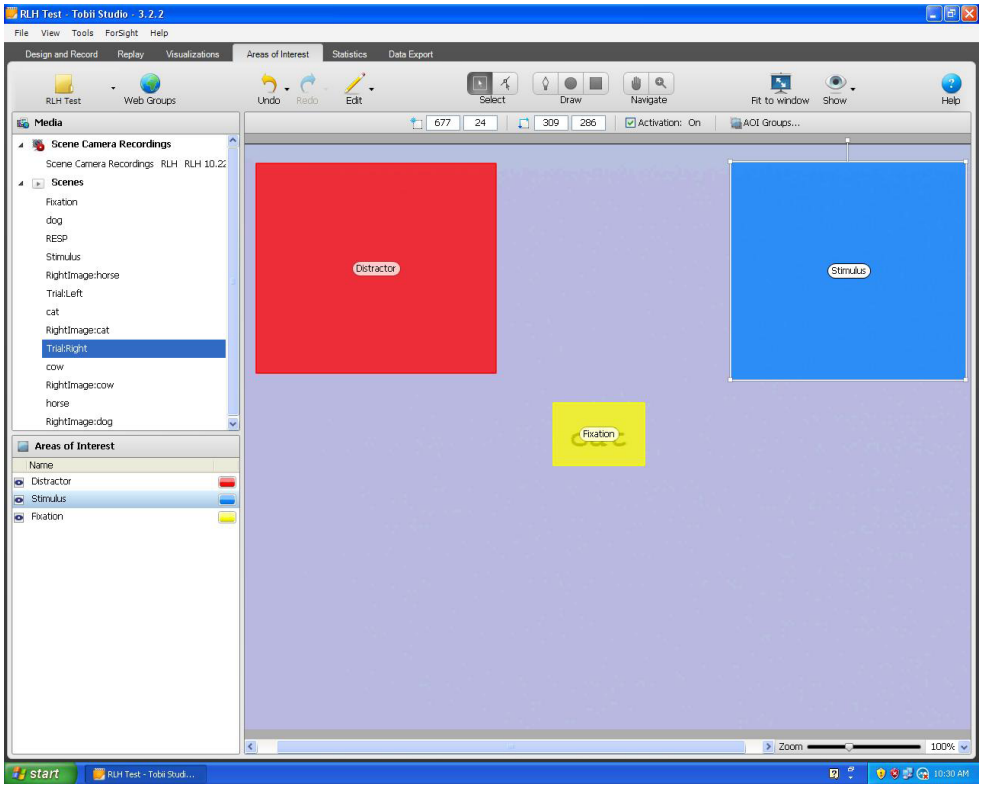
One of the most important steps of analyzing scenes in Tobii Studio is to create Areas of Interest (AOIs) for each scene. AOIs allow Tobii Studio to produce more robust data by defining areas of the screen considered to be the important or interesting parts of each scene.

To do this, first click on the Areas of Interest Tab. Next, select the scene from which you would like to analyze data. Click on the “Draw” button at the top of the window then drag and drop the areas of the screen of which you would like to make an AOI. Once the AOI is created, name the AOI something descriptive.

For this analysis, first click on the Trial:Left scene. We will need to create three AOIs here. The first AOI will cover the location of the distractor image as it appears on the screen. This AOI will be called “Distractor”. Since this is the Trial:Left condition, the distractor image will always be on the right. The next AOI will cover the target stimulus object as it appears on the screen. Since this is the Trial:Left condition, the target Stimulus will always appear on the left side of the screen. Name this AOI “Stimulus”. Finally, place a small AOI in the very center of the screen; this AOI will be the fixation area of interest, called “Fixation”. This Fixation AOI will not be directly used in our analysis as we are only interested in the distractor and target stimulus. However, adding the Fixation as an AOI is still very helpful, especially if further analysis of this data is planned.



The AOI scheme that is now placed on the Trial:Left scene needs to be repeated on the “Trial:Right” scene. This can be done by simply highlighting the AOIs on the Trial:Left scene object in Tobii Studio, pressing Ctrl+C to copy, opening up the Trial:Right scene and pressing Ctrl+V to paste these objects. This will allow you to automatically create the necessary AOIs. Keep in mind that the location of the Distractor and Stimulus AOIs will have to change for this scene. This is just a simple matter of renaming and re-coloring the AOIs.

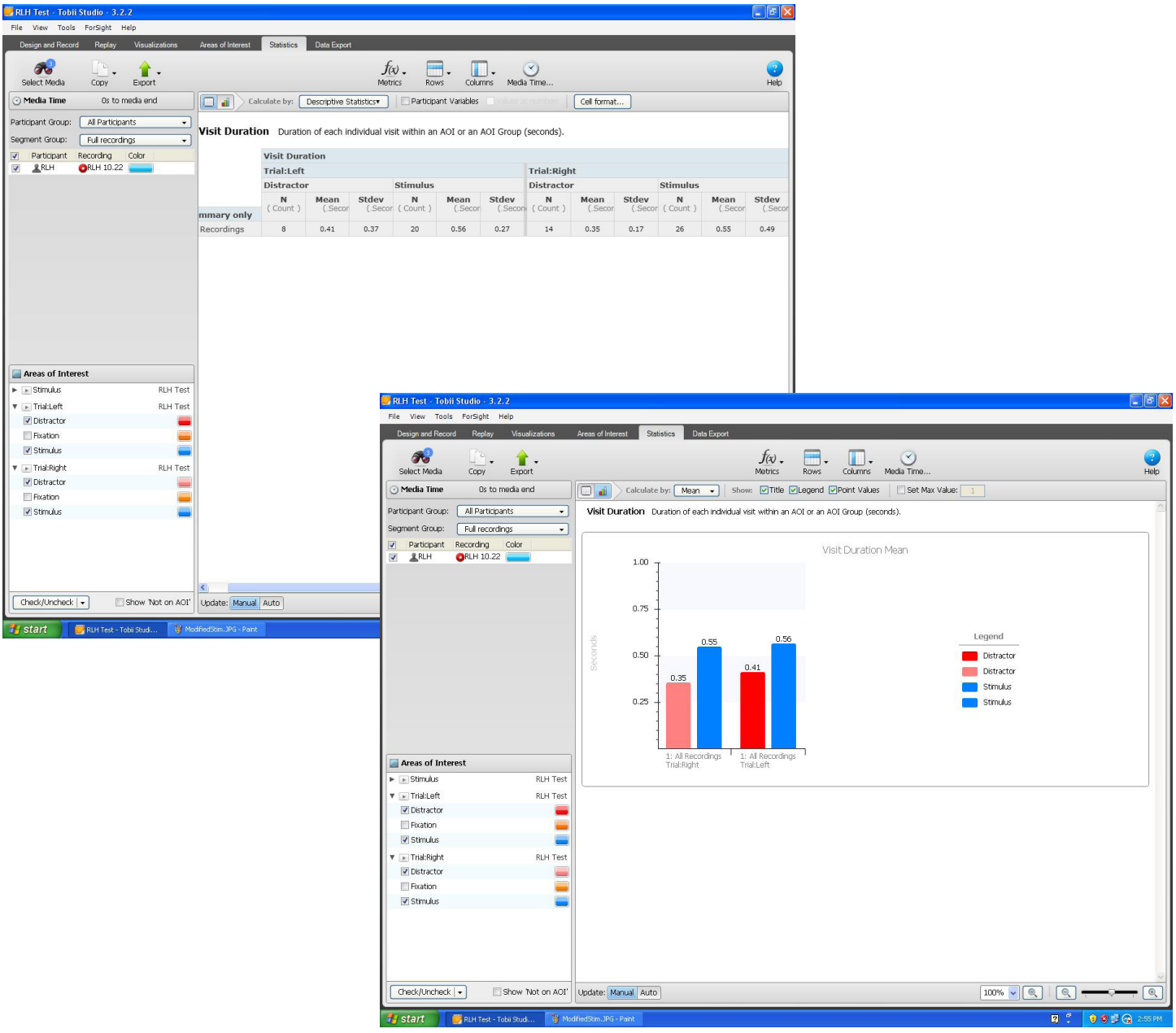


4) Display Statistics

To have Tobii Studio generate the statistics from the AOIs that were created in Step 4 above, click on the Statistics tab at the top of the window. In this tab, select the Trial:Left and Trial:Right scenes. They should be the only scenes that you are able to add because they are the only scenes that contain AOIs.

After making this selection, the screen on the right-hand side of the window will require you to update it. Click on the Update button to display the newly created statistics. Before clicking on this button, locate and select the Metrics button at the top of the screen. In the Metrics dropdown you will see that Time to First Fixation is already checked. Click on this once to remove its checkmark. Once you have done this, scroll down until you see the Visit Duration option and select this. If you would like to add additional statistics to your analysis, click on the Metrics button at the top of the window and select the analysis that best suits your paradigm. Now that the Visit Duration metric is selected, click the update button.

The data can be displayed in either a chart format or in a table format. Both formats are demonstrated below. This data will now tell us exactly how long participants looked at the distractor and the target stimuli for the experiment.



Tutorial 6: **Multiple Monitors: Creating a Participant Station**

Summary:

The TET PackageCalls allow you to set up a participant station. With this functionality you can display different parts of your experiment on more than one monitor. This allows you to create a Tobii display and an Experimenter display. The Tobii display presents the part of the experiment you want the participant to see, such as the calibration and experimental stimuli. The Experimenter display will in turn prevent the participant from seeing the housekeeping aspects of the experiment, e.g. calibration results and participant information entry screens. This is ideal for infant studies, or other instances where these aspects of the experiment can distract the participant.

Please keep in mind that the display settings and video cards may be different on each machine. In order to get the results to display in the way you are expecting you may have to change cable positions from your video card and/or switch display indexes for the Tobii and Experimenter display. The display index can be determined either by the machine's Screen Resolution dialog or by running dxdiag.exe via the start menu. If you need help with determining your display index, please consult your computer administrator. It is very important that operating system you are using sets the Tobii monitor as the main display (different from display index). This makes the upper left corner of the Tobii monitor 0,0 coordinate, which is necessary to correctly log the x,y eye coordinates within your .gazedata file. Once the Tobii display is set as the main display, identify which display index it is set as and use that index for the Tobii monitor in your experiment. Be advised that the type of cables you use to connect to the video card can also affect which display the computer thinks is display index 1 and 2. We advise using a dual headed video card and connecting the displays via which ever connectors are native to that video card (e.g., vga to vga and dvi to dvi).

During this tutorial, you will add and edit a Display Device Object to the TETFixedPositionAOI.es2 sample experiment that you have altered in the previous tutorials. You will also create script to assign different aspects of existing E-Prime Extensions for Tobii PackageCalls to different displays. When completed, the Tobii display will first ask to calibrate the Tobii Eye Tracker and the results of the calibration will be shown on the "Experimenter" display for analysis and approval. Upon approval, instructions will be shown on the Tobii display, followed by the fixations and stimuli. Once all of the trials have been completed, a Goodbye screen will be shown on the Tobii display and the experiment will terminate.

Goal:

This tutorial illustrates how to add multiple display capabilities to the TETFixedPositionAOI.es2 sample. When you have completed this tutorial, you will have created a paradigm that displays different aspects of the experiment on different displays.

Overview of Tasks:

- Open TETFixedPositionAOI.es2 in E-Studio.
- Save the experiment under a new name.
- Rename the existing Display Device in the Experiment Object.
- Add a second Display Device to the Experiment Object.
- Edit the properties of the second Display Device.
- Move the second Display Device to a higher position in the Experiment Object Properties.
- Add an InLine Object to the SessionProc.
- Add script to the InLine Object to display desired items on each monitor.
- Verify the overall experiment structure and run the experiment.

Estimated Time: 10-20 minutes

Task 1: Open the TETFixedPositionAOI.es2 Experiment in E-Studio

Locate the E-Studio icon in the Start > Programs > E-Prime menu and launch the application by selecting it. Load the TETFixedPositionAOI.es2 sample experiment.

The E-Studio application is installed as part of the typical E-Prime installation. This application is used to create, modify, and test experiments within E-Prime. Open the E-Studio application, navigate to My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOIMultiMonitor, and load the TETFixedPositionAOI.es2 sample experiment.

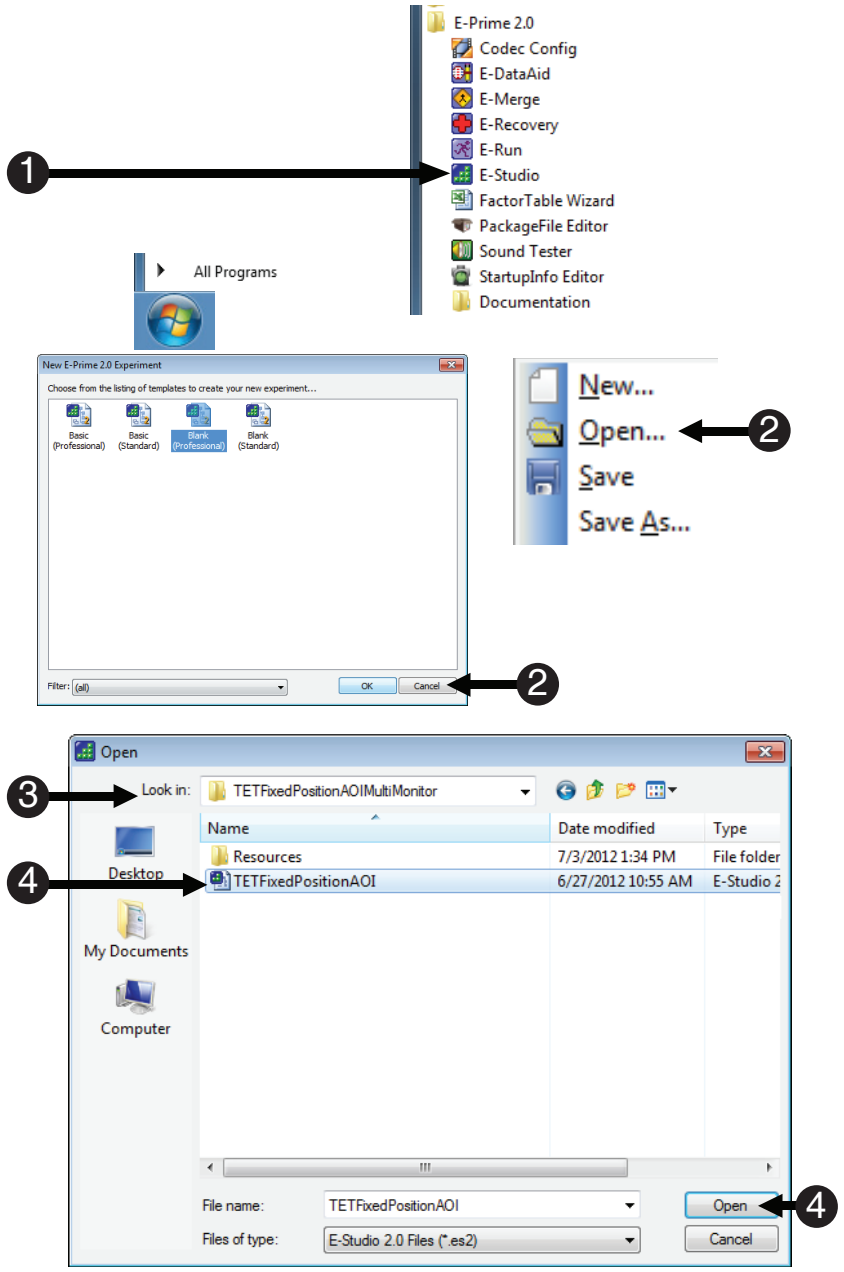
1) ***Click*** on the Windows **Start** menu, ***select*** **All Programs**, and then ***select*** **E-Prime**. From the menu, ***click*** on **E-Studio** to launch the application.

2) ***Click*** the **Cancel** button. ***Select*** **File** > **Open**.

3) ***Navigate*** to the "**…\My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOIMultiMonitor**" folder to ***load*** the paradigm.

4) ***Select*** the **TETFixedPositionAOI.es2** file and then ***click*** the **Open** button to ***load*** the **paradigm** into E-Studio.

⚠ **NOTE:** *If you cannot find the TETFixedPositionAOI.es2 file, you may need to refresh your E-Prime Samples and Tutorials folders. Select Tools\Options… from the E-Studio menu bar then click "Copy Samples and Tutorials to My Experiments folder…"*



Task 2: Save the Experiment Under a New Name

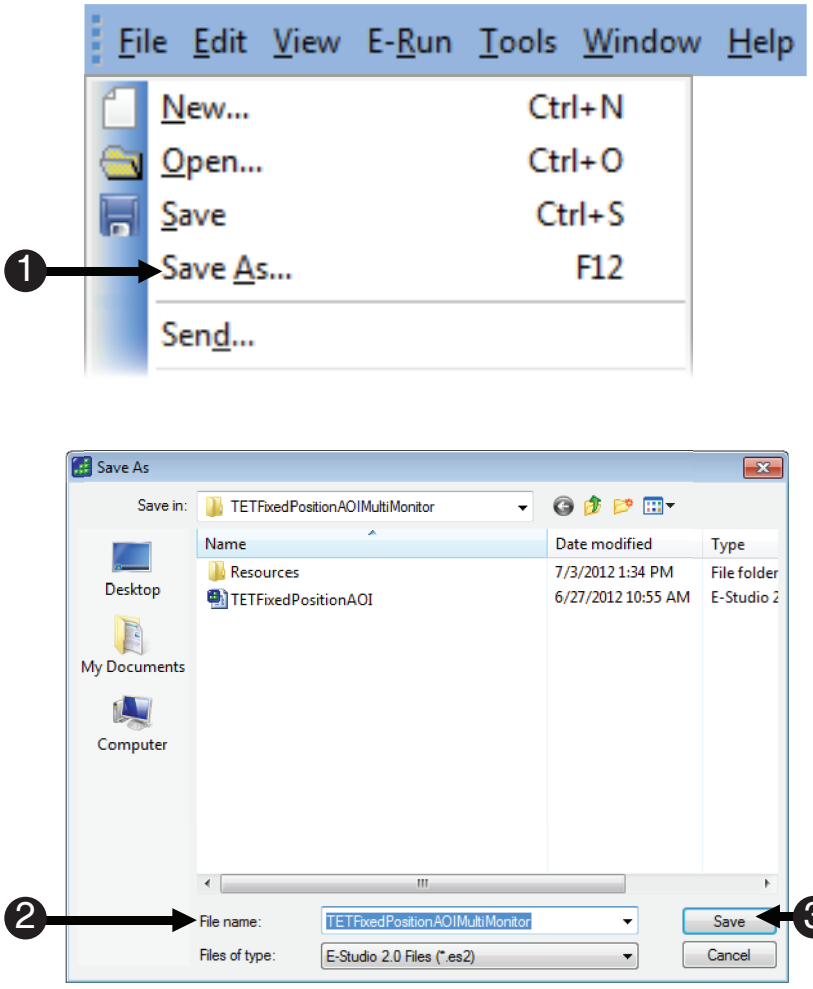
Save the TETFixedPositionAOI.es2 experiment in the same folder under the new name "TETFixedPositionAOIMultiMonitor.es2".

Rename the experiment, but be sure to save it in the same folder ("…\My Experiments\Tobii\Tutorials\TET\TETFixedPositionAOIMultiMonitor\") so that any resources within the experiment will remain valid and can be reused.

1) ***Select*** **File** > **Save As**… from the application menu bar.

2) ***Type*** "**TETFixedPositionAOI MultiMonitor.es2**" as the new name in the File name field.

3) ***Click*** the **Save** button.



Task 3: Rename the existing Display Device in the Experiment Object

Open the Property Pages for the Experiment Object and rename the Display device to "ExperimenterDisplay"

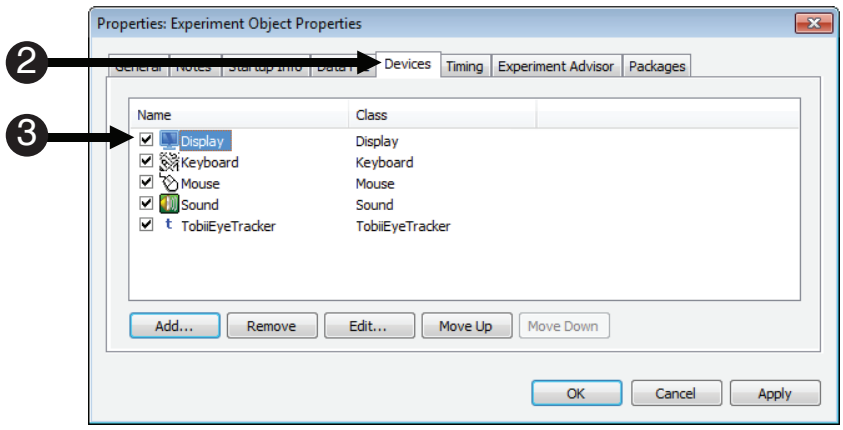
You will need to rename the default Display Device name from "Display" to "ExperimenterDisplay" in order to distinguish which Display Device will be used to display different aspects of the experiment.

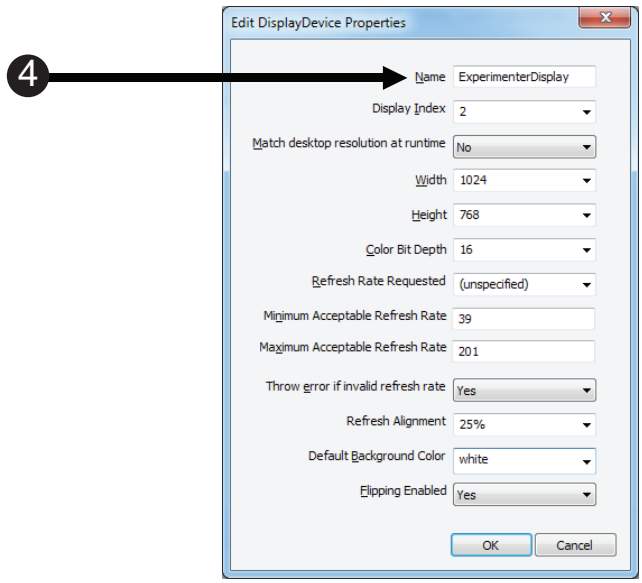
1) ***Double click*** the **Experiment Object** at the top of the tree in the Structure view.

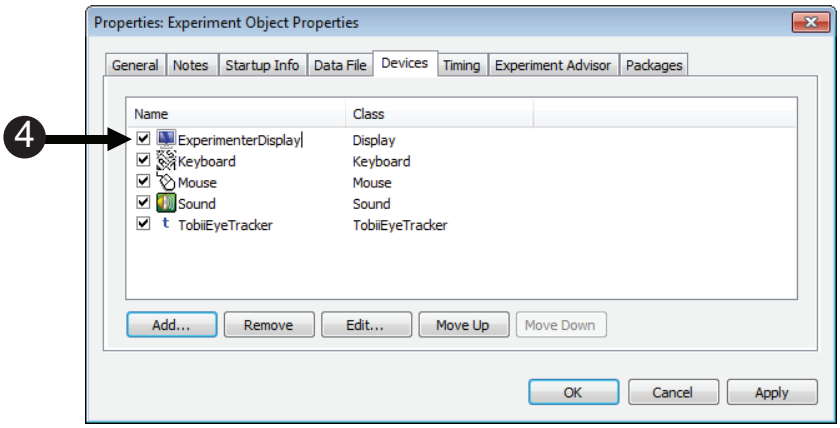
2) ***Click*** on the **Devices** tab of the **Experiment Object Properties** dialog ***double click*** **Display**.

3) ***Rename*** the **Display** object to "**ExperimenterDisplay**" and ***set*** the **Display Index** to **2**. ***Click*** **OK**.

4) ***Type*** "**ExperimenterDisplay**" as the new object name and then ***press*** **Enter** to accept the change.







Task 4: Add a Second Display Device to the Experiment Object

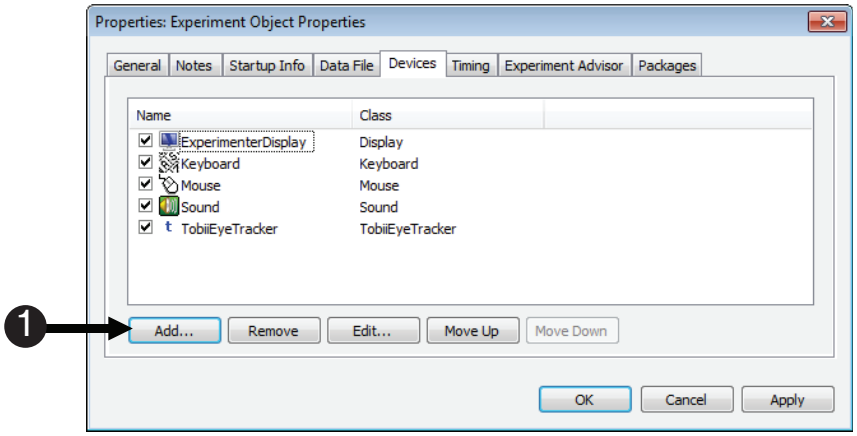
Add a Display Device to the experiment.

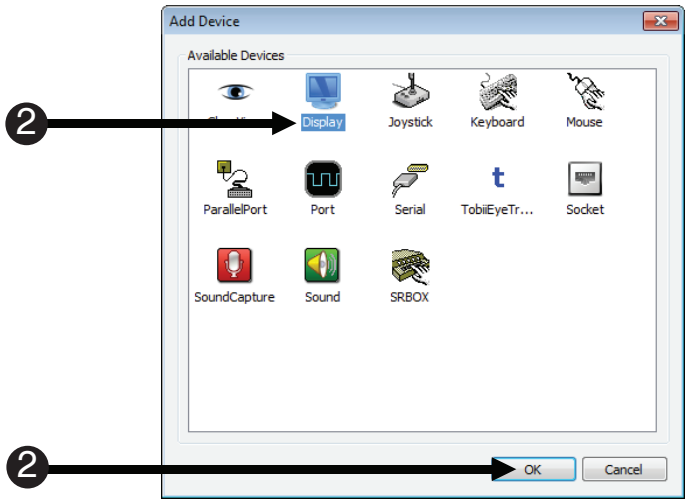
Select the Devices Tab of the Experiment Object Property Pages to add a second Display Device to the list of devices, and verify that it is the last device shown.

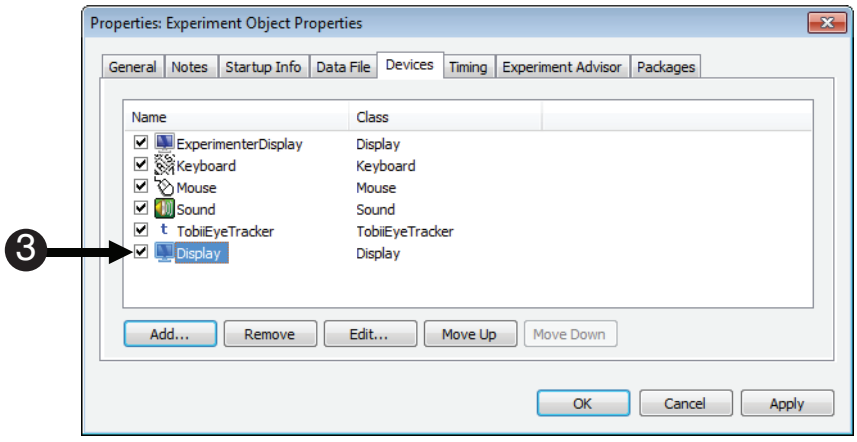
1) ***Click*** the **Add**… button.

2) ***Select*** the **Display Device** in the **Add Device dialog**. ***Click*** **OK** to close the **Add Device dialog**.

3) ***Verify*** the **Display Device** is listed last under the **Name** column and is checked.







Task 5: Edit the Properties of the second Display Device

Rename the new Display Device "TobiiDisplay" and edit the Display Index.

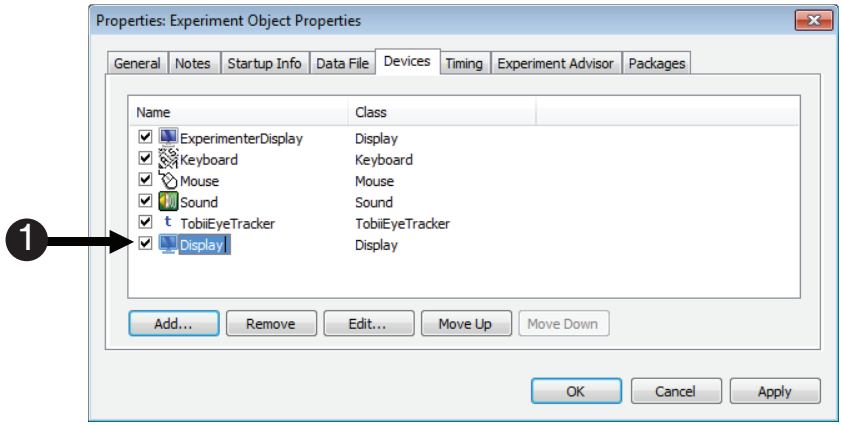
Now that you have added the second Display Device, it must be renamed, and assign the Display index one. This will indicate to E-Prime that the new monitor is the secondary display. For more information see 1.1.4. Support for Multiple Video Displays (Pages 8-9) in the New Features Guide.pdf.

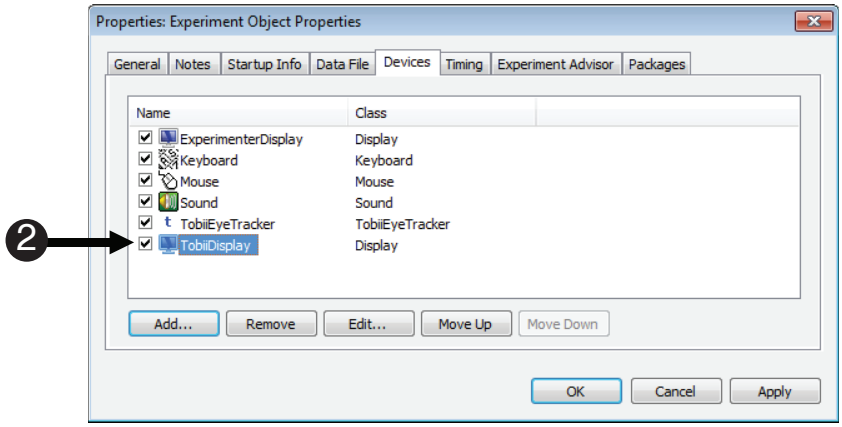
1) ***Click*** the **Display Device**. Then ***press*** **F2** to rename the object, **"TobiiDisplay"** and ***press*** **Enter** to ***accept*** the change.

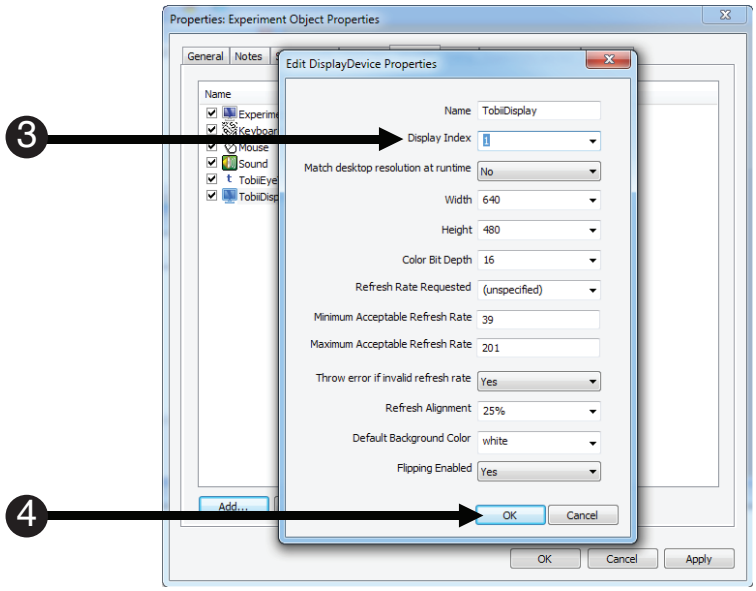
2) ***Double click*** on **"TobiiDisplay"** to open the **Edit Display Device Properties** page.

3) **Confirm** the **Display Index** is set to 1.

4) ***Click*** **OK** to apply these changes and to close the **Edit Display Device Properties** dialog.







Task 6: Move the second Display Device to a Higher Position in the Experiment Object Properties and Edit the TobiiEyeTracker Device

Relocate the TobiiDisplay device on the Properties Pages directly above the ExperimenterDisplay device.

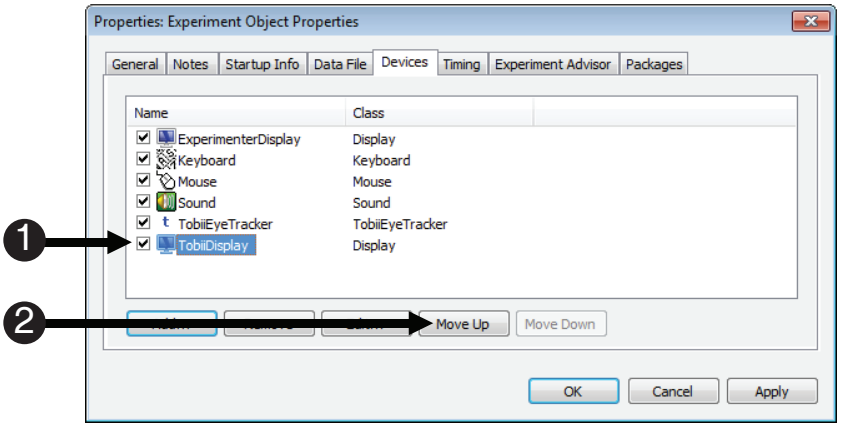
Before exiting the Devices tab of the Experiment Object Properties, move the TobiiDisplay display device directly above the ExperimenterDisplay display device, making it the default display device. The default display device will always show the dialogs for the subject and session number. Then set the display index to 1. The display index will differ for each machine and video card (see **Tutorial 6: Multiple Monitors: Creating a Participant Station, Page 135)**. Once this is achieved the TobiiEyeTracker Device will need to be set to the Tobii Display designating which display will perform the eye tracking.

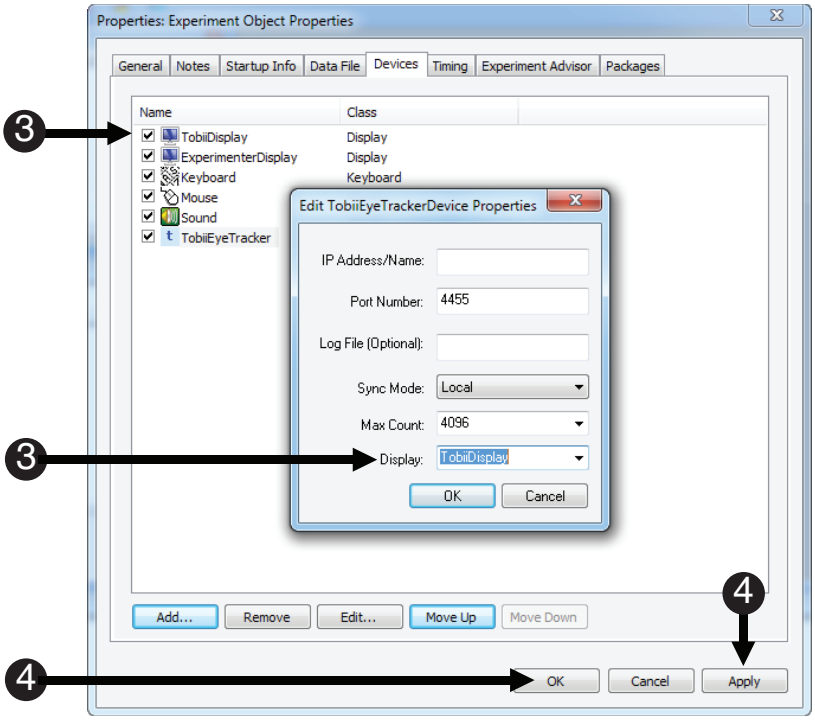
1) ***Highlight*** the **TobiiDisplay** display device.

2) ***Click*** on the **Move Up** button in the **Devices** tab of the **Experiment Object Properties** window until the **TobiiDisplay** display device is located at the top of the list.

3) ***Double click*** the **TobiiEyeTracker Device** to ***open*** the **Device** properties. ***Select*** **TobiiDisplay** from the dropdown, and ***click*** **OK.**

4) ***Click*** **Apply** to apply these changes and ***click*** **OK** to accept them and to dismiss the **Experiment Object Properties** window.





Task 7: Add an InLine Object to the SessionProc

Add an InLine to the SessionProc, under the Welcome TextDisplay Object and name the InLine AssignMonitor.

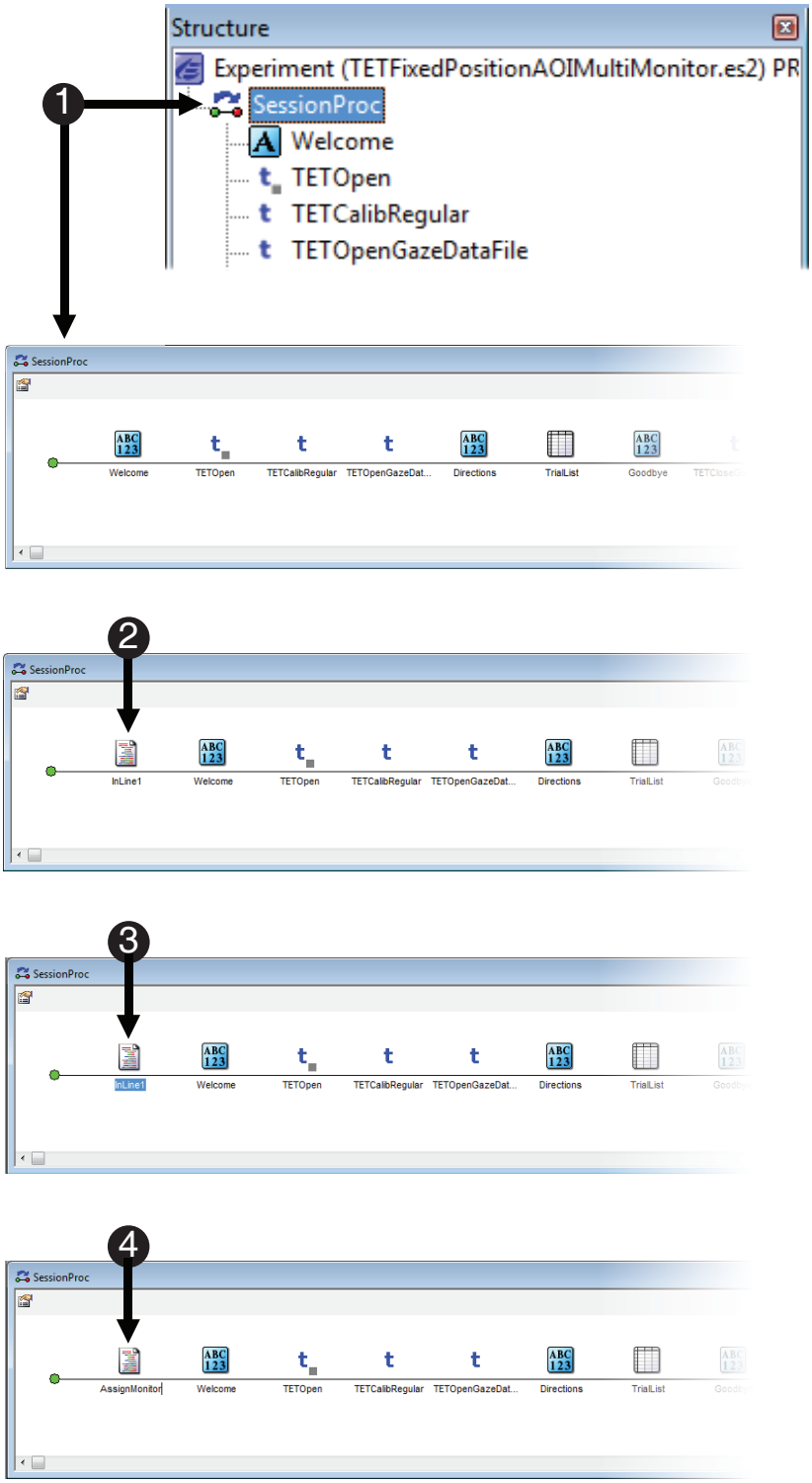
The InLine Object allows user script to be written directly into the experiment, allowing very specific functions to be placed into the experiment that the graphical user interface of E-Studio may not include.

1) ***Double click*** the **SessionProc Object** to open it in the workspace.

2) ***Drag*** a new **InLine Object** from the **E-Studio Toolbox** and ***drop*** it as the first object in the **SessionProc** procedure.
The object will be given a default name of **InLine1**.

3) ***Click*** on the **InLine1 Object** to select it then ***press*** **F2** to rename the object.
⚠ **NOTE:** *You may alternatively right click on the object and select Rename from the context menu.*

4) ***Type*** "**AssignMonitor**" as the new object name and then ***press*** **Enter** to accept the change.



Task 8: Add script to the InLine Object to display the desired items on each Monitor

Script the AssignMonitor InLine Object to direct which items from the experiment are to be displayed on either the TobiiDisplay or the ExperimenterDisplay.

The lines of user script that will be written into the AssignMonitor InLine Object allow the user to determine what part of the experiment will be seen by the participant (TobiiDisplay) or the experimenter (ExperimenterDisplay). This will be done in conjunction with the TrackStatus, MonitorCalibration, MonitorCalibrationResult and MonitorGazeReplay PackageCalls. We do not recommend using PackageCalls via InLine if the PackageCall is available through the E-Studio PackageCall GUI. For more information see **Chapter 3: Tobii PackageCall Reference, (Page 147)**.

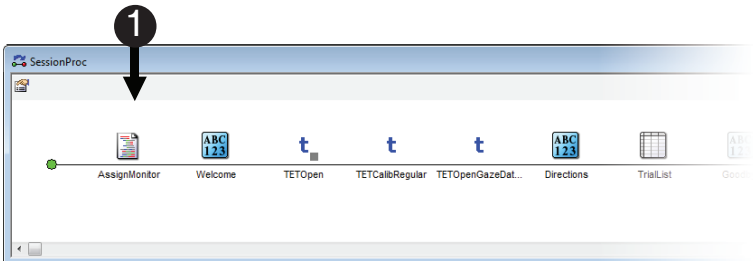
1) ***Double click*** on the **AssignDisplays InLine object** to open it.

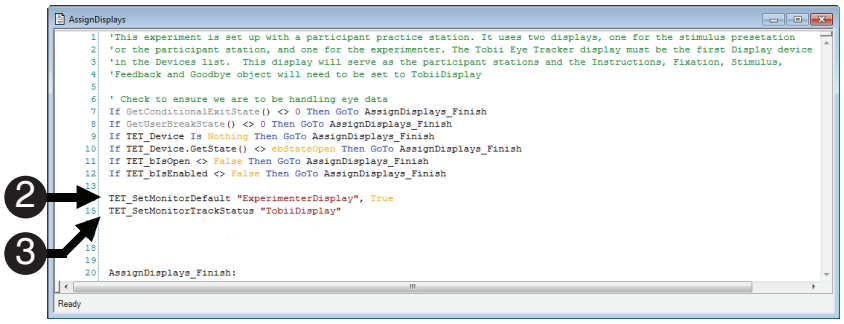
2) ***Set*** **ExperimenterDisplay** as the **Default Monitor** by ***typing*** **TET_SetMonitorDefault “ExperimenterDisplay”, True** into the first line of the **AssignMonitor InLine** object, then ***press*** **Enter**.

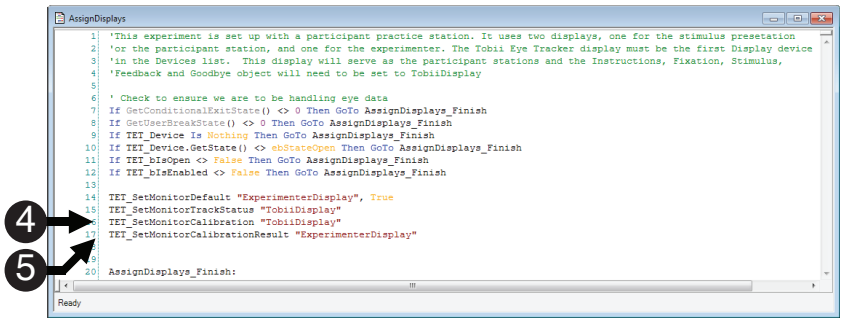
3) ***Set*** the **Monitor Track Status** to **TobiiDisplay** by ***typing*** **TET_SetMonitorTrackStatus “TobiiDisplay”** into the next line of the **AssignMonitor InLine object**, then ***press*** **Enter**.

4) ***Set*** the **Monitor Calibration** to run on **TobiiDisplay** by ***typing*** **TET_SetMonitorCalibration “TobiiDisplay”** into the next line of the **AssignMonitor InLine object**, then ***press*** **Enter**.

5) ***Set*** the results of the **Monitor Calibration** to run on **ExperimenterDisplay** by typing **TET_SetMonitorCalibrationResult “ExperimenterDisplay”** into the next line of the **AssignMonitor InLine object**, then ***press*** **Enter**.







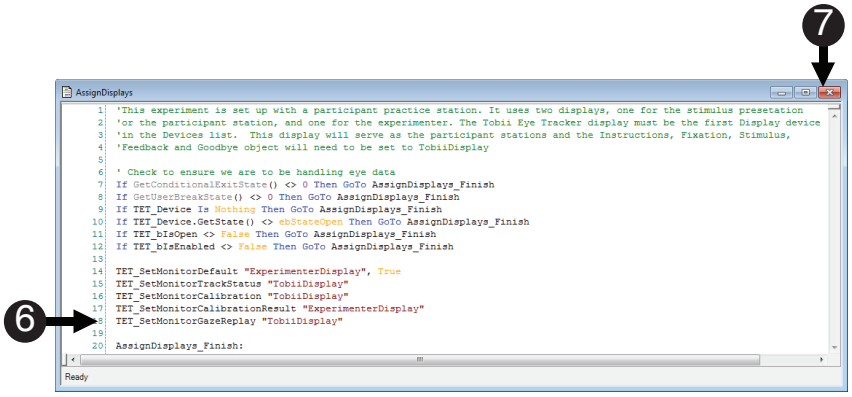
Task 8 (continued): Add script to the InLine Object to display the desired items on each Monitor

Script the AssignMonitor InLine Object to direct which items from the experiment are to be displayed on either the TobiiDisplay or the ExperimenterDisplay.

The lines of user script that will be written into the AssignMonitor InLine Object allow the user to determine what part of the experiment will be seen by the participant (TobiiDisplay) or the experimenter (ExperimenterDisplay). This will be done in conjunction with the TrackStatus, MonitorCalibration, MonitorCalibrationResult and MonitorGazeReplay PackageCalls.

6) ***Set*** the **Gaze Replay** to run on **TobiiDisplay** by ***typing*** **TET_SetMonitorGazeReplay "TobiiDisplay"** into the next line of the **AssignMonitor InLine object**, then ***press*** **Enter**.

7) ***Close*** the **AssignMonitor InLine object**.



Task 9: Run the Experiment

Run the experiment to verify that the eye tracker is working and to ensure the .gazedata file is written and the desired objects show up on the correct displays.

You have now completed the basic steps necessary to create an E-Prime Extensions for Tobii-enabled paradigm with dual monitor functionality. Before you run the task, be sure to hook up a second monitor. E-Prime Extensions for Tobii-enabled experiments can be run locally from E-Studio during development and testing. You should always fully test your experiment prior to scheduling actual participants or before using it to collect data.

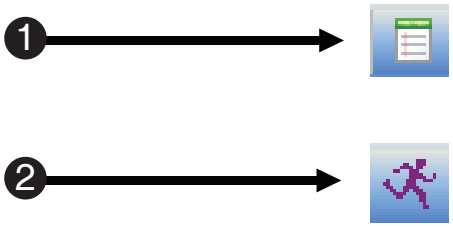
1) ***Press* Ctrl+S** to save your work before continuing. ***Click*** the **generate icon** or ***press* Ctrl+F7** to generate the script and ***check* it** for **errors**.

2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

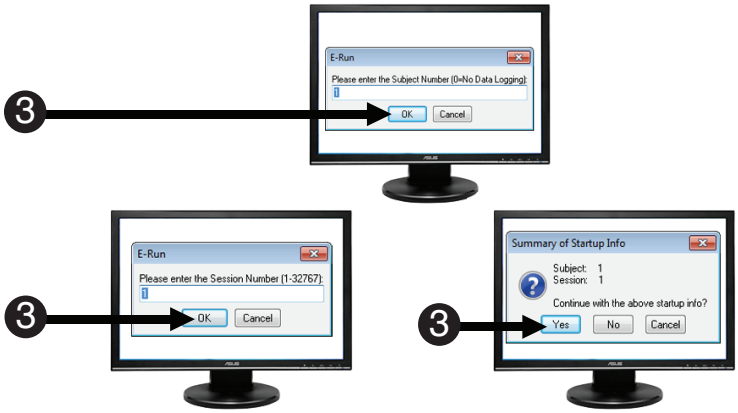
3) ***Click* OK** to accept the **default values** for **Subject Number**, **Session Number** and **Yes** for **Summary of Startup Info**.

4) ***Look*** at the **participant screen** to ***verify*** that the **eyes** are **stable** in the Track Status window. ***Ensure*** that there are **two white dots** which appear in the box and that the **bottom bar** is **green**. ***Press* Space** once both eyes are stable.

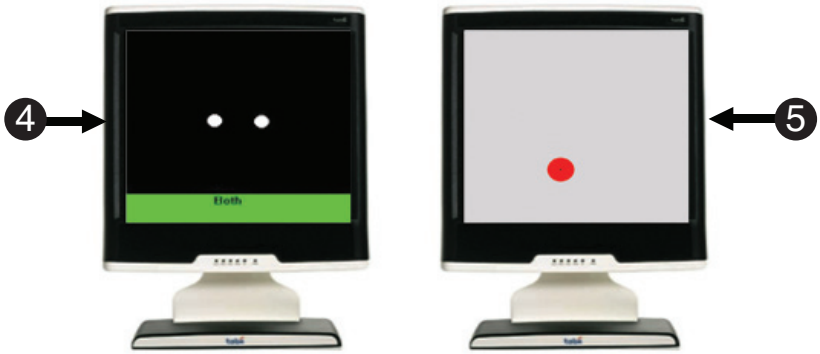
5) ***Run*** through the **calibration process** by ***following*** the **dots** on the **screen** with your **eyes**.



Experimenter View



Participant View



Task 9 (continued): Run the Experiment

Run the experiment to verify that the eye tracker is working and to ensure the .gazedata file is written and the desired objects show up on the correct displays.

You have now completed the basic steps necessary to create an E-Prime Extensions for Tobii-enabled paradigm with dual monitor functionality. Before you run the task, be sure to hook up a second monitor. E-Prime Extensions for Tobii-enabled experiments can be run locally from E-Studio during development and testing. You should always fully test your experiment prior to scheduling actual participants or before using it to collect data.

6) ***Accept*** the **calibration**.

⚠ **NOTE:** *Please refer to* **Tutorial 1, Task 17 (continued): Run the Experiment, (Page 44)** *for additional information about calibration.*

Experimenter View

Chapter 3: **Tobii PackageCall Reference**

3.1 Introduction

The PackageCall reference section contains the information you need to use the PackageCalls in an InLine. **We do not recommend using PackageCalls in InLine script if it is avoidable.** Please use the PackageCall object from the E-Studio toolbox if possible. If you must use any PackageCall(s) from an InLine you will need to add additional script to insure that your experiment will compile and to preserve the functioning of E-Prime 2.0.

The PackageCall reference section is divided into four parts. The first section explains how to make calls on the Tobii Eye Tracking (TET) device. The next section is devoted to PackageCalls that are used to calibrate from within your E-Prime experiment (also TET PackageCalls) and the last section covers the PackageCalls used to send markers about critical stimuli in E-Prime to Tobii Studio (the CV PackageCalls).

Each entry starts with the PackageCall listed at the top of the page. Under the Description heading is a description of what the PackageCall does, followed by the syntax you need to use in the InLine, and then the default parameters. The Parameter section lists the parameter name, followed by the data type and what the parameter is declared as. Then there is description for each parameter. Finally there is a Remarks section which includes information about the PackageCall. It can be anything from examples to value ranges.

The examples below are pulled from the SaveGazeData InLine included with your EET installation.

First, you will need to make sure the TET_Device exists in the context of your experiment and can manually handle the eye tracking data. If you set the "off" flag on the vState parameter in the TETOpen PackageCall when testing your experiment without the availability of the Tobii Eye Tracker your experiment will generate a run-time error unless you include this script in your InLine:

```
' Check to ensure we are to be handling eye data
If GetConditionalExitState() <> 0 Then GoTo SaveGazeData_Finish
If GetUserBreakState() <> 0 Then GoTo SaveGazeData_Finish
If TET_Device Is Nothing Then GoTo SaveGazeData_Finish
If TET_Device.GetState() <> ebStateOpen Then GoTo SaveGazeData_Finish
If TET_bIsOpen <> True Then GoTo SaveGazeData_Finish
If TET_bIsEnabled <> True Then GoTo SaveGazeData_Finish
```

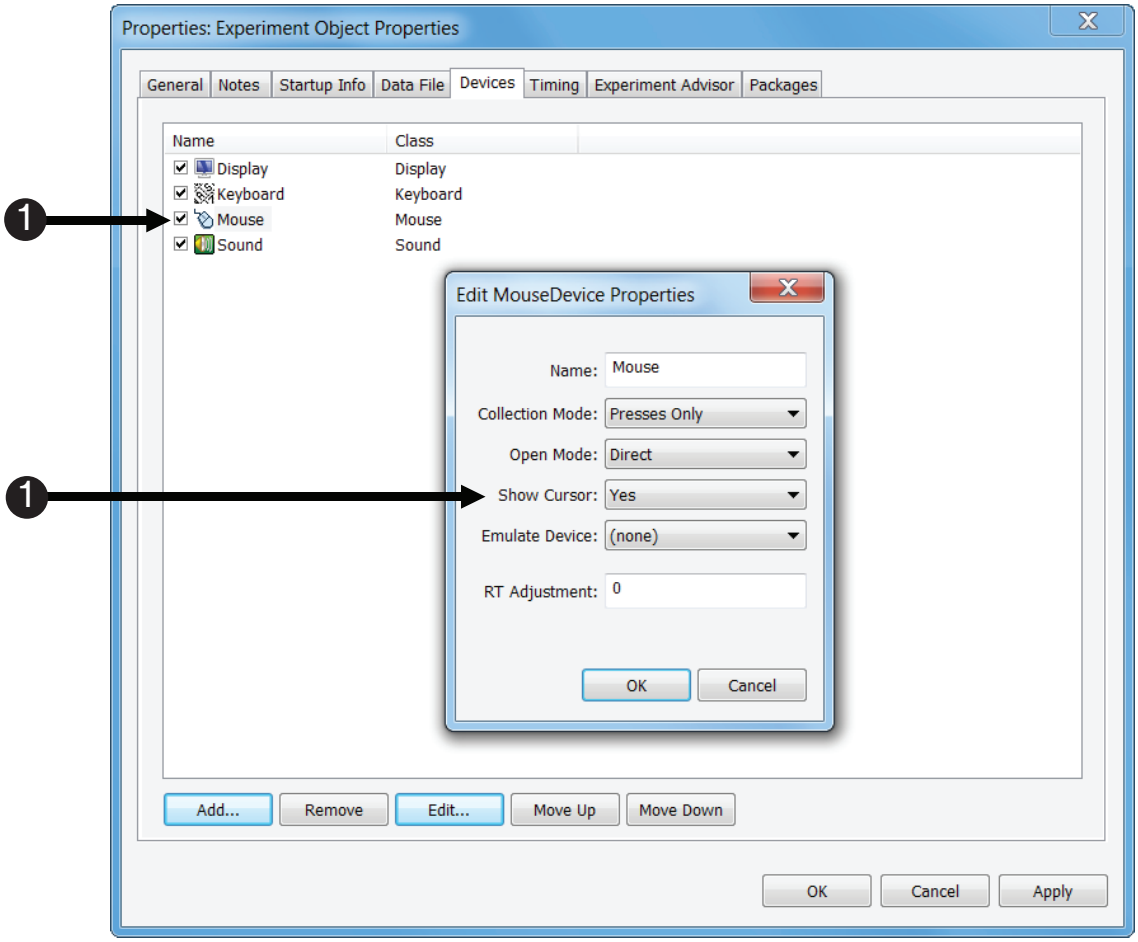
The second scenario you will need to account for is the use of Ctrl + Alt + Backspace or Ctrl + Alt + Shift. When the Ctrl + Alt + Backspace key combination is pressed during an experiment, the experiment terminates, but still produces an .edat2 file. The Ctrl + Alt + Shift key combination exits the experiment without producing an .edat2 file (e.g., during testing, when you don't care about getting a data file). The commands do not work when the PackageCalls are used from an InLine without the addition of the script below. This example is specifically for a For loop. See the SaveGazeData InLine in the VaryingPositionAOI.es2 experiment for use in a Do loop.

```
 ' User requested exit?
If GetConditionalExitState() <> 0 Then Exit For
If  GetUserBreakState() <> 0 Then Exit For
```

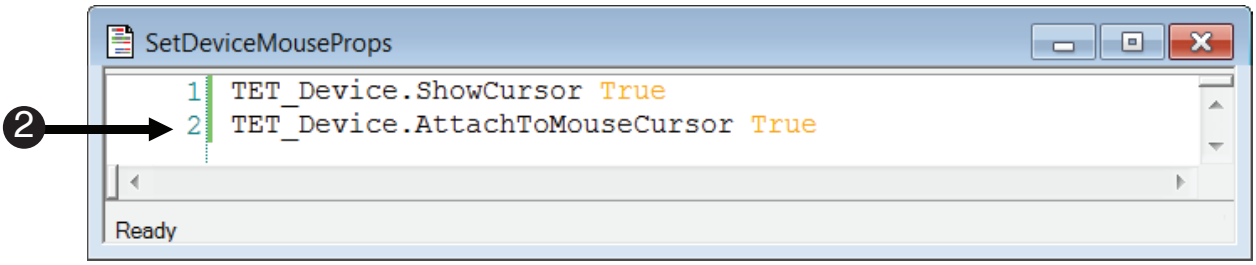
3.2 Calls to the Tobii Device

When collecting eye gaze data the eye tracker should be thought of as a device like a keyboard or mouse. Typically when the eye tracker is active the mouse cursor will not be visible to the participant. If you would like to make the cursor visible and active you will need to do two things. First, set the TET_Device.ShowCursor property to True. This can be done via the Devices Tab in the Experiment Property Pages or via In Line script. Second, set the TET_Device.AttachToMouseCursor = True in an InLine script.

1) ***Open*** the **Experiment Object's Property Pages** and ***go*** to the **Devices** tab. ***Double click*** the **mouse device**. ***Set*** the **Show Cursor** field to **Yes**.



2) In an **InLine**, ***set*** the **TET_Device.AttachToMouseCursor** **True**



TET_Open

Description

Opens the TobiiEyeTracker (TET) device.

Syntax

```
TET_Open c[,vState][,vConnect][,vHandlePreRelease]
```

Default Parameters

```
c
```

Parameters

c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vState As Variant

An optional parameter for the requested state of TET communications (“on”, “off”). If not specified this will default to “on”.

vConnect As Variant

An optional parameter to specify whether or not to immediately connect to the TET Server. If not specified, this will default to True.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks

None

TET_Connect

Description

Connects with the Tobii Eye Tracker server over the network.

Syntax

```
TET_Connect c[,vHandlePreRelease]
```

Default Parameters

```
c
```

Parameters

c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks

None

TET_ClearHistory

Description
Clears the history of accumulated eye tracking samples.

Syntax
```
TET_ClearHistory c[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_StartTracking

Description
Starts TET tracking eye data.

Syntax
```
TET_StartTracking c[,vClearHistory][,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vClearHistory As Variant
An optional parameter to specify whether or not the data in the TET history should be cleared when tracking is started (“True”, “False”). If not specified this will default to “True.”

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_TrackStatus

Description
Displays the track status window. The track status window is a black box. In the box, two white circles should appear. These circles are the visual representation of how the eye tracker sees your eyes.

Syntax
```
TET_TrackStatus c[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
The Slide sub-object needs to be called 'Fixation'.

TET_WaitForFixation

Description
Waits for the user to fixate on a designated object before continuing. When fixation is obtained, a key press will be simulated to terminate the input enabled on the specified Slide which can be used to obtain an RT related to the duration of the fixation period.

Syntax
```
TET_WaitForFixation c, theSlide, nMinFixationDuration [,vVisualFeedbackColor] [,vKey]
```

Default Parameters
```
c
```

Parameters
c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

theSlide As Slide

A Slide Object in the experiment which will be used to obtain the Fixation Object and provide visual feedback if enabled. The Slide Object provided must have a Sub-Object declared upon it named “Fixation”. The sub-object may be either a SlideImage or SlideText Object. The sub-object named Fixation must have its BorderWidth property set to a value greater than 0 if visual feedback is being used. You must also make sure to enable keyboard input on the Slide. Set the TimeLimit property to “(infinite)” and the EndAction property to “(none)”.

nMinFixationDuration As Long

The minimal amount of time specified in milliseconds of how long the participant’s gaze must remain on fixation before continuing.

vVisualFeedbackColor As Variant

An optional parameter to specify whether or not visual feedback will be provided when the participant is on fixation and to provide a color for the visual feedback. The feedback will be presented as a change in BorderColor on the sub-object. If you do not wish to see a persistent border around the Fixation object you can make the BorderColor property of the sub-object match the Slide’s background color. If not specified visual feedback presentation will not be enabled.

vKey As String

An optional parameter to specify a key to use to simulate a response when the participant has obtained fixation. If not specified the “1” key will be used.

Remarks
None

TET_StopTracking

Description
Stops the TET recording.

Syntax
```
TET_StopTracking c[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
After recording has stopped you must provide a valid filename through the TET interface in order to save the recording.

TET_Disconnect

Description
Disconnects the network connection with TETServer.

Syntax
```
TET_Disconnect c[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_Close

Description
Closes the TET PackageFile.

Syntax
```
TET_Close c[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_GazeReplay

Description
Visually replays the history of samples.

Syntax
```
TET_GazeReplay c, theSlide [,vStartTime][,vStopTime][,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

theSlide As Slide
A Slide object that will be drawn on screen before replay begins.

vStartTime As Variant
An optional parameter that creates a start time stamp for the first gaze point to be included in the replay. Typically, this is specified as the OnsetTime of an object in the experiment. If not specified, this will default to the first sample in the history.

vStopTime As Variant
An optional parameter that creates a stop time stamp for the last gaze point to be included in the replay. If not specified, this will default to the last sample in the history.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_OpenGazeDataFile

Description
Opens the tab delimited .gazedata file.

Syntax
```
TET_OpenGazeDataFile c[,vFilename][,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vFilename As Variant
An optional parameter for the filename of the file to be opened. If the parameter is not specified a filename will be created in the form of DataFile.BaseName. DataFile.BaseName defaults to [ExperimentName]-[Subject#]-[Session#].gazedata.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_ WriteGazeDataFile

Description
Writes a line of data to the .gazedata file and tracks the line count.

Syntax
```
TET_WriteGazeDataFile c, strLine [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

strLine As String
Tab delimited line of data to write out to the .gazedata file.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_ CloseGazeDataFile

Description
Closes the .gazedata file.

Syntax
```
TET_CloseGazeDataFile c[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

3.3 Calibration PackageCall Reference

The PackageCalls described in the following pages are for calibrating the Tobii Eye Tracker from within E-Prime. There are three types of calibration available. They are basic calibration, manual, and infant calibration.

Basic calibration is the calibration you are used to seeing in Tobii Studio. You have the ability to specify the number of points you wish to use for calibrating, visual control of the display elements, and the speed at which the dots move.

Manual calibration requires the operator to use the keyboard to mark the calibration points manually. You are able to edit the number of points and the visual elements of the display.

Infant calibration performs a manual calibration with an animation in place of the typical dot or ball, and has the added ability to play an attention grabbing animation that can be cued at any point during calibration.

The options of each calibration method are described in the pages that follow.

TET_CalibRegular

Description
This PackageCall performs the typical calibration seen in Tobii Studio. A red ball moves across the screen to nine different points where calibration data is collected automatically.

Syntax
```
TET_CalibRegular c [,vHandlePreRelease] [,vClearCalibration]
```

Default Parameters
```
c
```

Parameters
c As Context

The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

vClearCalibration As Variant

An optional parameter that specifies if the routine should delete all samples that are in the calibration. If not specified, this will default to True.

Remarks
The default number of points calibration points is nine, the default color of the ball is red (255,0,0), the default background color is grey (211, 211, 211), the default color on the center of the ball is black (0,0,0), and the default dot speed is medium slow (2). In order to change defaults see documentation for TET_CalibSetDefaultBackColor, TET_CalibSetDefaultCalibrationSpeed, TET_CalibSetDefaultDotColor, TET_CalibSetDefaultForeColor, and TET_CalibSetDefaultNumPoints.

TET_CalibInfant

Description
Performs infant calibration with a video.

The infant calibration is an interactive calibration mode where an operator uses the keyboard to control the calibration procedure. These are the available shortcuts:

- Right arrow key or left mouse button: Shows the calibration animation if not showing, or starts collecting calibration data at the current calibration point.
- Up arrow key: Shows the attention grabber animation.
- Left arrow key: Hides any animations showing.

Syntax
```
TET_CalibInfant c, strCalibrationAnimation.avi, strAttentionAnimation.avi
[,vHandlePreRelease] [,vClearCalibration]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

strCalibrationAnimation As String
The location of the calibration animation movie. See Remarks.

strAttentionAnimation As String
The file path of the attention animation movie. See Remarks.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

vClearCalibration As Variant
An optional parameter that specifies if the routine should delete all samples that are in the calibration. If not specified, this will default to True.

Remarks
The default number of calibration points is nine, and the default screen background is grey.
When designing an experiment, the use of a specific drive and/or folder (i.e. C:\Experiment\image.bmp) is typically discouraged. Instead, we recommend placing the files in the same folder as the experiment. This way, only the file name and its extension (e.g. “filename.wav”) must be specified, with no path or drive information in the E-Studio interface. If the files are not saved in the same folder as the .es2 or .ebs2 file, the experiment won’t be able to access them at runtime, and an error will occur. However, if you do need to place your files in a sub folder under where the experiment is, we typically encourage the use of the forward slash (/) instead of the backslash (\) to separate directories. In order to change defaults see documentation for TET_CalibSetDefaultBackColor, TET_CalibSetDefaultCalibrationSpeed, TET_CalibSetDefaultDotColor, TET_CalibSetDefaultForeColor, and TET_CalibSetDefaultNumPoints.

TET_CalibManual

Description
Performs manual calibration. The manual calibration is an interactive calibration mode where an operator uses the keyboard to control the calibration procedure.

Space key or left mouse button: Start collecting calibration data at the current calibration point.

Syntax
```
TET_CalibManual c [,vHandlePreRelease] [,vClearCalibration]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context. The routines in the TET PackageFile may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

vClearCalibration As Variant
An optional parameter that specifies if the routine should delete all samples that are in the calibration. If not specified, this will default to True.

Remarks
The default number of points calibration points is nine, the default color of the ball is red (255,0,0), the default background color is grey (211, 211, 211), the default color on the center of the ball is black (0,0,0), and the default dot speed is medium slow (2). In order to change the defaults see documentation for TET_CalibSetDefaultBackColor, TET_CalibSetDefaultCalibrationSpeed, TET_CalibSetDefaultDotColor, TET_CalibSetDefaultForeColor, and TET_CalibSetDefaultNumPoints.

TET_SetKeyboardDefault

Description
Sets the variable TET_kdDefaultKeyboard to the KeyboardDevice instance with the same Name as the "myKeyboard" parameter.

Syntax
```
Sub TET_SetKeyboardDefault(myKeyboard As String, throwErrorIfNonexsistent
As Boolean, Optional vHandlePreRelease As Variant
```

Default Parameters
"", True

Parameters
myKeyboard As String
The Name of the InputDevice that handles the default input operations of the Tobii Package File.

throwErrorIfNonexistent As Boolean
A default parameter that aborts the experiment if an InputDevice with Name myKeyboard is not found. Otherwise, the Routine will return regardless of whether or not the InputDevice was found and any error will be suppressed.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified, this will default to True.

Remarks
None

TET_SetMouseDefault

Description
Sets the MouseDevice to use if none is specified for CalibRegular, CalibInfant, and CalibManual. Will set the variable TET_kdDefaultMouse to the MouseDevice instance with the same Name as the "myMouse" parameter.

Syntax
```
Sub TET_SetMouseDefault(myMouse As String, throwErrorIfNonexsistent As Boolean, Optional vHandlePreRelease As Variant
```

Default Parameters
"", True

Parameters
myMouse As String
The Name of the InputDevice that handles the default mouse operations of the Tobii Package File.

throwErrorIfNonexistent As Boolean
A default parameter that aborts the experiment if an InputDevice with Name myKeyboard is not found. Otherwise, the Routine will return regardless of whether or not the InputDevice was found and any error will be suppressed.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified, this will default to True.

Remarks
None

Calibration PackageCall Chart

Below is a chart that explains which calibration routines can be used together and how they are implemented in E-Prime. The pages that follow contain the details of the routine's functionality.

PackageCall Name	Calibration Type			Implementation in E-Prime
	Regular	Manual	Infant	
TET_CalibAddPoint	X	X	X	InLine/PackageCall
TET_CalibCalculateAndSet	X	X	X	InLine/PackageCall
TET_CalibClear	X	X	X	InLine/PackageCall
TET_CalibGetDefaultBackColor	X	X	X	InLine
TET_CalibGetDefaultCalibrationSpeed	X			InLine
TET_CalibGetDefaultDotColor	X			InLine
TET_CalibGetDefaultForeColor	X	X		InLine
TET_CalibGetDefaultNumPoints	X	X	X	InLine
TET_CalibGetResult	X	X	X	InLine
TET_CalibIsValid	X	X	X	InLine
TET_CalibLoadFromFile	X	X	X	InLine
TET_CalibRemovePoints	X	X	X	InLine/PackageCall
TET_CalibSaveToFile	X	X	X	InLine
TET_SetMonitorDefault				InLine/PackageCall
TET_CalibSetDefaultBackColor	X	X	X	InLine
TET_CalibSetDefaultCalibrationSpeed	X			InLine
TET_CalibSetDefaultDotColor	X			InLine
TET_CalibSetDefaultForeColor	X	X		InLine
TET_CalibSetDefaultNumPoints	X	X	X	InLine
*TET_SetMonitorTrackStatus				InLine
*TET_SetMonitorCalibration				InLine
*TET_SetMonitorCalibrationResult				InLine
*TET_SetMonitorGazeReplay				InLine

*Refer to 3.4 Multiple Monitor PackageCalls Reference, (Page 188)

TET_CalibAddPoint

Description
This is a blocking function that will start collecting gaze data samples for a specific gaze target point. The samples are added to the set of new data in the calibration under construction.

Syntax
```
TET_CalibAddPoint x, y, nrofdata, interval[,vHandlePreRelease]
```

Default Parameters
```
0.5, 0.5, 6, 0
```

Parameters
x As Double
Horizontal position of target point ranging from 0 to 1 where 0 is leftmost position and 1 is rightmost from the eye tracking participant's point of view.

y As Double
Vertical position of target point ranging from 0 to 1 where 0 is the topmost position and 1 is bottommost.

nrofdata As Long
Hint to the system of how many good samples to collect for this calibration point. A value of 6 (default) is considered optimal for many eye tracking environments.

interval As Long
Interval in milliseconds between calls to the callback function will be called. If set to 0 (default), the callback will never be called for timer reason. Note that there's a lower limit of this interval dependent of the system the application runs on. The interval is not guaranteed to be followed strictly.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_ CalibCalculateAndSet

Description
This call reads the calibration under construction and writes it to the calibration in use, which is immediately used by the eye tracking algorithm. Any old data is replaced.

Syntax
```
TET_CalibCalculateAndSet[,vHandlePreRelease]
```

Default Parameters
None

Parameters
vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_CalibClear

Description
This call deletes all samples that are in the calibration under construction. It is a good practice to always start a new calibration with this function call.

Syntax
```
TET_CalibClear [,vHandlePreRelease]
```

Default Parameters
None

Parameters
vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_CalibGetDefaultBackColor

Description
Gets the default calibration background color.

Syntax
`TET_CalibGetDefaultBackColor`

Default Parameters
None

Parameters
None

Remarks
None

TET_CalibGetDefaultCalibrationSpeed

Description
Gets the default speed for ball movement in regular calibration. Range is from 1-5 (inclusive).

Syntax
```
TET_CalibGetDefaultCalibrationSpeed
```

Default Parameters
None

Parameters
None

Remarks
None

TET_CalibGetDefaultDotColor

Description
Gets the default foreground color for the ball dot in regular calibration (returns string).

Syntax
```
TET_CalibGetDefaultDotColor
```

Default Parameters
None

Parameters
None

Remarks
None

TET_CalibGetDefaultForeColor

Description
Gets the default foreground color for the ball (in regular calibration) and the line (in manual calibration) in a string format.

Syntax
```
TET_CalibGetDefaultForeColor
```

Default Parameters
None

Parameters
None

Remarks
None

TET_CalibGetDefaultNumPoints

Description
Gets the default number of calibration points for the calibration (returns long). The default number of calibration points is nine.

Syntax
```
TET_CalibGetDefaultNumPoints
```

Default Parameters
None

Parameters
None

Remarks
None

TET_CalibGetResult

Description
This function gets information about a calibration for a quality inspection. The inspection is easiest implemented as viewing the data to manually approve it or take necessary action to improve it. Provided are target points, the resulting mapped points, and an indication whether or not the point was discarded, by the eye tracker. The result is from the current calibration in use. Returns calibration information as a TobiiCalibAnalyzeDataCollection Object.

Syntax
`TET_CalibGetResult`

Default Parameters
None

Parameters
None

Remarks
A TobiiCalibAnalyzeDataCollection Object is a collection of TobiiCalibAnalyzeData type.
The TobiiCalibAnalyzeData type consists of these values:

truePointX (Double) Horizontal position of the gaze point target where it was displayed to the participant.

truePointY (Double) Vertical position of the gaze point target where it was displayed to the participant.

leftMapX (Double) Horizontal position of the mapped point for the left eye.

leftMapY (Double) Vertical position of the mapped point for the left eye.

leftValidity (Long) Left eye, (-1) – was not found, (0) – found but not used, (1) – used.

leftQuality (Double) Left eye quality measurement (feature to come, not implemented).

rightMapX (Double) Horizontal position of the mapped point for the right eye.

rightMapY (Double) Vertical position of the mapped point for the right eye.

rightValidity (Long) Right eye, (-1) – was not found, (0) – found but not used, (1) – used.

rightQuality (Double) Right eye quality measurement (feature to come, not implemented).

The purpose of CalibGetResult() is to inspect the data to see how good the samples were in order to approve the calibration to improve it or to redo it.

TET_CalibGetResult continued

The data consists of the true point (where the participant gazed at) and where the sample was mapped to, using the calibration that was partly based on the sample itself.

Note that some points are not part of the calibration. It is those for which the eye was found by the eye tracker or those for which the eye tracker found them but discarded them for other reasons.

The leftValidity and rightValidity values indicate whether it was used or not.

Example

```
Dim tc As TobiiCalibAnalyzeDataCollection
Set tc=TET_CalibGetResult()

If tc Is Nothing Then
    Err.Raise TET_ERR_CORE_DEVICE_CANT_OPEN, , "Unable to gather any
calibration results. Please consider restarting this experiment."
Else
    For i = 1 To tc.Count

        Dim tdat As TobiiCalibAnalyzeData
        Set tdat = tc(i)

       Debug.Print "x: " + CStr(tdat.TruePointX) + " y: " + CStr(tdat.
TruePointY)
    Next i
End If
```

TET_CalibIsValid

Description
This call returns (True, False) whether a valid calibration is currently present in the eye tracker.

Syntax
None

Default Parameters
None

Parameters
None

Remarks
None

TET_CalibLoadFromFile

Description
Sets a calibration stored in a file. A file that has been stored by `TET_CalibSaveToFile` is read and written to the calibration in use and to the calibration under construction. Any old data is overwritten (boolean).

Syntax
```
TET_CalibLoadFromFile strFilename
```

Default Parameters
“ ”

Parameters
strFilename As String
Path and filename of the calibration file to load.

Remarks
None

TET_CalibRemovePoints

Description
The purpose of this call is to improve the calibration under construction by removing bad samples. It enables the deletion of samples for all calibration points within a specified area. The area is given by a circle, a point, and a radius position when TET_CalibAddPoint is called.

Syntax
```
TET_CalibRemovePoints eye, x, y, dRadius[,vHandlePreRelease]
```

Default Parameters
```
3, x, y, dRadius
```

Parameters
eye As Long
The eye(s) to remove calibration samples for.

Enum Name	Value	Description
TET_EYE_LEFT	1	Left eye from participant (person being tracked) point of view
TET_EYE_RIGHT	2	Right eye from participant (person being tracked) point of view
TET_EYE_BOTH	3	Both eyes

x As Double
Horizontal position of circle center ranging from 0 to 1 where 0 is leftmost position and 1 is rightmost from the eye tracking participant's point of view.

y As Double
Vertical position of circle center ranging from 0 to 1 where 0 is topmost position and 1 is bottommost from the eye tracking participant's point of view.

dRadius As Double
The distance from point (x,y) defining the circle from within which the calibration points will be removed. Same unit as x and y.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
None

TET_CalibSaveToFile

Description
Saves the current calibration in use to file for a future reuse. Call `TET_CalibLoadFromFile` at any time to load the calibration to the eye tracker (returns boolean).

Syntax
```
TET_CalibSaveToFile strFilename
```

Default Parameters
“ ”

Parameters
strFilename As String
Path and filename of the calibration file to save.

Remarks
If the file exists, the old one is overwritten.

TET_CalibSetDefaultBackColor

Description
Sets the default color for the calibration background (returns string).

Syntax
```
TET_CalibSetDefaultBackColor myBackColor
```

Default Parameters
“ ”

Parameters
myBackColor As String
The color to use for the calibration background. You can use Color.Gray or the string should be in the format used with the CColor function:
“211,211,211”
“gray”

Remarks
If this accessor is not called in an experiment, the implied default value for the myBackColor parameter is “211,211,211”.

TET_CalibSetDefaultCalibrationSpeed

Description
Sets the default speed for ball movement in regular calibration.

Syntax
```
TET_CalibSetDefaultCalibrationSpeed myCalibrationSpeed
```

Default Parameters
“ ”

Parameters
myCalibrationSpeed As Long

The speed for ball movement. Accepted values range from 1 to 5:
1: Slow
2: Medium Slow
3. Medium
4. Medium Fast
5. Fast

Remarks
If the value in myCalibration is not one of the accepted speed values, a ebERR_ INDEXOUTOFBOUNDS error will be thrown.

If this accessor is not called in an experiment, the implied default value for the myCalibrationSpeed parameter is 2 (Medium Slow).

TET_CalibSetDefaultDotColor

Description
Sets the default color for the ball dot in regular calibration.

Syntax
```
TET_CalibSetDefaultDotColor myDotColor
```

Default Parameters
“ ”

Parameters
myDotColor As String
The color to use for the ball dot in regular calibration The string should be in the format used with the CColor function:
“0,0,0”
“black”

Remarks
If this accessor is not called in an experiment, the implied default value for the myDotColor parameter is “0,0,0”.

TET_CalibSetDefaultForeColor

Description
Sets the default foreground color for the ball (in regular calibration) and the line (in manual calibration).

Syntax
```
TET_CalibSetDefaultForeColor myForeColor
```

Default Parameters
“ ”

Parameters
myForeColor As String
The foreground color to use for the ball (in regular calibration) and the line (in manual calibration). The string should be in the format used with the CColor function:
“255,0,0”
“red”

Remarks
If this accessor is not called in an experiment, the implied default value for the myForeColor parameter is “255,0,0” (red).

TET_CalibSetDefaultNumPoints

Description
Sets the default number of calibration points for the calibration.

Syntax
```
TET_CalibSetDefaultNumPoints myNumOfCalibPoints
```

Default Parameters
```
9
```

Parameters
myNumOfCalibPoints As Long
The number of calibration points to display.
The accepted values are 2, 5, and 9.

Remarks
If myCalibration is not one of the accepted values, an ebERR_INDEXOUTOFBOUNDS error will be thrown.

If this accessor is not called in an experiment, the implied default value for the MyNumOfCalibPoints parameter is nine.

3.4 Multiple Monitor PackageCalls Reference

This section of the documentation deals with the PackageCalls used to view the experiment on multiple monitors. These PackageCalls have the ability to determine which screen the calibration validation and participant start up information is viewed on.

TET_SetMonitorDefault

Description
Sets the DisplayDevice to use if none is specified for Calibration, CalibrationResult, Experiment, GazeReplay, or TrackStatus.

Syntax
```
TET_SetMonitorDefault “myMonitor”, True [,vHandlePreRelease]
```

Default Parameters
```
“”, True
```

Parameters
myMonitor As String
The monitor to use for Tobii calibration.

throwErrorIfNonexistent As Boolean
A default parameter that aborts the experiment if a DisplayDevice with Name myMonitor is not found. Otherwise, the Routine will return regardless of whether or not the DisplayDevice was found and any error will be suppressed.

vHandlePreRelease
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified, this will default to True.

Remarks
If myMonitor represents a non-existent or invalid display device, a ERR_DISPLAY_INVALID_INDEX error will be thrown.

If this accessor is not called in an experiment, the implied default value for the myMonitor parameter will be the first monitor found.

If TrackStatus, Calibration, Calibration Result, or Gaze Replay doesn’t have a specific display device assigned, it will use whatever is set in the default.

TET_SetMonitorTrackStatus

Description
Sets the DisplayDevice used for the Tobii track status screen.

Syntax
```
TET_SetMonitorTrackStatus “myMonitor”
```

Default Parameters
“”

Parameters
myMonitor As String
The monitor to use for the track status screen.

Remarks
If myMonitor represents a non-existent or invalid display device, a ERR_DISPLAY_INVALID_INDEX error will be thrown.

If this accessor is not called in an experiment, the implied default value for the myMonitor parameter is the first monitor detected.

TET_SetMonitorCalibration

Description
Sets the DisplayDevice used for the Tobii calibration screen.

Syntax
```
TET_SetMonitorCalibration “myMonitor”
```

Default Parameters
“”

Parameters
myMonitor As String
The monitor to use for the calibration screen.

Remarks
If myMonitor represents a non-existent or invalid display device, a ERR_DISPLAY_INVALID_INDEX error will be thrown.

If this accessor is not called in an experiment, the implied default value for the myMonitor parameter is the first monitor detected.

TET_SetMonitorCalibrationResult

Description
Sets the DisplayDevice used for the Tobii calibration result screen.

Syntax
```
TET_SetMonitorCalibrationResult “myMonitor”
```

Default Parameters
“ ”

Parameters
myMonitor As String
The monitor to use for the calibration results screen.

Remarks
If myMonitor represents a non-existent or invalid display device, a ERR_DISPLAY_INVALID_INDEX error will be thrown.

If this accessor is not called in an experiment, the implied default value for the myMonitor parameter is the first monitor detected.

TET_SetMonitorGazeReplay

Description
Sets the DisplayDevice used for the Tobii gaze replay screen.

Syntax
```
TET_SetMonitorGazeReplay “myMonitor”
```

Default Parameters
“”

Parameters
myMonitor As String
The monitor to use for the gaze replay screen.

Remarks
If myMonitor represents a non-existent or invalid display device, a ERR_DISPLAY_INVALID_INDEX error will be thrown.

If this accessor is not called in an experiment, the implied default value for the myMonitor parameter is the first monitor detected.

3.5 Event Marker PackageCalls Reference

This section of the documentation deals with the PackageCalls used to send information to Tobii Studio. These PackageCalls have the ability to send markers to Tobii Studio regarding critical stimuli that can be used to create scenes and segment data for analysis purposes.

ClearView_Close

Description

Closes the ClearView Package File.

Syntax

```
ClearView_Close c[,vHandlePreRelease]
```

Default Parameters

c

Parameters

c As Context

The current experiment context. The routines in the CV Package File may optionally use the experiment context to retrieve the current values of attributes stored in the context.
The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to, “True.”

Remarks

None

ClearView_Init

Description

Initializes the ClearView system.

Syntax

```
ClearView_Init c, strState [,vHandlePreRelease]
```

Default Parameters

c, on

Parameters

c As Context

The current experiment context. The routines in the CV Package File may optionally use the experiment context to retrieve the current values of attributes stored in the context.
The context is typically the first parameter of every package file routine.

strState As String

The requested state of ClearView communications (“on”, “off”).

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks

None

ClearView_Open

Description

Makes the ClearView system available to the experiment.

Syntax

```
ClearView_Open c[,vState][,vConnect][,vHandlePreRelease]
```

Default Parameters

c

Parameters

c As Context

The current experiment context. The routines in the CV Package File may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

vState As Variant

The requested state of ClearView communications ("on", "off").

vConnect As Variant

Optional parameter to specify whether or not to immediately connect to the ClearView application. If not specified this will default to True.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to, “True.”

Remarks

None

ClearView_SendGenericEvent

Description

Sends a generic event information to the ClearView log. This is normally interpreted by ClearView as the keypress controlling event logging during recordings.

Syntax

```
ClearView_SendGenericEvent c[,vHandlePreRelease]
```

Default Parameters

c

Parameters

c As Context

The current experiment context. The routines in the CV Package File may optionally use the experiment context to retrieve the current values of attributes stored in the context.
The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks

None

ClearView_SendLogEvent

Description

Sends event information to the ClearView log.

Syntax

```
ClearView_SendLogEvent c,"EventText"[,vHandlePreRelease]
```

Default Parameters

c

Parameters

c As Context

The current experiment context. The routines in the CV Package File may optionally use the experiment context to retrieve the current values of attributes stored in the context. The context is typically the first parameter of every package file routine.

strEventText As String

Text string that will appear in the event data file exported from ClearView. The String can be used to send multiple events using the “|” pipe key as a delimiter. You may also use the Tab to force data into separate output columns.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks

ClearView_SendLogEvent c, “SceneStart myScene|My second scene|[stimulus]|SceneFour”
ClearView_SendLogEvent c, “SceneStop myScene|My second scene|[stimulus]|SceneFour”

ClearView_StartRecording

Description

Starts ClearView recording. The data from E-Prime is sent via ethernet port to Tobii Studio in about 10 ms. The video information is sent to Tobii Studio via the video capture card with a delay of less than approximately 40 ms.

Syntax

```
ClearView_StartRecording c[,vHandlePreRelease]
```

Default Parameters

c

Parameters

c As Context

The current experiment context. The routines in the CV Package File may optionally use the experiment context to retrieve the current values of attributes stored in the context.
The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks

None

ClearView_StopRecording

Overview

Stops ClearView recording. After recording has stopped you must provide a valid filename through the ClearView interface in order to save the recording.

Syntax

```
ClearView_StopRecording c[,vHandlePreRelease]
```

Default Parameters

c

Parameters

c As Context

The current experiment context. The routines in the CV Package File may optionally use the experiment context to retrieve the current values of attributes stored in the context.
The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks

None

ClearView_Uninit

Overview

Uninitializes ClearView.

Syntax

```
ClearView_Uninit c[,vHandlePreRelease]
```

Default Parameters

c

Parameters

c As Context

The current experiment context. The routines in the CV Package File may optionally use the experiment context to retrieve the current values of attributes stored in the context.
The context is typically the first parameter of every package file routine.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks

None

Appendix A: **Locate Your IP Address and Ping Another Computer**

When you are trying to determine that one machine is connected to another machine, the simplest way to do this is to ping one machine from the other machine. In order to ping another computer first get the IP Address of the computer you want to talk to and then go to the other machine you want to communicate with. Open a command prompt and type ping “IP Address”, and then Enter. If the machines can communicate you will get reply messages. If the machines cannot communicate you will get error messages.

1) To ***find*** the **IP address** of the **Tobii Studio machine** on a **Windows machine**, go to **Start Menu** > **Run**.

⚠ **NOTE:** *The monitor serial number can be used to ping the Tobii monitor. When doing so, you must include the .local.*

2) In the **Run** dialog, ***type*** “**cmd**” and ***click*** **OK**.

3) A **command prompt** (a dialog box with a black background and white text) will ***open***.

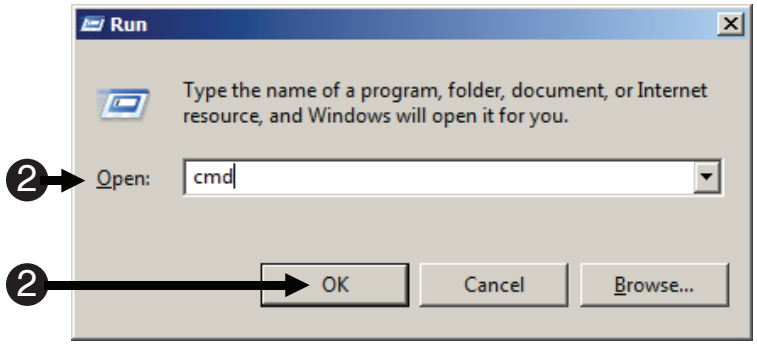
4) The **cmd.exe** will ***display*** a prompt, ***type*** “**ipconfig**”, ***press*** **Enter**.

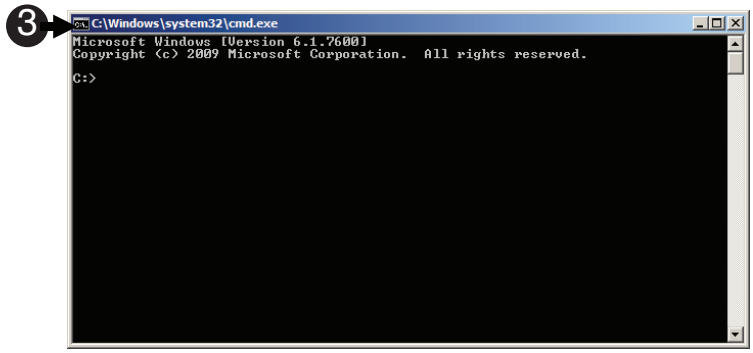
5) The **IP Address** will be labeled in the **command prompt window**.

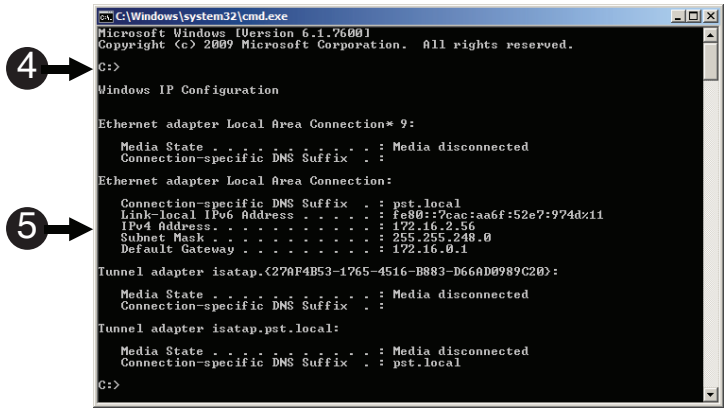
⚠ **NOTE:** *If you are using the T-Series, you can use the **monitor serial number ‘monitor serial number. local.’***

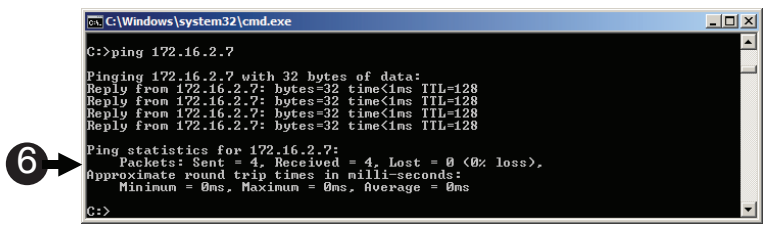
⚠ **NOTE:** *Not all IP Addresses will look the same. In this example, the **IP address is IPv4 Address.***

6) At the prompt, ***type*** **"ping 172.16.2.7 (your IP address)"**, ***press*** **Enter**. If the **machines** are able to ***communicate*** you will get **reply messages**. If the **machines** cannot ***communicate*** you will get **error messages**.







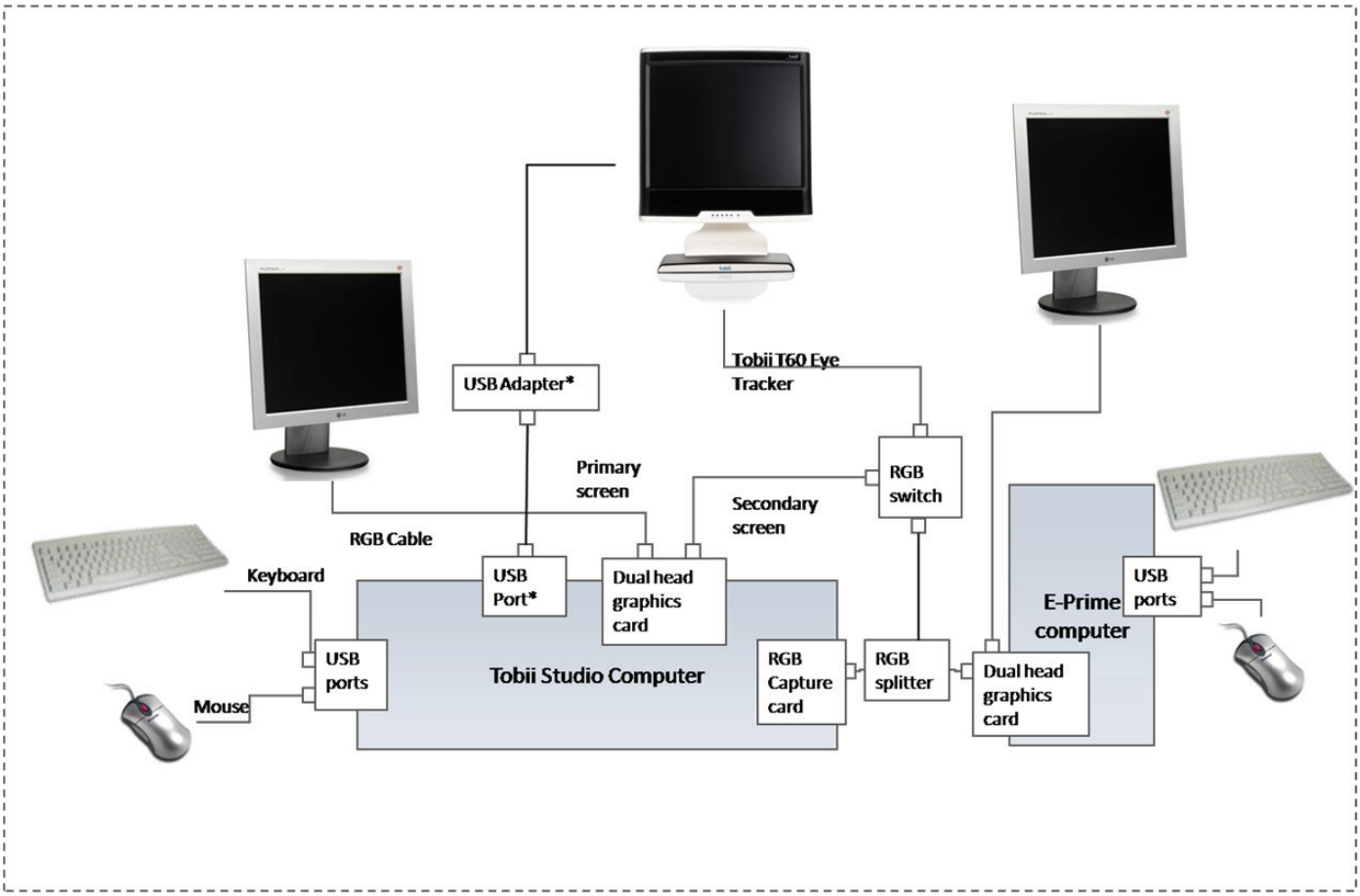


Appendix B: **Hardware Configurations**

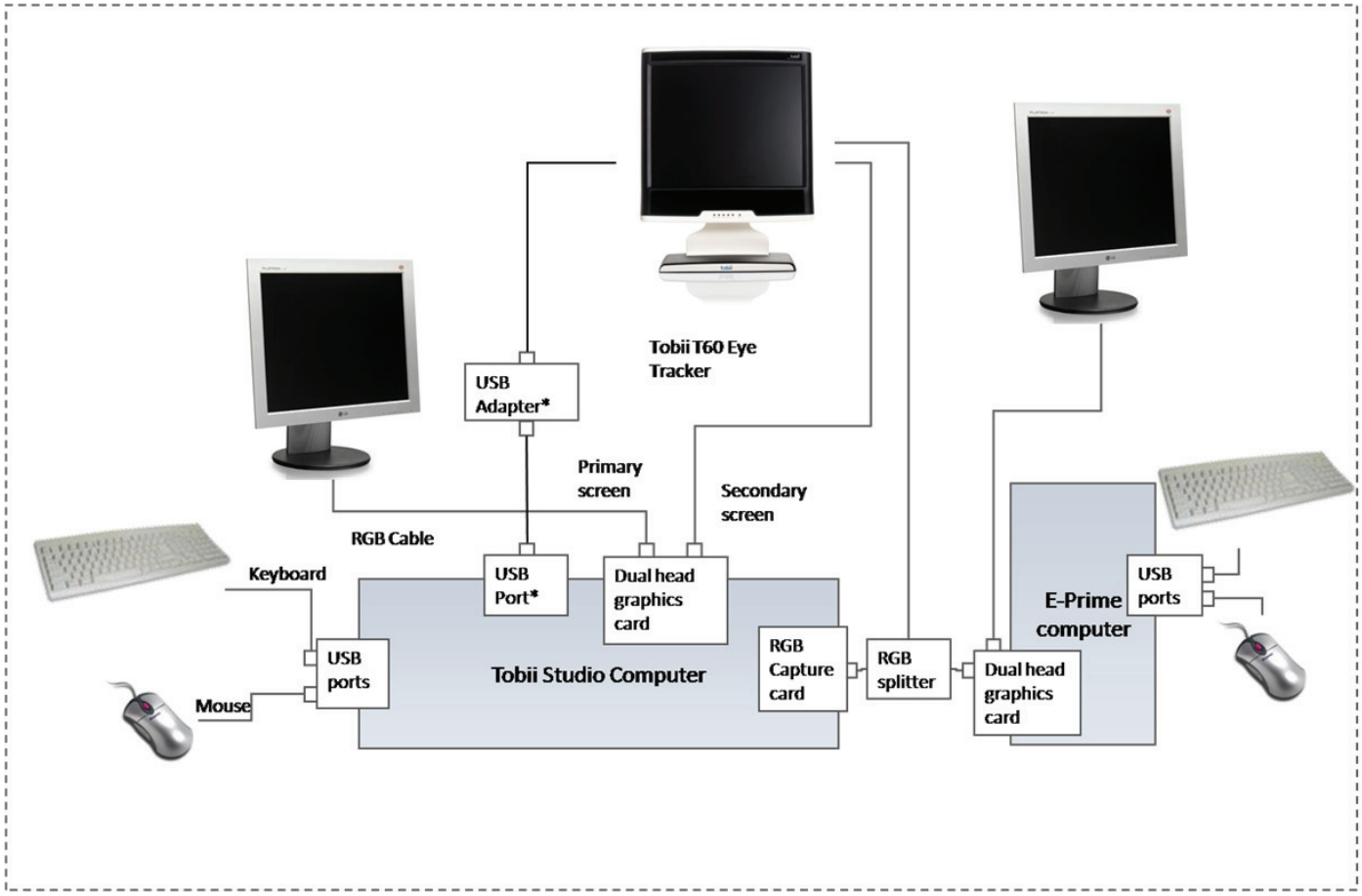
Please refer to your eye tracker manual for more hardware configurations and troubleshooting information.

⚠ **NOTE:** *Bonjour must be installed on both the Tobii Studio and the E- Prime computer machine. This can be achieved by installing the Tobii browser included in your Tobii Studio installation, or downloading Bonjour from the web.*

DualHead Graphics Card External Switch



DualHead Graphics Card Internal Switch



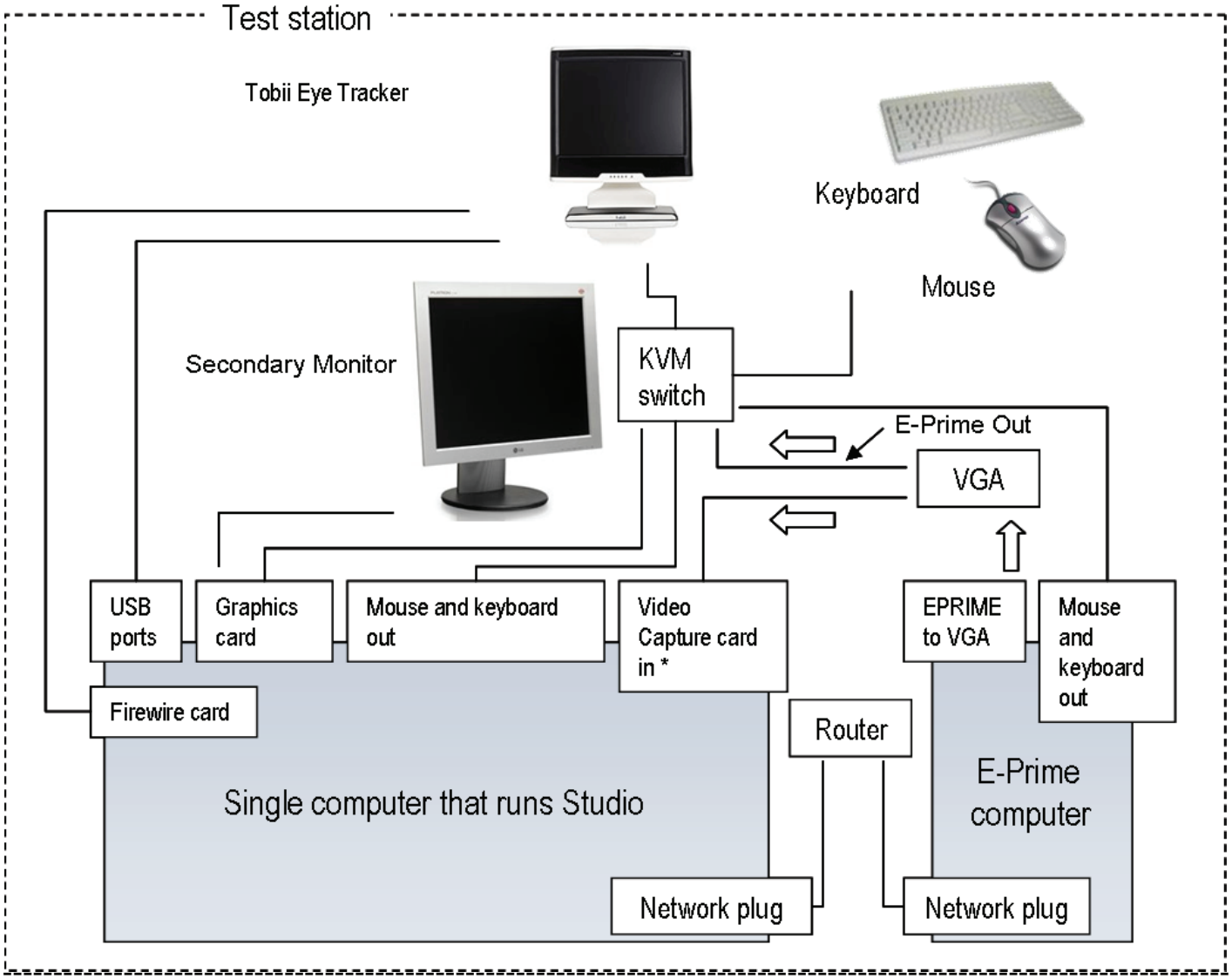
- The network cable can be connected directly to the computer, to a network or switch (do not use a hub) to which the computer is also connected, or to the USB adapter delivered with the eye tracker.

- There are two monitor ports on the T60/T120 and TX300 eye tracker: one VGA and one DVI connector. If all cables used in the setup are VGA cables, a VGA-DVI adapter can be used. The image from the stimulus presentation screen will be recorded in Tobii Studio and should be shown on the eye tracker screen for the test participant.

- To switch between the inputs, press the source button on the T60/T120 and TX300 Eye Tracker.

- To show the image from the stimulus computer on a separate supervisor screen, use a splitter between the stimulus computer and the stimulus presentation screen. A splitter can also be used to split the signal to the RGB capture card and the T60/T120 and TX300 Eye Tracker monitor if a laptop or other computer with a single head graphics card is used.

Configuration for LiveViewer with Tobii Studio



Appendix C: **Timing and Synchronization**

Writing to the Buffer and Buffer Size

It is important to make sure all of the data from the experiment is being written to the .gazedata file. In order to do this we recommend that you use the SaveGazeData InLine included in the samples that are installed with your EET installation and modify it to fit your experiment. Keep in mind when you modify the file there are two critical things that you must consider when collecting data: buffer size and when the data is written to the buffer.

We recommend that you begin to fill the buffer when you use the TETStartTracking PackageCall and stop filling the buffer when you call TETStopTracking PackageCall. Contained in the TETStartTracking PackageCall there is an option to clear the buffer when the call is made. This option ensures the buffer is empty and ready to be filled with data every time TETStartTracking PackageCall is called. The option is set to “clear” by default, but if you need to clear the buffer at other times during your experiment you can also use the TETClearHistory PackageCall in the experiment to do this manually.

The data will also need to be written to the .gazedata file during a non-critical timing moment like at the end of a trial to prevent disruption of the experiment timing. This means that you will need to choose a buffer size that is at minimum as long as your trial in order to be able to collect all of the eye gaze data for that trial. Contained in the table below are Eye Tracker Speeds (Hz), Buffer Sizes (Samples), and the corresponding number of seconds it will take to fill the buffer at a given speed. The default buffer size is 4096 samples. Please double check that this is enough time to collect all of your data.

		Time to Fill Buffer (secs)							
		Eye Tracker Speed (Hertz)							
		30Hz	50Hz	60Hz	120Hz	300Hz	500Hz	1000Hz	
“Buffer Size (Samples)”	1024	34.13	20.48	17.07	8.53	3.41	2.05	1.02	
	2048	68.27	40.96	34.13	17.07	6.83	4.10	2.05	
	4096	136.53	81.92	68.27	34.13	13.65	8.19	4.10	default
	8129	270.97	162.58	135.48	67.74	27.10	16.26	8.13	
	16384	546.13	327.68	273.07	136.53	54.61	32.77	16.38	
	32768	1092.27	655.36	546.13	273.07	109.23	65.54	32.77	
	65536	2184.53	1310.72	1092.27	546.13	218.45	131.07	65.54	
	131072	4369.07	2621.44	2184.53	1092.27	436.91	262.14	131.07	
	262144	8738.13	5242.88	4369.07	2184.53	873.81	524.29	262.14	
	524288	17476.27	10485.76	8738.13	4369.07	1747.63	1048.58	524.29	
	1048576	34952.53	20971.52	17476.27	8738.13	3495.25	2097.15	1048.58	

Synchronization

There are three available timing synchronization modes that can be selected for your experiment. The table below highlights the features and appropriate use cases for each mode. The default mode for new experiments is Local. This setting will compensate for eye tracker latency. Every eye tracker has some latency, Tobii has documented an the average latency for each of their eye trackers. For example, the T-Series has a latency of 30-35 ms. The Local setting will compensate for the eye tracker latency by adjust the E-Prime time stamp in the .gazedata. This setting will also minimize any drift between the independent E-Prime and Tobii clock over time. This data is saved in the RTTime column of the .gazedata file.

Time Synchronization			
Synchronization Mode	***Description***	***When to use***	
None	No synchronization. The gaze data will be time stamped using the eye tracker clock, and the E-Prime data will be time stamped using the E-Prime clock.	Timestamp compatibility is not an issues, all of the applications, including the TETServer are running on the same host, and longitudinal data collection.	
Server	Synchronization on. The TETServer time will be used to timestamp the gaze data.	Account for eye tracker latency.	
Local	**Synchronization on. The TETServer time will be converted to the E-Prime time to time stamp the gaze data.**	**Account for eye tracker latency, account for drift, and long experiments.**	**default**

More information about the Time Synchronization can be found in the Developer documentation included with the Tobii SDK in Appendix B: Time Synchronization.

Please contact Tobii Support to obtain a copy of the documentation.

Appendix D: **Contact Information**

For additional information or support

PSYCHOLOGY SOFTWARE TOOLS
Solutions for Research, Assessment, and Education

Contact us at
Psychology Software Tools, Inc
311 23rd Street Extension, Suite 200
Sharpsburg, PA 15215-2821
Phone: 412-449-0078
Fax: 412-449-0079
www.pstnet.com

For Product Service and Support:
Please visit us at https://support.pstnet.com