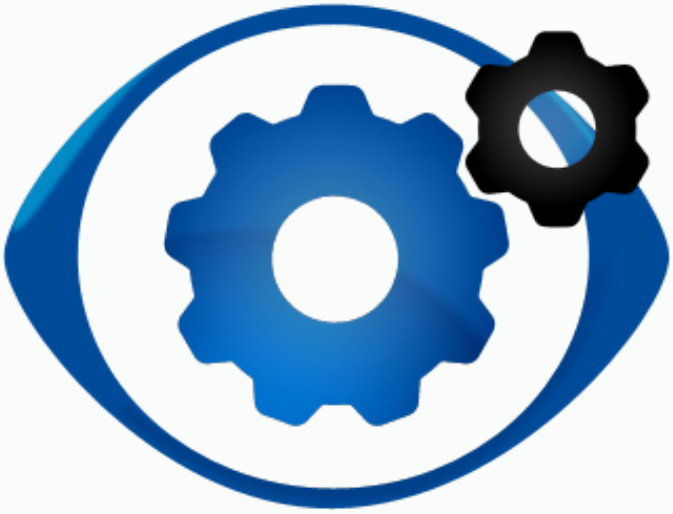


E-Prime Extensions for SMI

User Manual

For Research 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-101676

E-Prime Extensions for SMI 1.0 User Manual
PST-101680
Rev 1

If you have any questions or comments regarding this manual, or require installation assistance, please contact Psychology Software Tools, Inc. Product Service and Support Department at:

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

This manual describes the installation procedure for the E-Prime Extensions for SMI 1.0. Please review the manual completely and thoroughly before beginning the system installation.

The E-Prime Extensions for SMI 1.0 Software (PST-101676) is for research and educational purposes only.

Table of Contents

Chapter 1: **Getting Started Guide**

1.1 Before You Begin

E-Prime Extensions for SMI 1.0 is compatible with E-Prime 2.0 Service Pack 1 or newer, Professional edition. E-Prime Extensions for SMI 1.0 is not compatible with any E-Prime 1.x versions or editions. The E-Prime 2.0 Professional file extension is .es2, and this manual uses the .es2 extension when referring to the E-Prime Sample and Tutorial experiments that are distributed with E-Prime Extensions for SMI.

Prior to the E-Prime Extensions for SMI 1.0 installation, you should determine which version of E-Prime is installed. For an overview of E-Prime's versions, editions, and features, see the Knowledge Base article, KB 19564 INFO: FAQ about E-Prime Versions and Editions. A compatible version of E-Prime must be installed prior to installing E-Prime Extensions for SMI 1.0.

E-Prime Extensions for SMI 1.0 also requires that the SMI iView SDK be installed on the same computer that is running E-Prime before an SMI-enabled experiment can be generated and run. The SMI SDK is available from the SMI web site.

1.2 E-Prime Extensions for SMI 1.0 Overview

E-Prime Extensions for SMI 1.0 (EESMI) is a set of software routines that allow communication between the SMI Server (iView server) and E-Prime during the run of an experiment.
⚠ **NOTE:** *The iView server is the server for the SMI software.*
It will allow you to create E-Prime experiments or update existing E-Prime experiments to function with the SMI eye tracker device. 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

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 SMI 1.0. If you are unsure of your administrative privileges, contact your System Administrator.

Prerequisite: Install E-Prime 2.x if necessary

If you already have the Professional Edition of E-Prime 2 (SP1 or later) installed, then skip this section.

1) If necessary, ***uninstall*** any **previous versions** of **E-Prime**.
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, Company, 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).*
   ⚠ **NOTE:** *In order to use PST online Product Service and Support you will need to install your software with a valid Serial Number.*

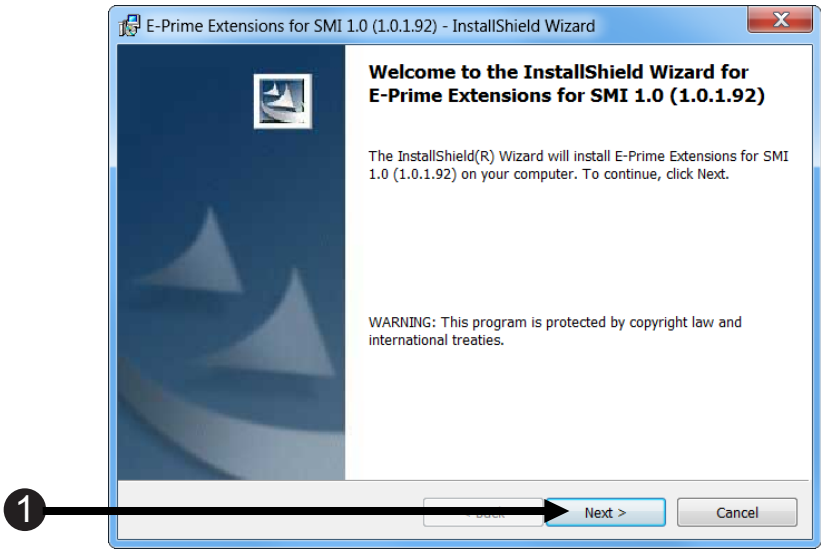
Install E-Prime Extensions for SMI 1.0

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 SMI 1.0 (EESMI). If you are unsure of your administrative privileges, contact your System Administrator.

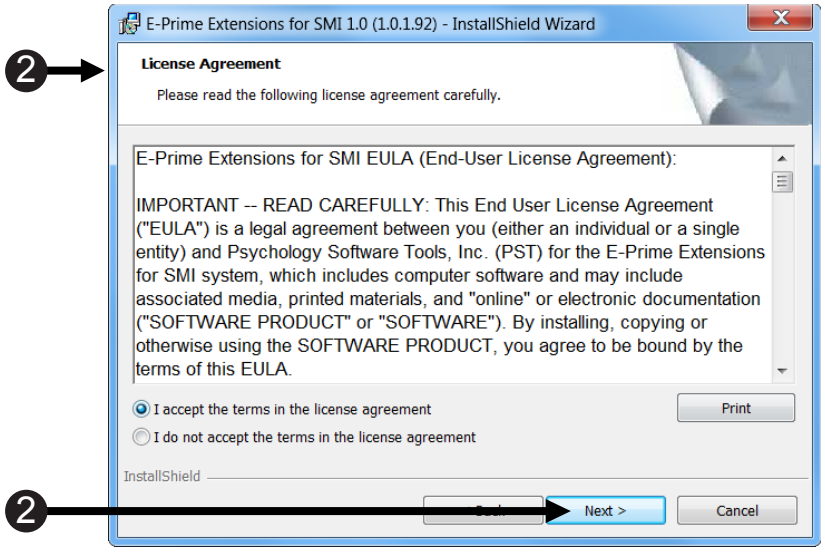
⚠ **NOTE:** *E-Prime Extensions for SMI 1.0 is compatible with E-Prime 2.0 Professional Edition only/ Service Pack 1 or later.*

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

1) ***Insert*** the **SMI 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.



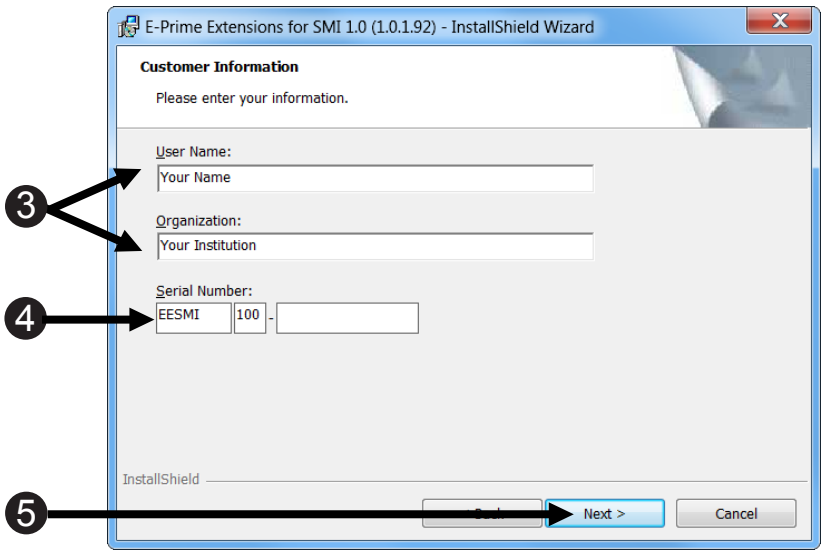
3) ***Specify*** **Your Name** and **Your Institution** or check with your system administrator for appropriate information.

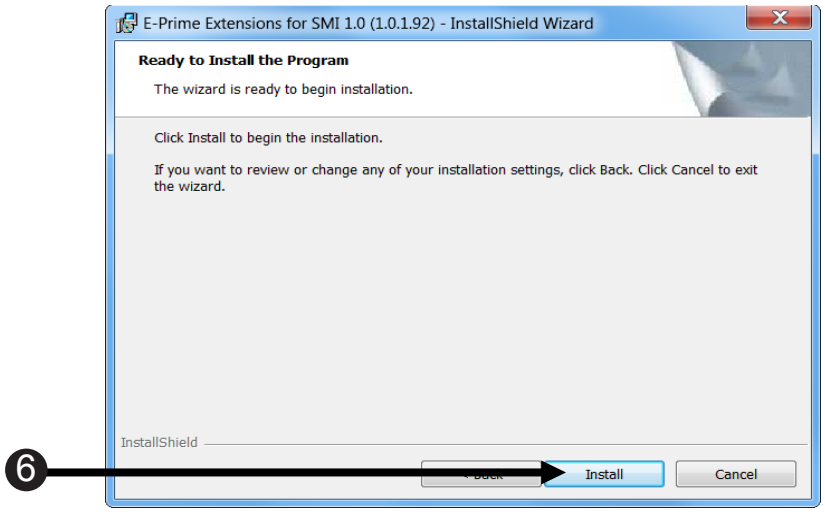
4) The **Serial Number field** must be ***complete*** to obtain access to online **Product Service and Support**.

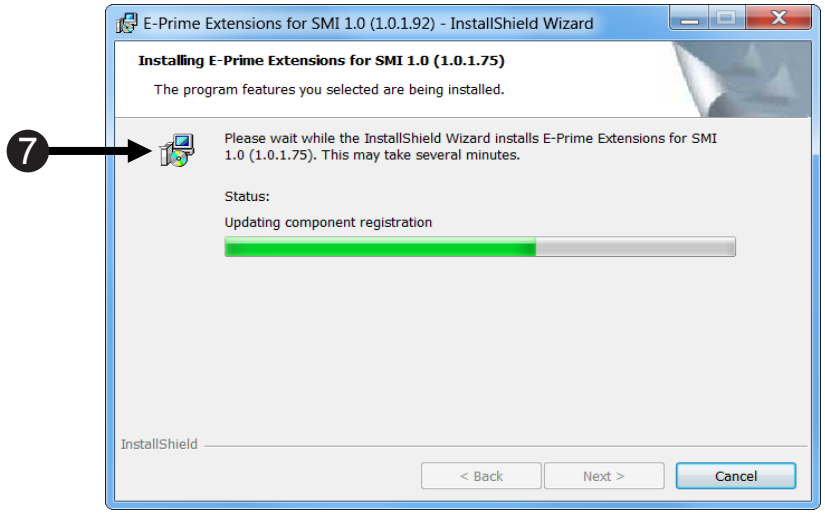
5) ***Click*** **Next** to begin transferring files to your computer.

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

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

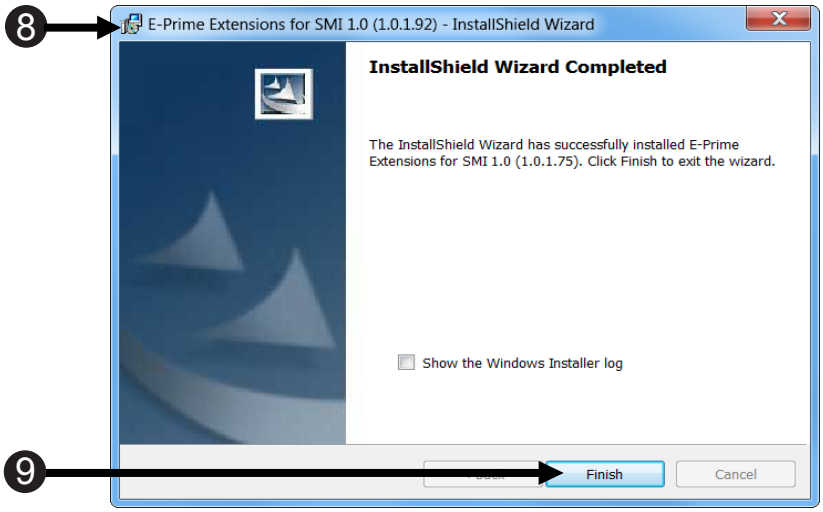






8) If **E-Prime Extensions for SMI 1.0** is installed properly, you will see the following window.

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



1.3 Software Installation (continued)

Find the "My Experiments" Folder

E-Prime 2.0 Extensions for SMI is compatible with Microsoft Windows 7, 8, and 10. (Check with your SMI documentation to determine operating system compatibility for your particular SMI eye tracker). 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 2.0 User's Guide*, and the "Tutorials" folder, which stores the E-Studio files that are documented in the *E-Prime Getting Started Guide*. E-Prime Extensions for SMI also installs Samples and Tutorials folders within the "My Experiments\SMI" folder.

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")
Windows 7	<drive>\Users\<user name>\My Documents
Windows 8 and 10	<drive>\Users\<user name>\Documents\

When the E-Prime and EESMI 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.

Run a Test Experiment

1) Test the iView server and EESMI installations. The following test of the E-Prime Extensions for SMI 1.0 (EESMI) installation assumes the proper functioning of both the iView server and SMI eye tracking device. If you are new to SMI, confirm proper functioning by verifying the system works by going through the Hardware and Software System Set-up in the iView server Manual.
2) Launch iView server to calibrate and validate the eye tracker. Leave iView server running. ***The iView server must be running in order to run an EESMI experiment.***
3) Launch E-Studio.
4) When E-Studio launches, it will detect that new EESMI sample and tutorial files are available. Answer 'yes' when prompted to copy these new files into your "My Experiments" folder.
5) After E-Prime has copied the samples, navigate to …My Experiments\SMI\Samples\ SMIVaryingPositionAOITracking.
6) Locate and open the SMIVaryingPositionAOITracking.es2 file.
7) In the E-Studio Structure window, double-click the Experiment Object at the top of the tree.
8) Click on the Devices tab of the Experiment Object Property Pages.
9) Double-click on the SMI eye tracker device and confirm that the IP settings are correct for computer(s) running the iView server and E-Prime.
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 SMIVaryingPositionAOITracking sample.
13) SMIVaryingPositionAOITracking:
    - 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 E-Prime Product Service and Support

This User Manual as well as proper use of EESMI requires a working knowledge of E-Prime. 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 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.

1.5 SMI and EESMI Resources

1) SMI manuals are included in the SMI product shipment.
2) Contact SMI for support requests for any issues related to the installation and operation of your eye tracker.
3) EESMI 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, go to support.pstnet.com. Registration requires a valid EESMI serial number and an e-mail address.

1.6 EESMI Samples and Tutorials

Samples

Your EESMI system comes with a set of pre-programmed Sample experiments. These experiments are intended as examples of how to utilize a particular EESMI feature, such as gaze contingency, and each Sample .es2 file can be run on a computer with a compatible and configured SMI eye tracker. Sample experiments minimally create an .edat2 data file that can be analyzed with E-DataAid. However, no Sample experiment is intended to be used for data collection "out of the box": it is always the experimenter's responsibility to ensure that the experiment design, stimulus sampling method, and data logging is fully understood and further that any experiment is piloted and the data analyzed prior to collecting experimental data. For general guidelines, see **Appendix B: Considerations in Computerized Research** in the *E-Prime 2.0 User Manual*. For a brief summary of each experiment, see the abstract on Experiment Object's General tab.

The following table summarizes the main EESMI features utilized in each Sample experiment:

Utilizes the following EESMI functionality:							
Sample Name (.es2)	Calibration and Validation	Wait for Fixation	Gaze Data File Creation	Number and Type of AOIs*	Gaze File Replay	.idf File with User Event Markers**	Other
SMIFixedPositionAOI	X	X	X	2, fixed	X		
SMIFixedPositionAOIMarkers	X	X	X	2, fixed	X	X	
SMIFixedPositionAOIMultiMonitor	X	X	X	2, fixed	X		Uses a second monitor for experimenter display
SMIVaryingPosition AOI	X	X	X	5, random	X		
SMIVaryingPosition AOITracking	X	X	X	5, random	X		Dynamically draws a border around the stimulus being fixated upon
SMISlideShow.es2	n/a – corresponds to the SlideshowProfessional.es2 example provided with the SMI SDK						

** Not to be confused with an SMI AOI; these are E-Prime AOIs that are created and tracked with E-Basic script within the E-Prime experiment.*

*** All of the EESMI Sample and Tutorial experiments create .idf files due to the inclusion of the SMIStartRecording and SMIStopRecording PackageCalls;see Tutorial 4 for details on adding User Event Markers to the .idf file.*

Tutorials

In addition to the Samples experiments described above, your EESMI system comes with a set of Tutorials. The tutorials walk you through the steps that are necessary to implement one or more EESMI features. While we recommend that you go through each tutorial in order, the tutorials are designed to be implemented independently. For example, you do not need to complete Tutorial 2 prior to starting Tutorial 3. For each tutorial, we provide a starting point E-Prime experiment (.es2 file) to be loaded into E-Studio. The first task of each tutorial includes a step to save a personal copy of the currently loaded E-Studio experiment. For example, Tutorial 1 begins by instructing you to load the file FixedPositionAOI.es2 and save it as “MySMIFixedPositionAOITutorial1”. In addition, we provide a copy of the expected state of the E-Studio experiment at the completion of the tutorial. For Tutorial 1, this file is named “SMIFixedPositionAOITutorial1Completed”. This file is provided for two reasons. First, in the event that you make a mistake while working through the tutorial, you can examine what the Sample experiment is expected to look like once it has been completed. Second, many of the tutorials build upon the prior one and begin by loading a completed version of the previous tutorial. By having the completed versions of each tutorial available, you can work on any tutorial of interest, rather than having to complete all of the tutorials in order.

Lastly, some of the completed tutorials match the Sample experiments. The following table illustrates the relationship between the Tutorial experiments and the Sample experiments:

Tutorials to Samples Mapping			
Tutorial	**Starting Point**	**Ending Point**	**Completed Tutorial maps to Sample experiment**
1: Adding SMI Support to an E-Prime Experiment	FixedPositionAOI	SMIFixedPositionAOITutorial1Completed	None
2: Writing to the Tab Delimited .gazedata file	SMIFixedPositionAOITutorial1Completed	SMIFixedPositionAOITutorial2Completed	SMIFixedPositionAOI
3: Introduction to the .gazedata file	SMIFixedPositionAOITutorial2Completed *Tutorial 3 reviews in detail the output .gazedata file that is created from Tutorial #2. The tutorial does not make any changes to any of the tutorial .es2 files.*		SMIFixedPositionAOI
4: Adding Markers for BeGaze	SMIFixedPositionAOITutorial2Completed	SMIFixedPositionAOIMarkersTutorial4Completed	SMIFixedPositionAOIMarkers
5: Working with the Eye Gaze Data Interactively	SMIVaryingPositionAOITrackingTutorial5Start	SMIVaryingPositionAOITrackingTutorial5Completed	SMIVaryingPositionAOITracking

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

2.1 Overview

This chapter illustrates the creation of an E-Prime/SMI experiment using one of the sample paradigms provided with the E-Prime Extensions for SMI 1.0 installation (EESMI), the SMIFixedPositionAOI.es2 paradigm. It also introduces and explains some simple E-Prime programming methods.

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

While this document includes some basic introduction to programming concepts specific to using E-Prime with SMI, 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 SMI 1.0 (EESMI) combines two technologies to create an integrated programming environment — PackageCalls and InLine Script. The typical EESMI experiment uses both technologies. A brief overview of each is provided below.

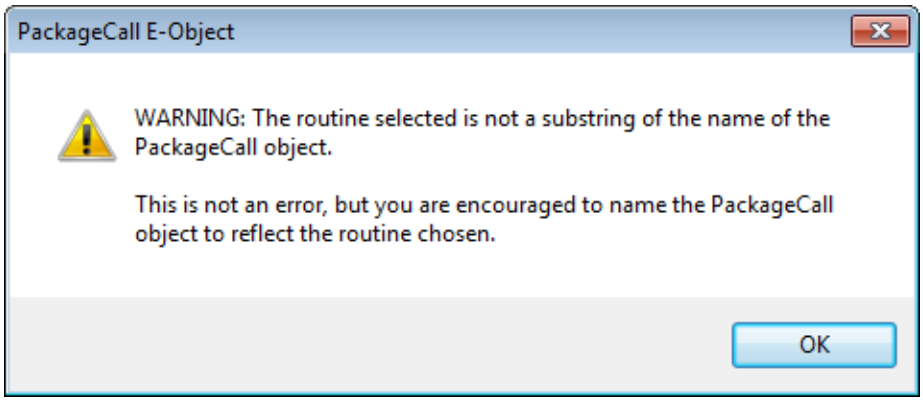
Use of PackageCalls in the EESMI

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 EESMI installation. The EESMI PackageCalls allow you to perform various functions automatically, such as hardware initialization and data output formatting, via the graphical representations shown in the E-Studio Structure window. Some of the EESMI PackageCalls require you to set or modify parameters to enable them to work with your experiment. The **Chapter 3: SMI PackageCall Reference, Page 105** 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 SMI package file, to create a name for each object by concatenating “SMI” with the name of the routine being called. For example, a PackageCall that is configured to call the “Open” routine would be named “SMIOpen”.

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 script. The InLine method can be used to call PackageCalls 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 direct PackageCalls via E-Basic script, please refer to **Chapter 3: SMI PackageCall Reference, Page 105** and **Tutorial 2: Writing to the Tab Delimited .gazedata File, Page 50**. These sections of the manual provide PackageCall parameters and functions as examples of the script commonly required to successfully invoke each routine.

⚠ **NOTE:** *EESMI must be fully and correctly installed prior to using the SMI PackageCalls by either method.*

2.3 Critical Setup Requirements

⚠ **NOTE:** *You must ensure that your SMI eye tracker is operating properly prior to running any SMI-enabled experiment. This means that you need to run the iView server and run both calibration and validation on your participant prior to running your E-Prime experiment.*

Further, iView server must continue running in order for E-Prime to communicate successfully with the SMI eye tracker. Do NOT exit out of iView server after calibration and validating your participant. You may minimize the program, but you cannot close it.

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

Summary:

Incorporating SMI support into an existing E-Prime experiment primarily involves adding both the SMI package file and the SMI 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 SMI Package to meet their individual needs.

During this tutorial, you will add E-Prime Extensions for SMI 1.0 support to an existing E-Prime experiment, FixedPositionAOI.es2 that is distributed with the tutorials. The FixedPositionAOI.es2 experiment presents one block of three trials. 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. To gain the most from this tutorial, it is recommended that you load and run the FixedPositionAOI.es2 experiment as a participant prior to making the modifications outlined the tutorial. This will allow you to develop a clear understanding of the experiment task prior to making the SMI eye tracker-specific modifications.

Goal:

This tutorial illustrates how to add the SMI PackageCalls into the FixedPositionAOI.es2 experiment included with the EESMI system. When you have completed this tutorial, you will create a basic “SMI-enabled” paradigm.

Overview of Tasks:

- Load FixedPositionAOI and resave it as MySMIFixedPositionAOITutorial1.es2.
- Add the SMI package file to the Experiment Object.
- Add the SMI eye tracker device to Devices tab and edit the properties to enable communication between E-Prime and the iView server.
- Add the SMI PackageCall to open/close the SMI eye tracking device.
- Add the SMI PackageCall to display the track status Window.
- Add the SMI PackageCall to open/close the .gazedata file.
- Add the SMI PackageCall to start/stop tracking.
- Add the SMI 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 experiment from the SMI\Tutorials folder.

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 the appropriate folder, and load the FixedPositionAOI.es2 experiment "…My Experiments\SMI\Tutorials\SMIFixedPositionAOI\FixedPositionAOI.es2."

1) ***Click*** on the Windows Start menu, ***select*** **All Programs**, and then ***select*** **E-Prime 2.0**. From the menu, ***click*** on **E-Studio** to launch the application.

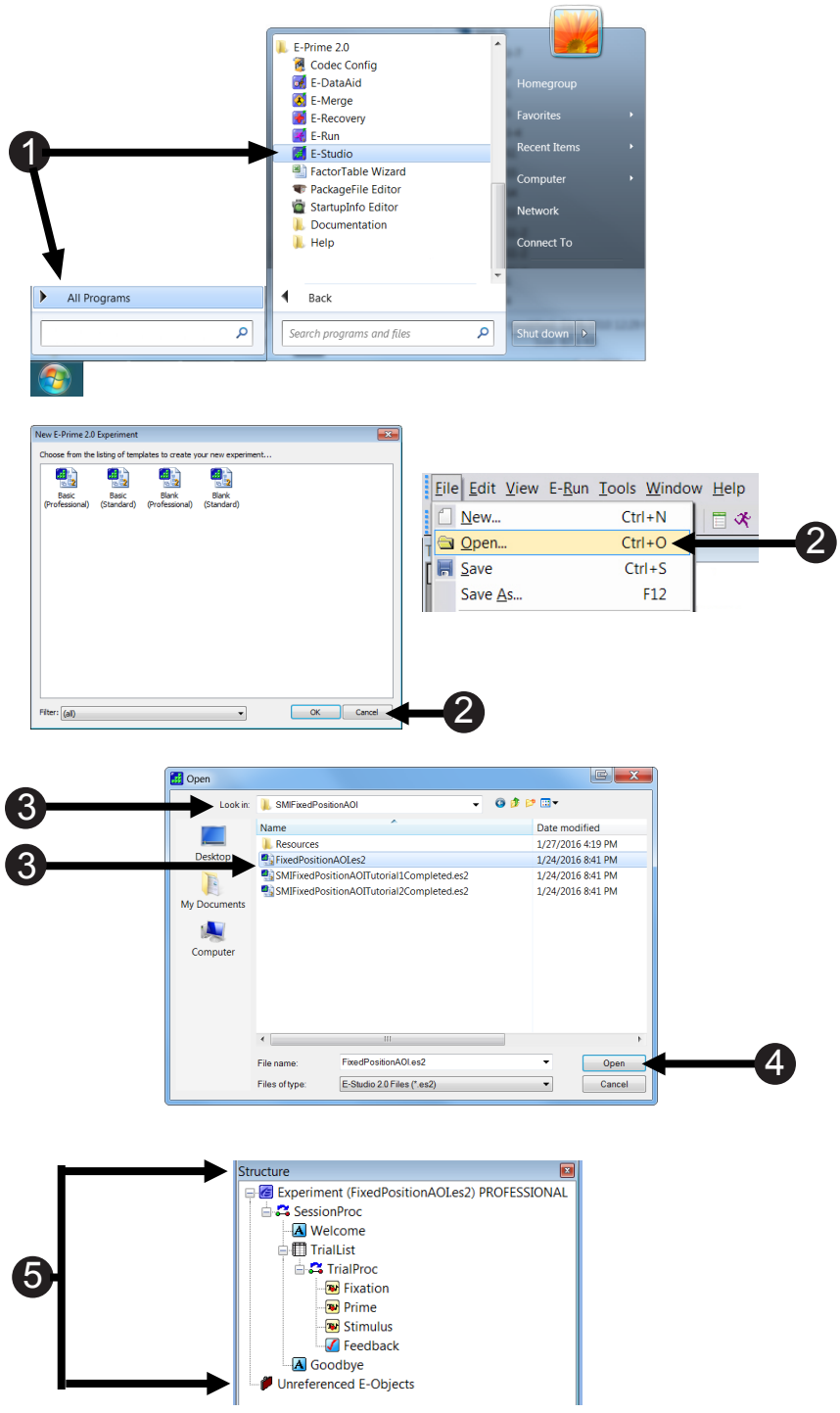
2) ***Click*** the **Cancel** button. ***Select*** **File** > **Open**.

3) ***Navigate*** to the "**…My Experiments\SMI\Tutorials\SMIFixedPositionAOI**" folder to load the paradigm.

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

   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

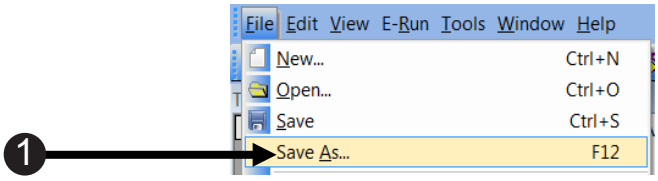
Save the FixedPositionAOI.es2 experiment in the same folder under the new name "MySMIFixedPositionAOITutorial1.es2".

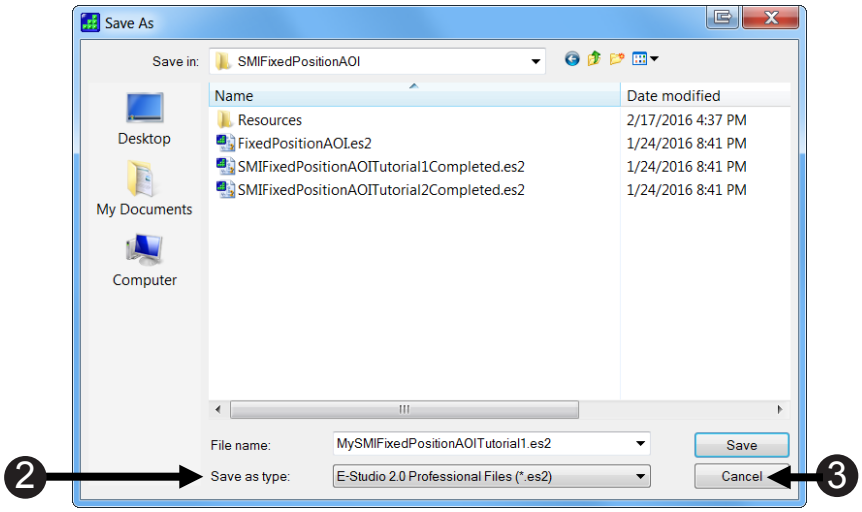
Rename the experiment, but be sure to save it in the same folder ("…My Experiments\SMI\Tutorials\SMI\SMIFixedPositionAOI") so that any references to resources within the experiment remain valid and can be reused.

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

2) **Type *MySMIFixedPositionAOITutorial1.es2*** as the new name in the **File name** field.

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





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

Open the Property Pages for the Experiment Object and use the Packages tab to add the SMI package file 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 SMI software at runtime are contained within the SMI package file.

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

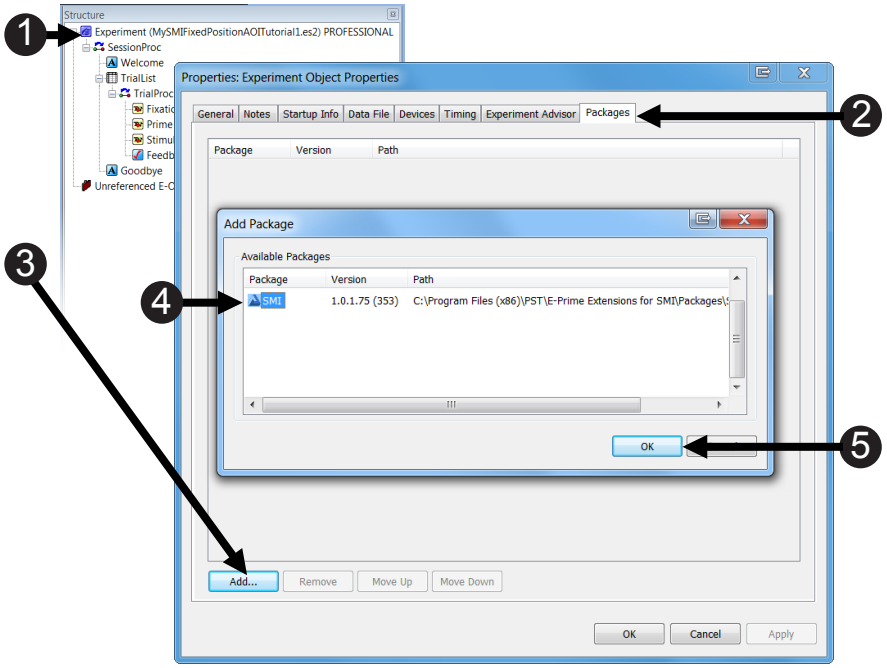
2) ***Click*** on the **Packages** tab of the **Experiment Object Property Pages**.

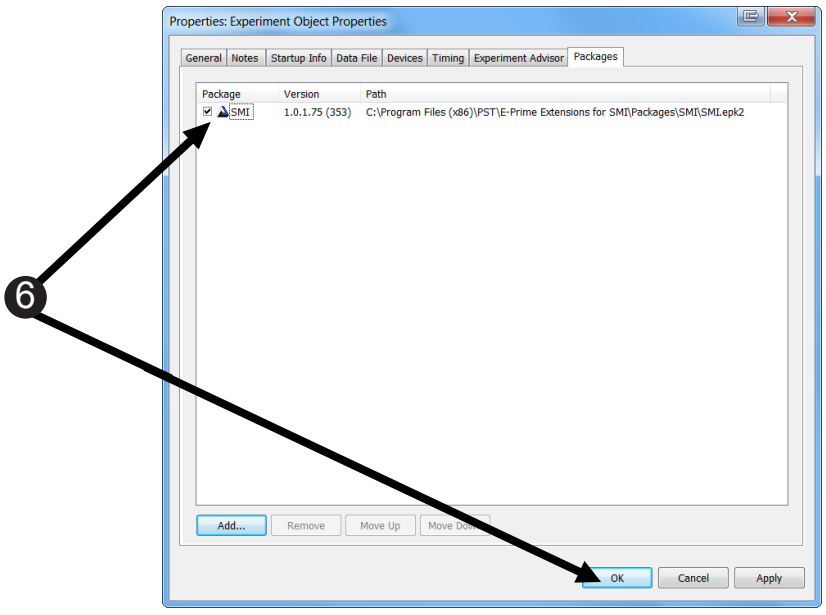
3) ***Click*** the **Add…** button.

4) ***Select*** the **SMI** 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 **SMI package file** 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 SMI package file that is currently installed on your machine and may not match the picture.





Task 4: Add the SMI Eye Tracker Device to the Experiment Properties

Open the Property Pages for the Experiment Object and select the Devices tab to add the SMI eye tracker device to the experiment.

Select the Devices tab of the Experiment Object Property Pages to add the SMI eye tracker 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.**

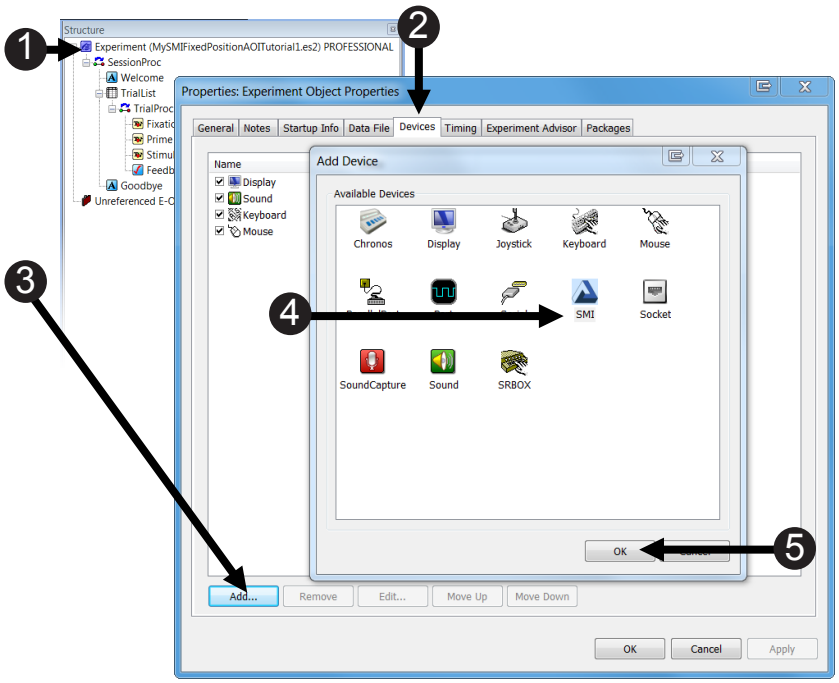
3) ***Click*** the **Add…** button.

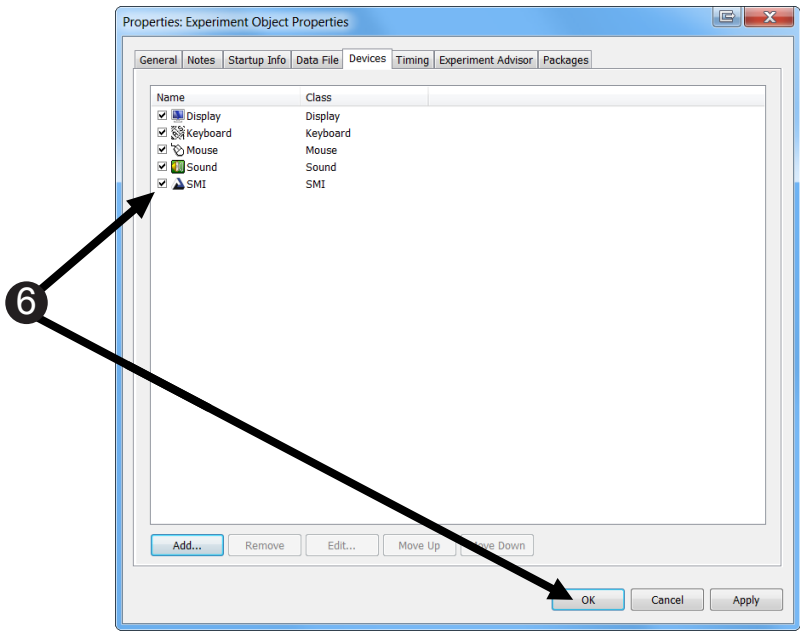
4) ***Select*** the **SMI eye tracking device** in the **Add Device dialog**.

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

6) ***Verify*** the **SMI eye tracker 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 SMI device to the top as the Display device must open prior to the SMI eye tracker device. Otherwise you will encounter a fatal error, and E-Studio will close when you try to run an experiment.





Task 5: Edit the SMI Eye Tracker Device Property Pages

Open the SMI eye tracker device Property Pages, specify the SMI Server IP Address, and confirm the Port number.

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

⚠**NOTE:** *The iView server is the server for the SMI software.*

1) ***Click*** the **SMI eye tracker device** to highlight it.

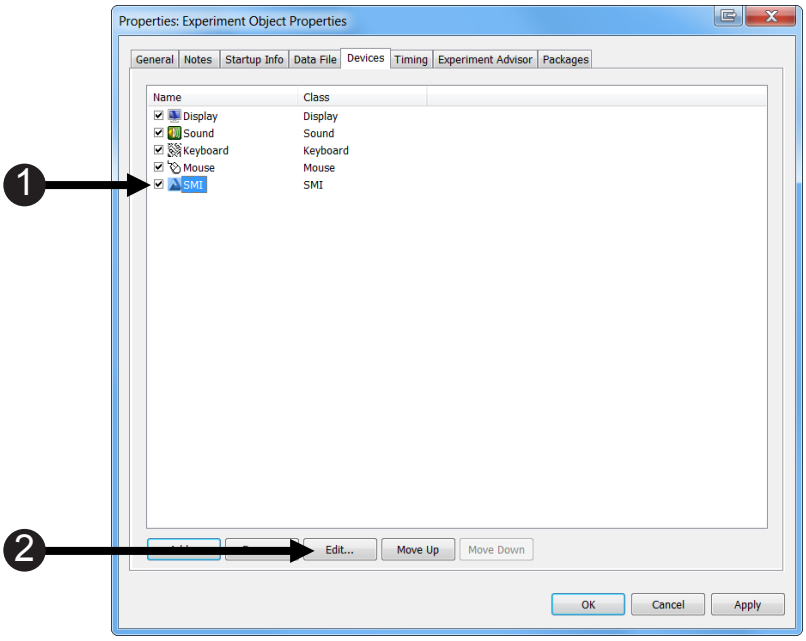
2) ***Click*** **Edit.**

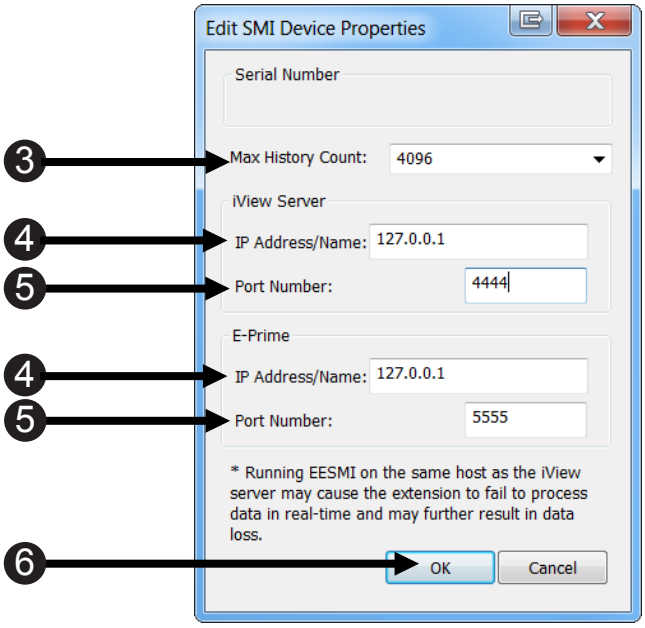
3) **Max History Count** controls the amount of time that the eye tracker can save eye gaze data before the oldest data in the buffer is overwritten. The default is 4096 samples. When using a 60Hz eye tracker device the buffer will hold ~68 seconds of eye gaze data. This is per trial and not for the length of the experiment. You will need to verify that the size of this buffer is sufficiently large to meet the needs of your experiment.

4) ***Verify*** the **IP Addresses**.
   An EESMI experiment can be deployed on a single (both E-Prime and iView server run on the same computer) or a dual PC (E-Prime runs on one computer, iView server runs on another computer) setting. The default IP Addresses are for a single PC setup. Modify these as needed. See **Appendix A: Locate Your IP Address and Ping Another Computer, see Page 134**.

5) ***Review*** the **Port Number.**
   This number must match the port number on the iView server and E-Prime computers; see Appendix A.

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





Task 6: Add the SMIOpen PackageCall to Open the SMI Package

Add a PackageCall at the beginning of the SessionProc and Name the PackageCall SMIOpen.

The SMIOpen PackageCall opens the SMI eye tracker device in preparation for eye movement data collection. The SMIOpen PackageCall will be added at the beginning of the SessionProc for this purpose.

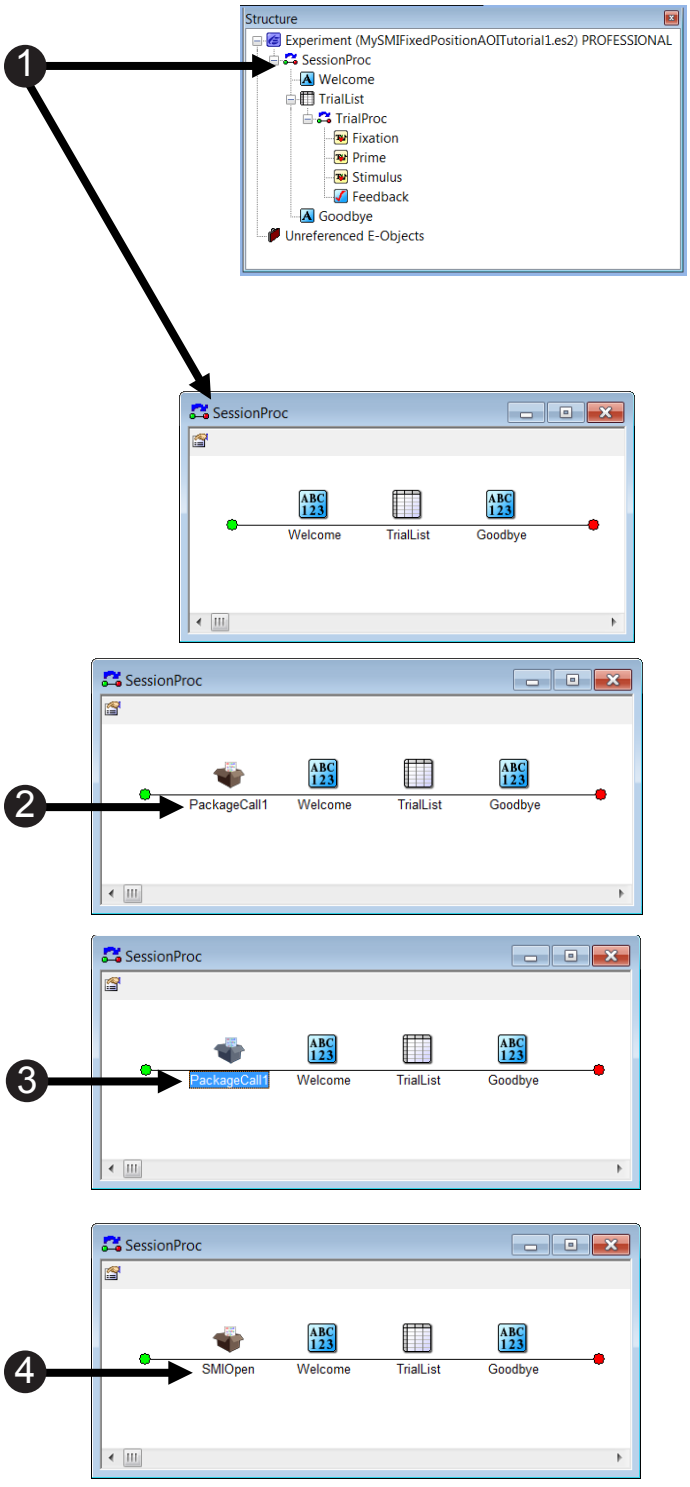
1) ***Double click*** the SessionProc Object to open it in the workspace.

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*** **"SMIOpen"** as the new object name and then ***press*** **Enter** to accept the change.



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

Configure the SMIOpen PackageCall to call the Open routine in the SMI 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 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 SMIOpen PackageCall includes parameters that allow you to turn on/off the communication between the E-Prime experiment and the SMI eye tracker device.

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

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

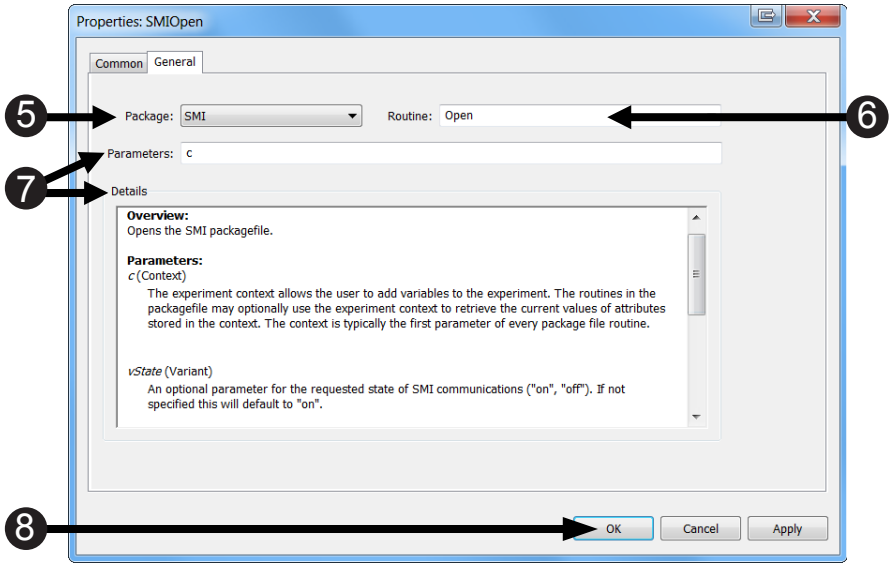
7) ***Select*** **Open** from the **Routine** dropdown list.

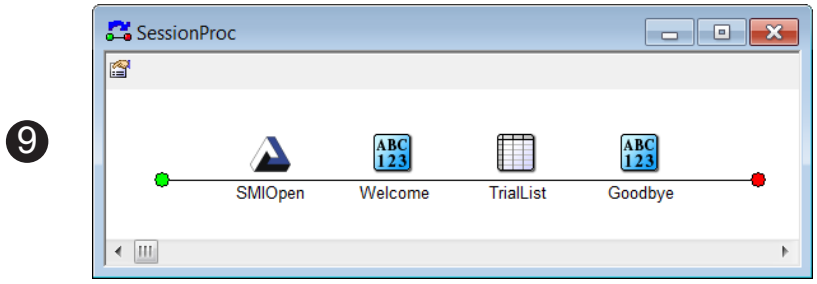
8) ***Review*** the **SMIOpen** parameters listed in the **Parameters** and **Details fiel**ds.

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

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

When the Package file defines an icon for the routine it will replace the default PackageCall icon.





Task 7: Add the SMICalibRegular PackageCall to enable the Calibration Routines

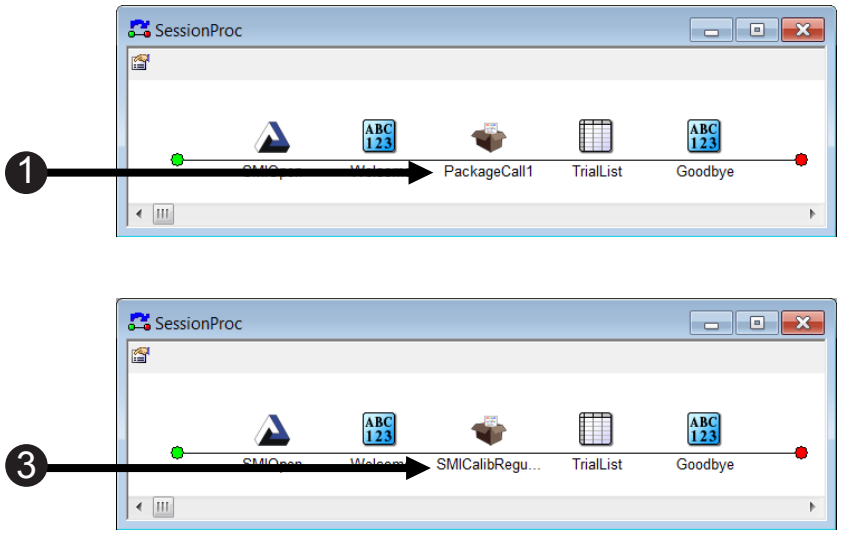
Add a PackageCall to the SessionProc and Name the PackageFile SMICalibRegular.

The SMICalibRegular PackageCall is used to display the track status window and calibrate the participant. This PackageCall will result in a track status screen appearing at the start of the experiment. A black box will appear with two white circles representing the participant's eyes. The track status window remains on the screen until a key press is made. The basic purpose of the track status window is to verify that you are getting sufficient data via the SMI Server. If there are no circles corresponding to the user's eyes, you will not get accurate and valid eye gaze data. If the track status window does not show two white circles corresponding to the participant's eyes, you may want to exit the experiment (by pressing Ctrl+Alt+Shift simultaneously), Check your system configuration, and/ or recalibrate the participant in the iView server before starting the experiment again. Once you are satisfied that the quality of the eye tracking is sufficient, simply press a key to dismiss the track status window (while running an experiment). Calibration (5 points by default) will then start, followed by validation (4 points by default). At the end of calibration, you will be shown the results of the calibration and prompted to accept or reject the calls.

1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Welcome** object 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*** **"SMICalibRegular"** as the new object name and then then ***press*** **Enter** to accept the change.



Task 7 (continued): Add the SMICalibRegular PackageCall to enable the Calibration Routines

Add a PackageCall to the SessionProc and Name the PackageFile SMICalibRegular.

After you select SMI as the Package and CalibRegular as the Routine, the Parameters field will be set to the default parameter “c” and the Details field will be filled with the text that the package file author included for the selected routine. The SMICalibRegular PackageCall includes a parameter that passes the current experiment as the context. For an introduction to the experiment context, see **Stage 3: Communicating with E-Prime 2.0 Objects** in the *E-Prime 2.0 User’s Guide*.

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

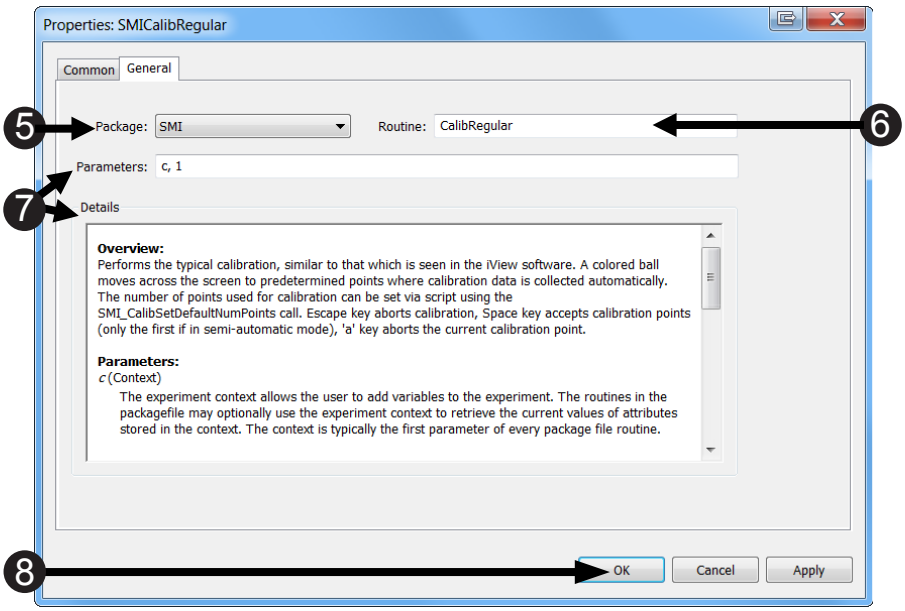
5) ***Select*** **SMI** from the **Package** dropdown list.

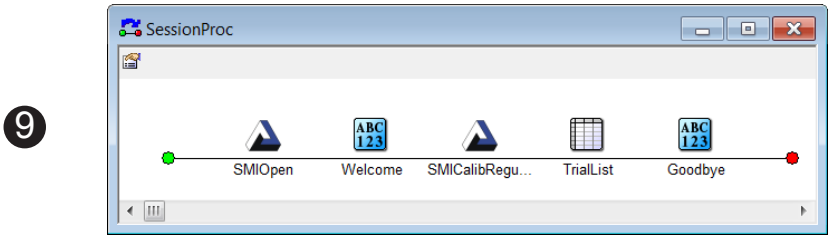
6) ***Select*** **CalibRegular** from the **Routine** dropdown list.

7) ***Review*** the **SMICalibRegular** parameters listed in the **Parameters** and **Details** 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 SMIOpenGazeDataFile PackageCall to Create and Open the Tab Delimited .gazedata file

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

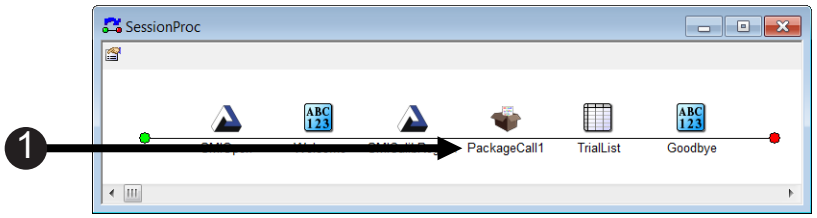
The next step is to add the SMIOpenGazeDataFile PackageCall after the SMICalibRegular PackageCall. The purpose of this call is to open the tab delimited .gazedata file that will be created (in addition to the .edat2 file) once an experiment has completed. (Operations associated with data collection and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

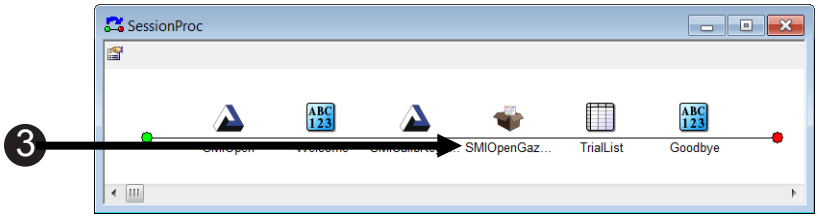
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **SMICalibRegular** PackageCall in the **SessionProc** procedure. The object will be given the default name of **PackageCall1**.

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*** **SMIOpenGazeDataFile** as the new object name and then ***press*** **Enter** to accept the change.

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





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

Call the OpenGazeDataFile routine of the SMI 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 then a filename will be created in the form of DataFile.BaseName, which defaults to [ExperimentName]-[Subject#]-[Session#].gazedata. 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**".

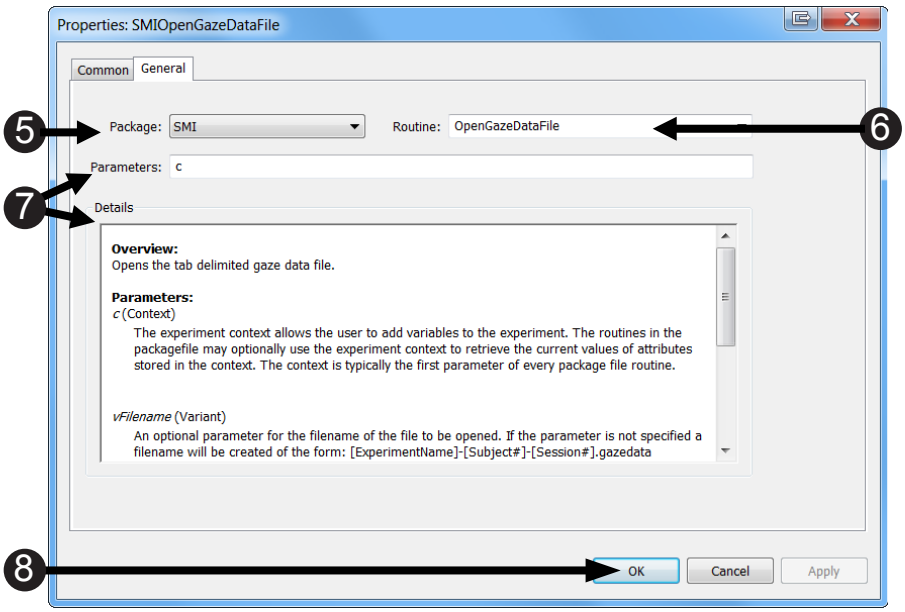
5) ***Select*** **SMI** from the **Package** dropdown list.

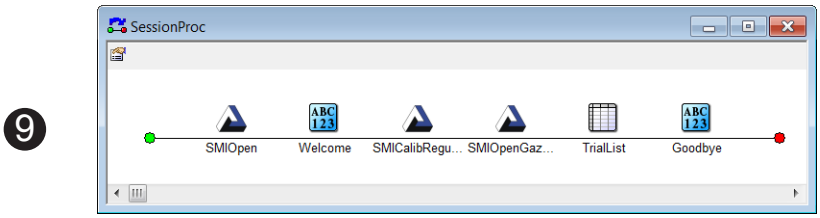
6) ***Select*** **OpenGazeDataFile** from the **Routine** dropdown list.

7) ***Review*** the **SMIOpenGazeDataFile** parameters listed in the **Parameters** and **Details** fields.

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

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





Task 9: Add the SMIStartRecording PackageCall

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

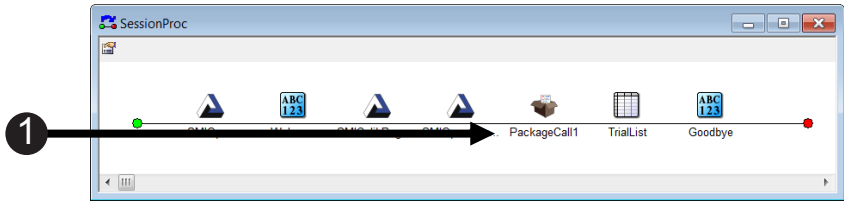
The last PackageCall that needs to be added to the SessionProc before we move our attention to the trial events is the SMIStartRecording PackageCall. This PackageCall clears any data that may be in the recording buffer and begins sending data to the .idf file. We place the SMIStartRecording PackageCall at this point in the SessionProc because calibration data is typically not analyzed in BeGaze and therefore does not need to be included in the .idf file. Note that in a later task we will call SMIStopRecording to stop the flow of eye tracking data to the .idf file prior to ending the experiment.

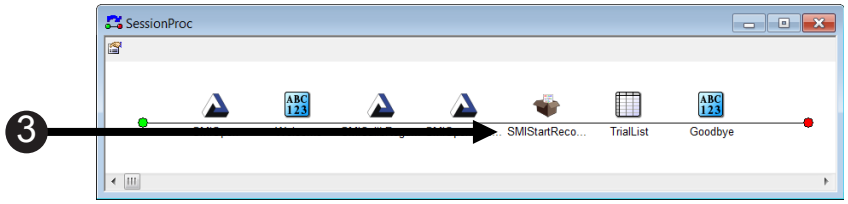
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **SMIOpenGazeDataFile** PackageCall in the **SessionProc** procedure. The object will be given the default name of **PackageCall1**.

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*** **SMIStartRecording** as the new object name and then ***press*** **Enter** to accept the change.

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





Task 9 (continued): Add the SMIStartRecording PackageCall

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

The last PackageCall that needs to be added to the SessionProc before we move our attention to the trial events is the SMIStartRecording PackageCall. This PackageCall clears any data that may be in the recording buffer and begins sending data to the .idf file. We place the SMIStartRecording PackageCall at this point in the SessionProc because calibration data is typically not analyzed in BeGaze and therefore does not need to be included in the .idf file. Note that in a later task we will call SMIStopRecording to stop the flow of eye tracking data to the .idf file prior to ending the experiment.

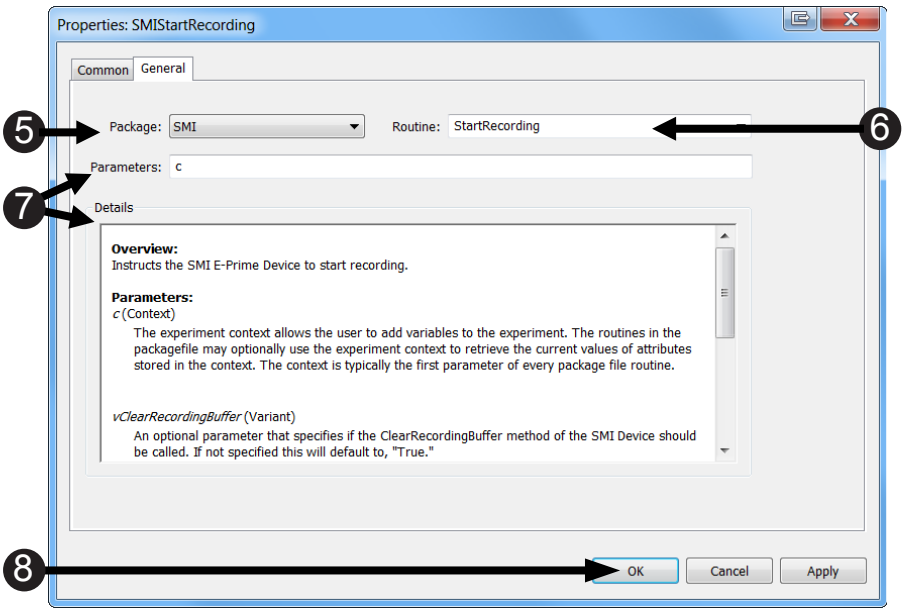
5) ***Select* SMI** from the **Package** dropdown list.

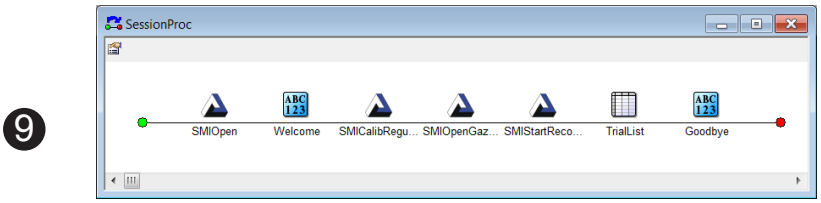
6) ***Select* StartRecording** from the **Routine** dropdown list.

7) ***Review*** the **SMIStartRecording** parameters listed in the **Parameters** and **Details** fields.

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

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





Task 10: Add the SMIStartTracking PackageCall to Begin Tracking Eye Movements

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

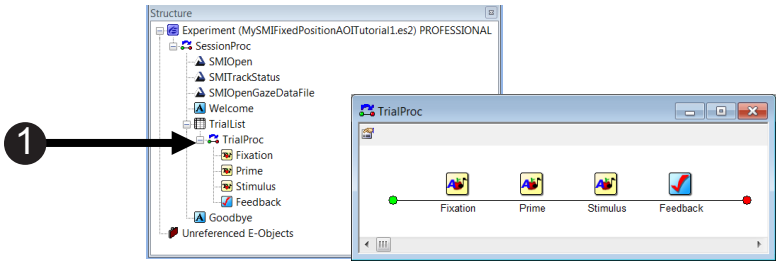
The next step is to add the SMIStartTracking PackageCall. This PackageCall should be placed at the location in the procedure where you decide you want to start tracking eye movements. This is often at the beginning of a trial, but may vary depending on the experiment. Typically when the eye tracker is active the mouse cursor will not be visible to the participant. If you would like the mouse cursor to move with eye movements, please see **3.2 Calls to the SMI Device, see Page 106** for details.

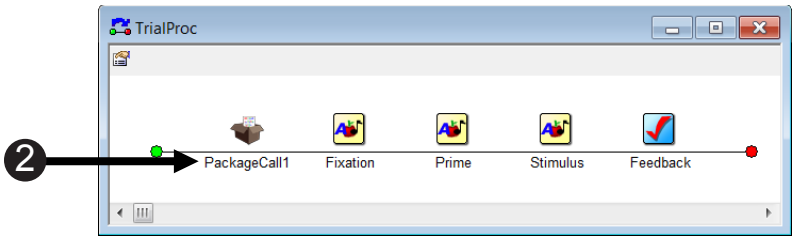
1) ***Double click*** the **TrialProc** Object to open it in the workspace.

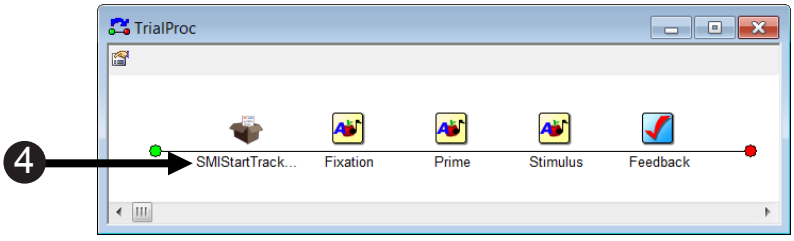
2) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it *before* the Fixation Object in the **TrialProc** procedure. The object will be given a default name of **PackageCall1**.

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*** **"SMIStartTracking"** as the new object name and then ***press*** **Enter** to accept the change.







Task 10 (continued): Add the SMIStartTracking PackageCall to Begin Tracking Eye Movements

Configure the SMIStartTracking PackageCall to call the StartTracking routine of the SMI package and accept the default properties.

⚠ **NOTE:** *The default values of this PackageCall clear the SMI history buffer when the eye tracking is started. If you do not wish to clear the SMI history buffer, then you must set this value to "False." The default values of this PackageCall result in an ImageMessage being sent to the .idf file (assuming that SMIStartRecording has already been called).*

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

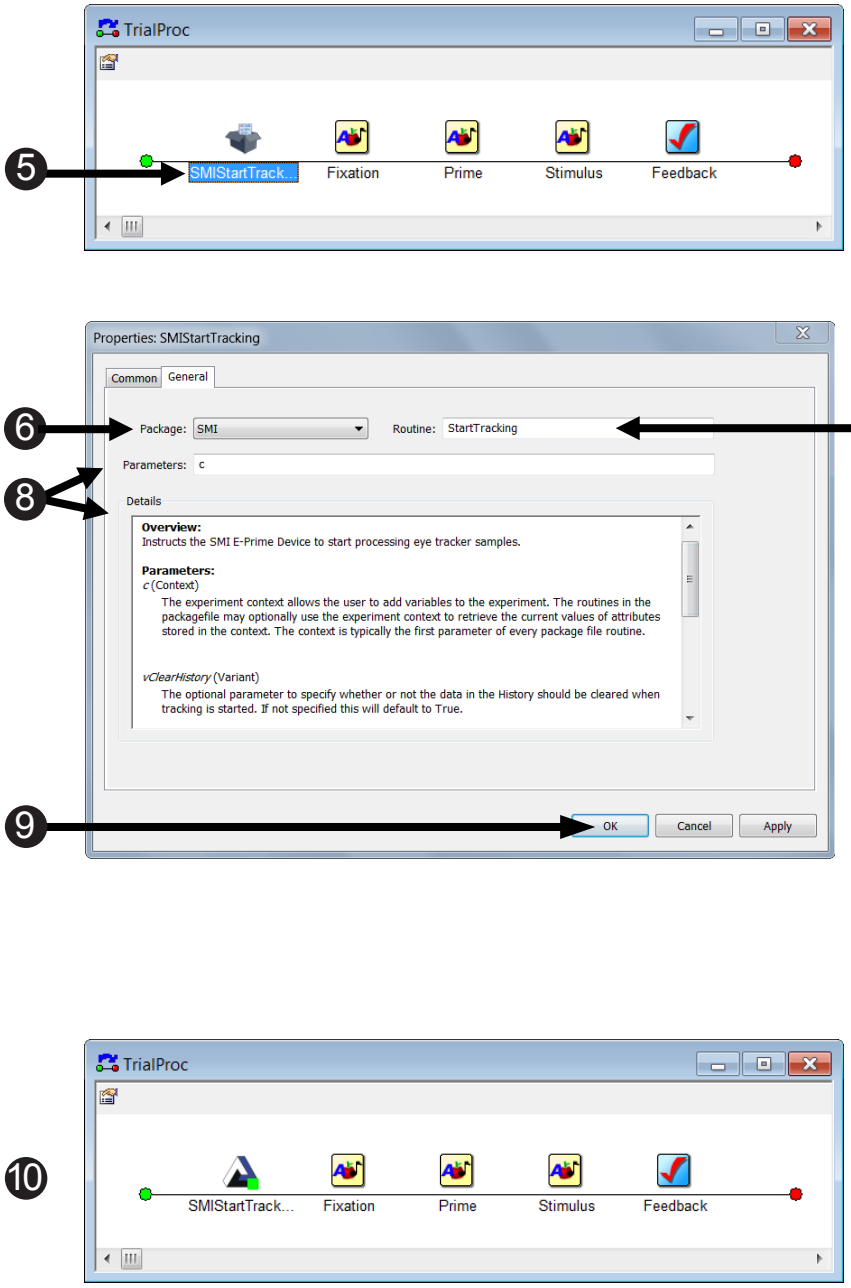
6) ***Select*** **SMI** from the **Package** dropdown list.

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

8) ***Review*** the **SMIStartTracking** parameters listed in the **Parameters** and **Details** fields, particularly **vImageMessage**.

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

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



Task 11: Add the SMIWaitForFixation 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 SMIWaitForFixation.

The next step is to add the SMIWaitForFixation 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.

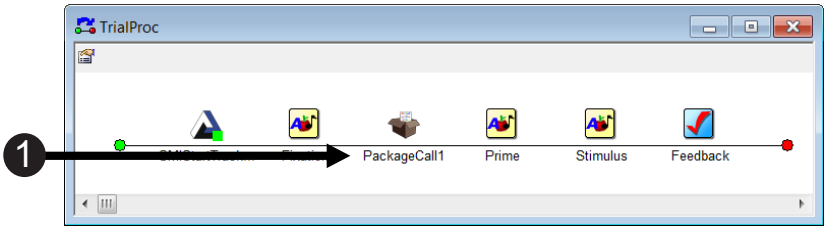
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Fixation** Object in the **TrialProc** procedure. The object will be given a default name of **PackageCall1**.

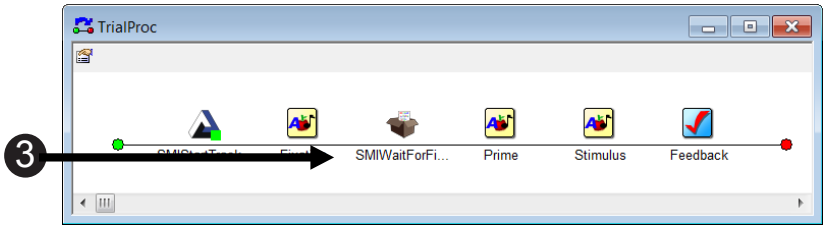
2) ***Click*** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.

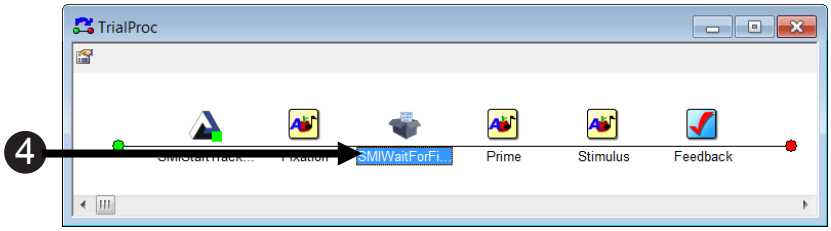
   You may alternatively right click on the object and select Rename from the context menu.

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

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







Task 11 (continued): Add the SMIWaitForFixation PackageCall to Halt the Trial until the Participant Fixates on a Particular Object for a Given Amount of Time

Configure the SMIWaitForFixation PackageCall to call the WaitForFixation routine of the SMI package and edit the Parameters.

This PackageCall is used in conjunction with a Slide Object. In this example, the Fixation Object is the name of the Slide Object that works with WaitForFixation. When using this call, it is important to account for all of the Parameters that can be set for the WaitForFixation call as well as configure the Slide Object accordingly. We set up the Parameters for the Slide Object in this task, and in Tasks 11 and 12, we will set and discuss the Fixation Slide Object in detail.

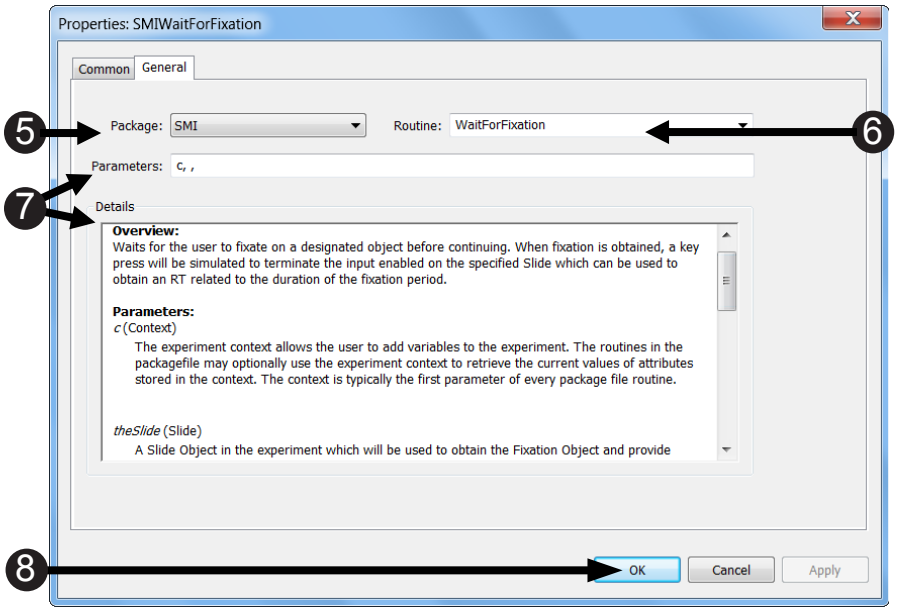
5) ***Select*** **SMI** from the **Package** dropdown list.

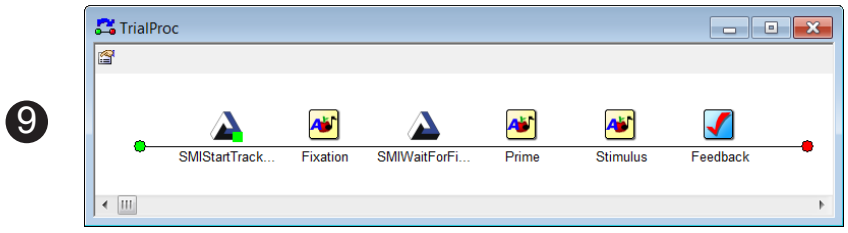
6) ***Select*** **WaitForFixation** from the **Routine** dropdown list.

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 **SMIWaitForFixation Property Pages**.

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





Task 12: 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.

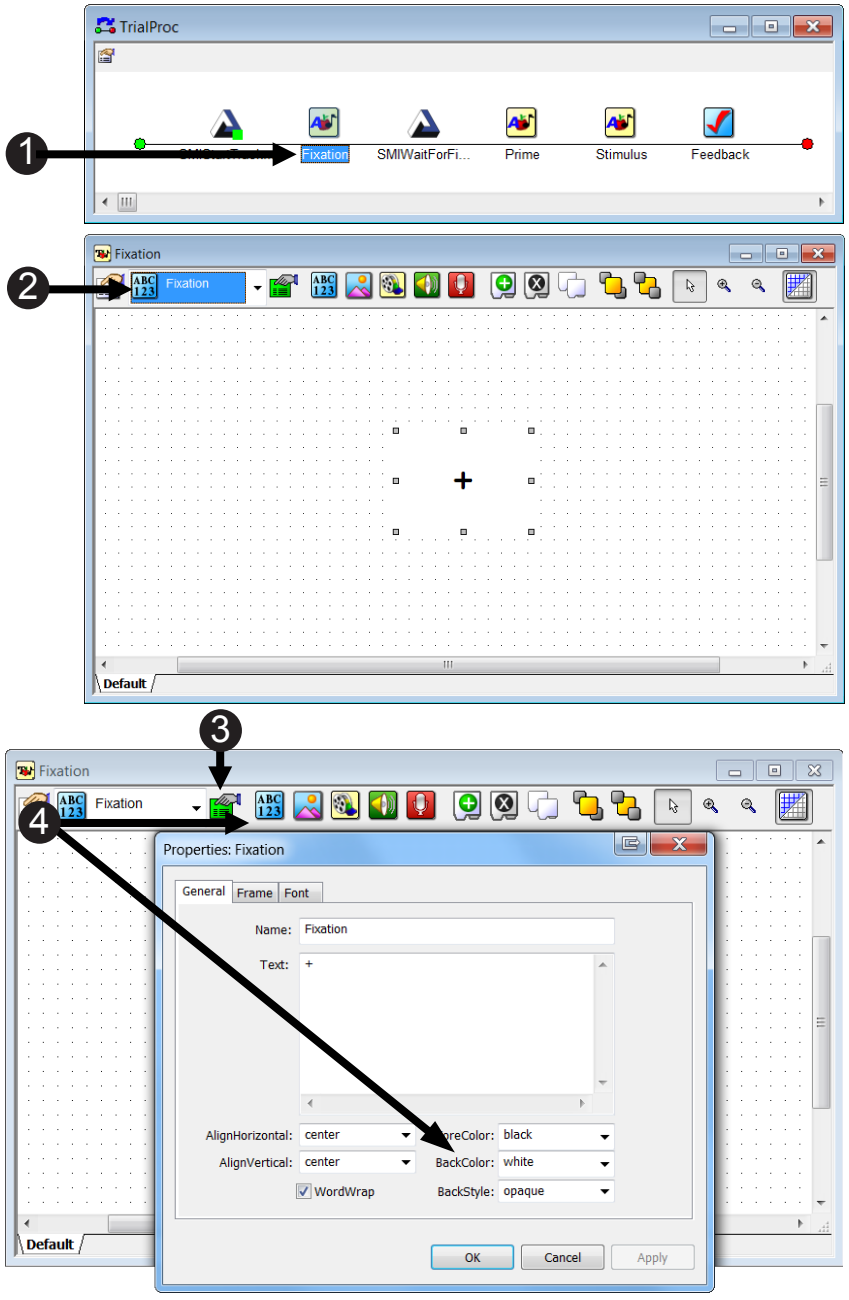
The WaitForFixation PackageCall takes a parameter that identifies the name of the Slide object with which it interacts. In this tutorial, we have configured the WaitForFixation PackageCall to communicate with the Fixation Slide Object; this occurred when we specified "Fixation" as the second parameter in Step 7 of the prior task. In this current task, we confirm the Slide Object named "Fixation" contains a sub-object named "Fixation" which displays the "+". This step can be confusing because ***both*** the *Slide Object* and the *Slide sub-object* are named "**Fixation**". While the name of the Slide Object is configurable, the name of the Slide sub-object is not; the WaitForFixation PackageCall requires a sub-object name "Fixation". Note the BorderWidth property on the Fixation sub-object's Frame tab has been changed from the default of 0 to 3. The border color that is specified in the WaitForFixation PackageCall (red in this case) will not be visible unless the sub-objects' border width is changed to a non-zero value.

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 green **Property Pages** button to open the **Fixation sub-object Properties.**

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



Task 13: Modify the Fixation Slide Object to work in Conjunction with the SMIWaitForFixation PackageCall

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

The general method employed by the SMIWaitForFixation 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 gaze data information. 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 Pages** button.

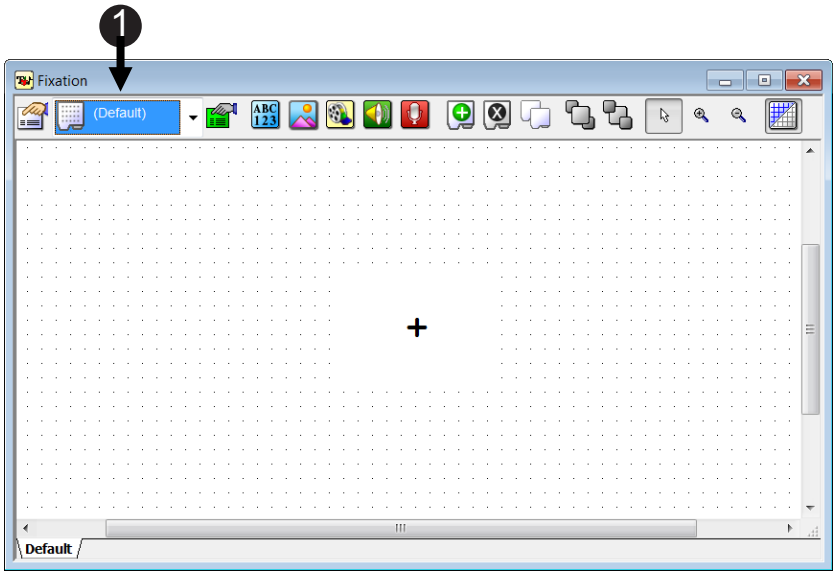
3) ***Select*** the **Duration/Input** tab.

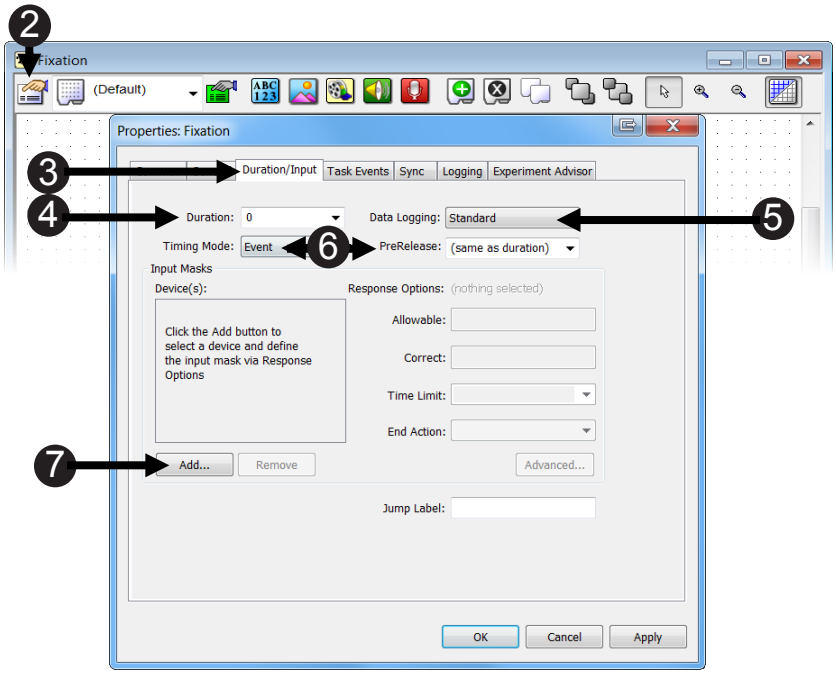
4) ***Edit*** the **Duration** to **0.**

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

6) ***Verify*** that the **Timing Mode** is **Event** and the **PreRelease** is **(same as duration).**

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





Task 13: (continued) Modify the Fixation Slide Object to Work in Conjunction with the SMIWaitForFixation PackageCall

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

Using EESMI, 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 SMIWaitForFixation is an example of “Active” processing.

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

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

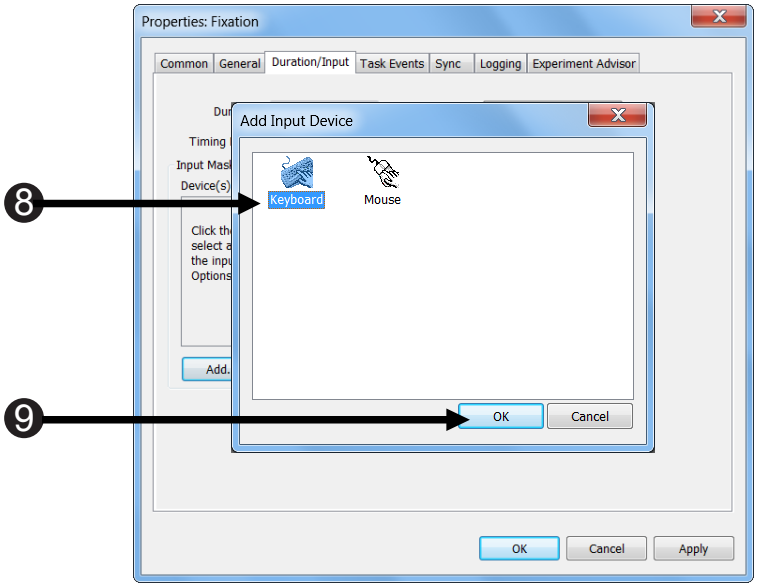
10) ***Verify*** that **Keyboard** is listed *under* **Device(s)** and that it is checked.

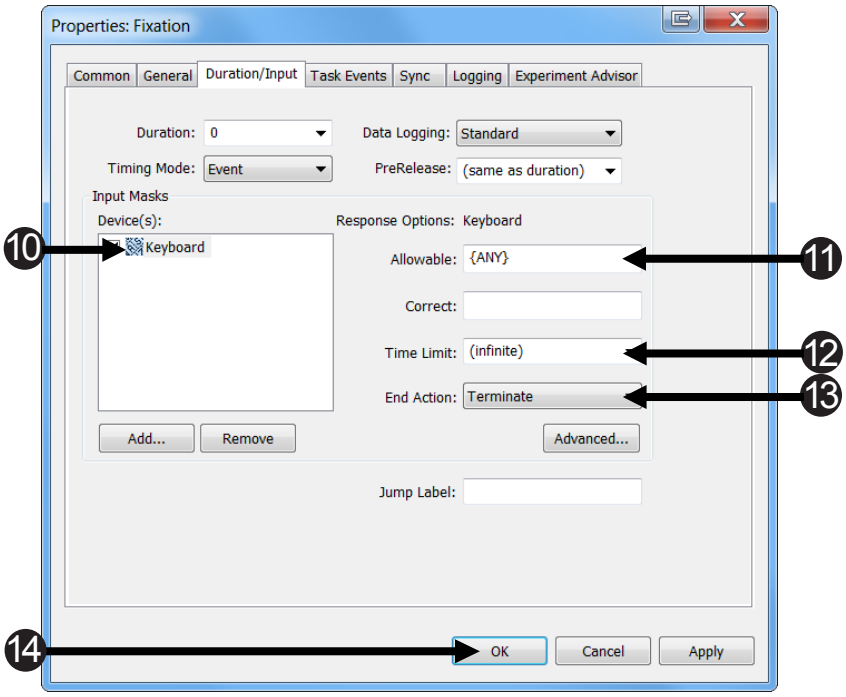
11) ***Verify*** that **Allowable** reads “**{ANY}**”.

12) ***Edit*** the **Time Limit** to “**(infinite)**”.

13) ***Confirm*** the **End Action** reads “**Terminate**”.

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





Task 13: (continued) Modify the Fixation Slide Object to Work in Conjunction with the SMIWaitForFixation PackageCall

Explanation of configuration and trouble shooting.

Troubleshooting:

A common mistake is failure to enable an Input Mask. If you encounter the following error message:

… go back and check the input properties of the Fixation Slide to see that they comply with these instructions.

Task 14: Add the SMIStopTracking PackageCall to Stop Tracking Eye Movements

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

Once the critical stimuli have been presented, it is appropriate to place the SMIStopTracking PackageCall. It is rare that an experimenter would want to collect gaze data 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.

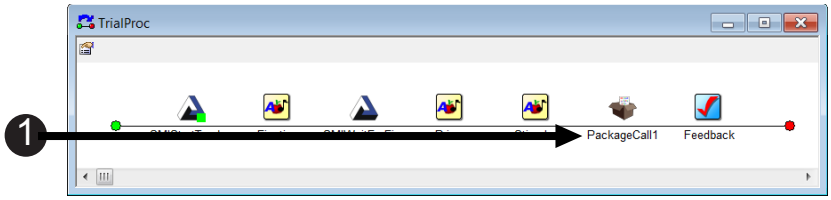
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Stimulus** Object in the **TrialProc** procedure. The object will be given a default name of **PackageCall1**.

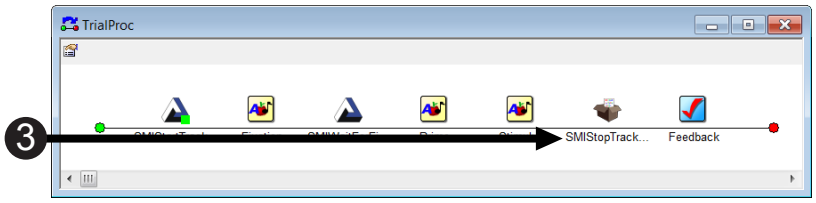
2) **Click** on the **PackageCall1** to select it then ***press*** **F2** to rename the Object.

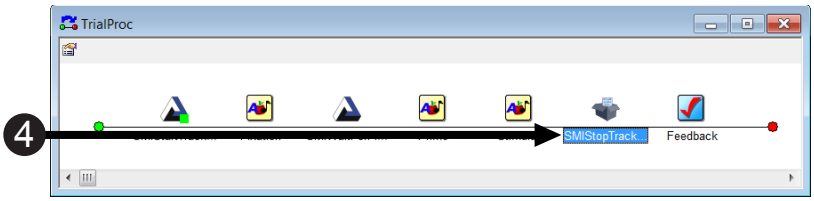
   You may alternatively right click on the Object and select Rename from the context menu.

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

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







Task 14 (continued): Add the SMIStopTracking PackageCall to Stop Tracking Eye Movements

Configure the SMIStopTracking PackageCall to call the StopTracking routine of the SMI package and accept the defaults.

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

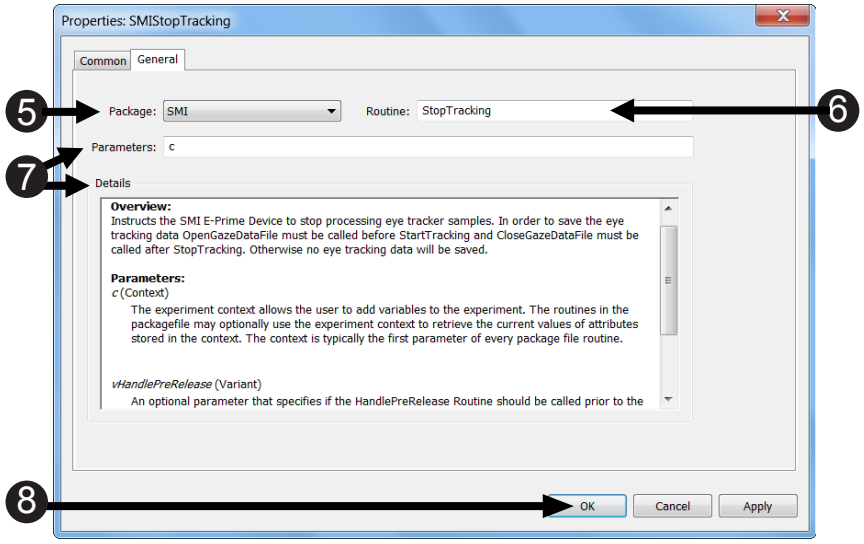
5) ***Select* SMI** from the **Package** dropdown list.

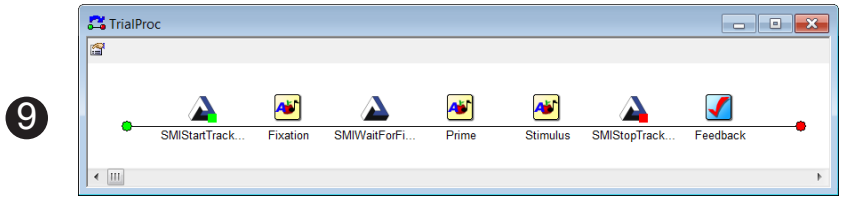
6) ***Select* StopTracking** from the **Routine** dropdown list.

7) ***Review*** the **SMIStopTracking** parameters listed in the **Parameters** and **Details** fields.

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

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





Task 15: Add the SMIGazeReplay PackageCall to Visually Replay the Eye Movements

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

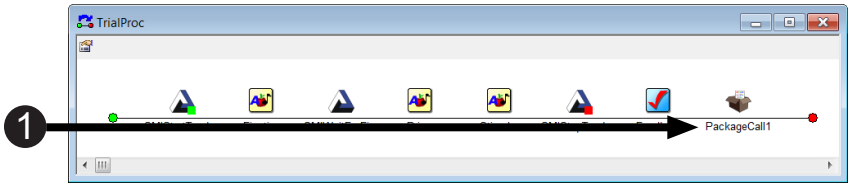
For this paradigm, we want to present the gaze replay as the last event in the trial sequence, so the PackageCall is inserted after the Feedback object. The PackageCall is added to the trial sequence here; see the next page for important parameter settings for the PackageCall.

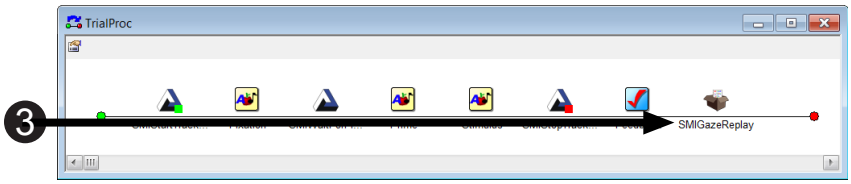
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Feedback** Object in the **TrialProc** procedure. The object will be given a default name of **PackageCall1**.

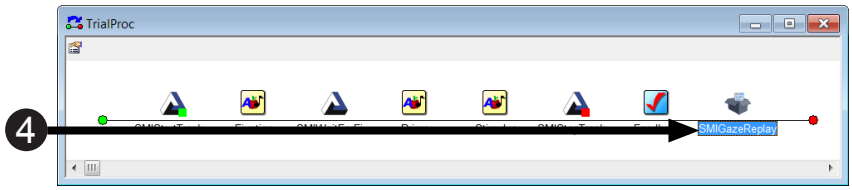
2) **Click** on the **PackageCall1** to select, it then ***press*** **F2** to rename the object.
   You may alternatively right click on the object and select Rename from the context menu.

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

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







Task 15 (continued): Add the SMIGazeReplay PackageCall to Visually Replay the Eye Movements

Configure the SMIGazeReplay PackageCall to call the GazeReplay routine of the SMI package and edit the Parameters.

In the SMIFixedPositionAOI experiment, the critical stimulus was presented by the Stimulus Object. The gaze replay can occur starting with any object that appears after the initial call to SMIStopTracking. For this tutorial, we will specify that replay starts when the Stimulus Object is displayed and continues until SMIStopRecording is called. It is possible to designate a specific time in which the gaze replay should stop via an optional parameter. However, it is unnecessary in this case because the SMIStopTracking PackageCall is called immediately after the offset of the Stimulus Object, which supplants the use of the optional parameters.

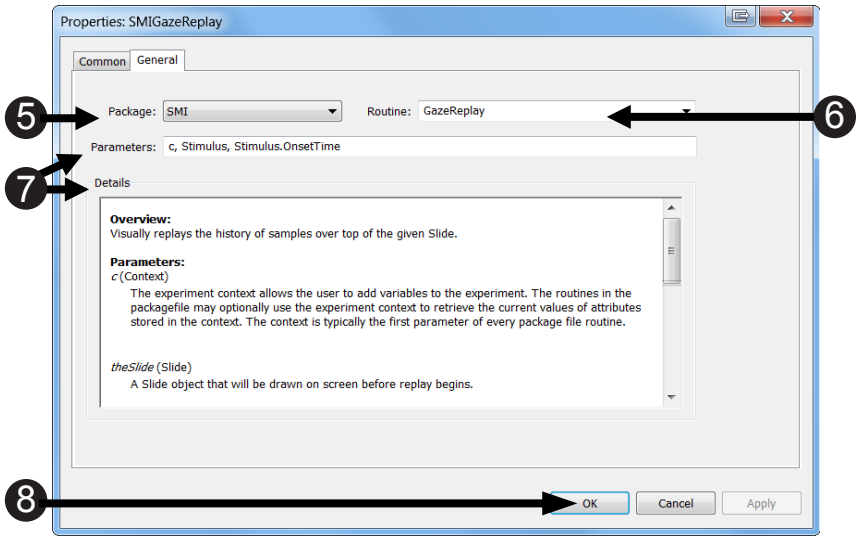
5) ***Select*** **SMI** from the **Package** dropdown list.

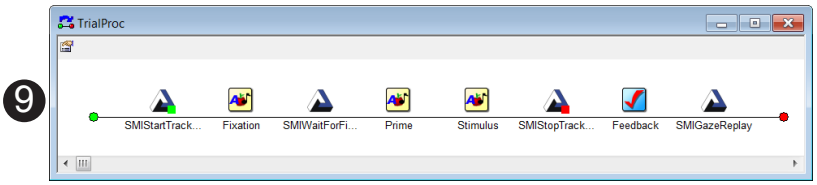
6) ***Select*** **GazeReplay** from the **Routine** dropdown list.

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

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

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





Task 16: Add the SMIStopRecording PackageCall

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

The SMIStopRecording PackageCall instructs the eye tracker to stop recording data. This PackageCall is placed as the first object on the SessionProc after the TrialList Session object, and is called after all of the trials have been completed. This PackageCall stops the output of eye tracking data to the .idf file.

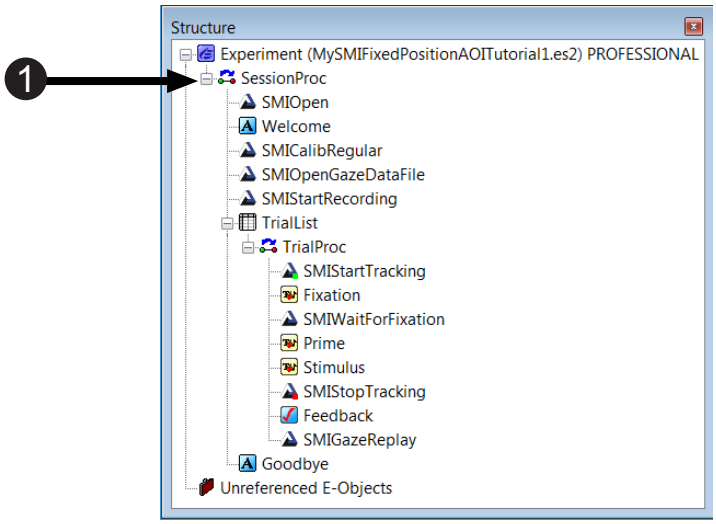
1) ***Double click*** the **SessionProc** Object to open it in the workspace.

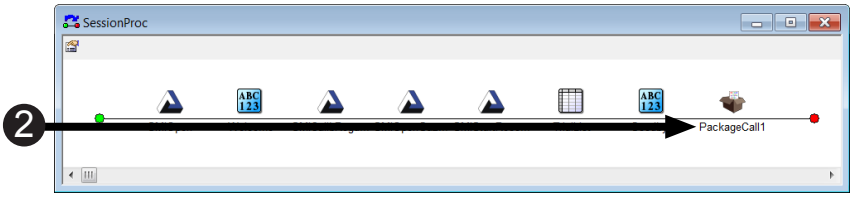
2) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **Goodbye** Object in the **SessionProc** procedure. The object will be given a default name of **PackageCall1**.

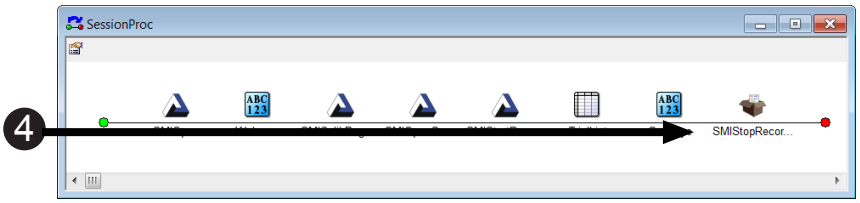
3) ***Click*** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.

   You may alternatively right click on the Object and select Rename from the context menu.

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







Task 16 (continued): Add the SMIStopRecording PackageCall

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

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

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

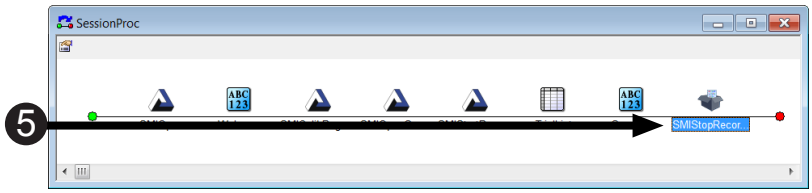
6) ***Select*** **SMI** from the **Package** dropdown list.

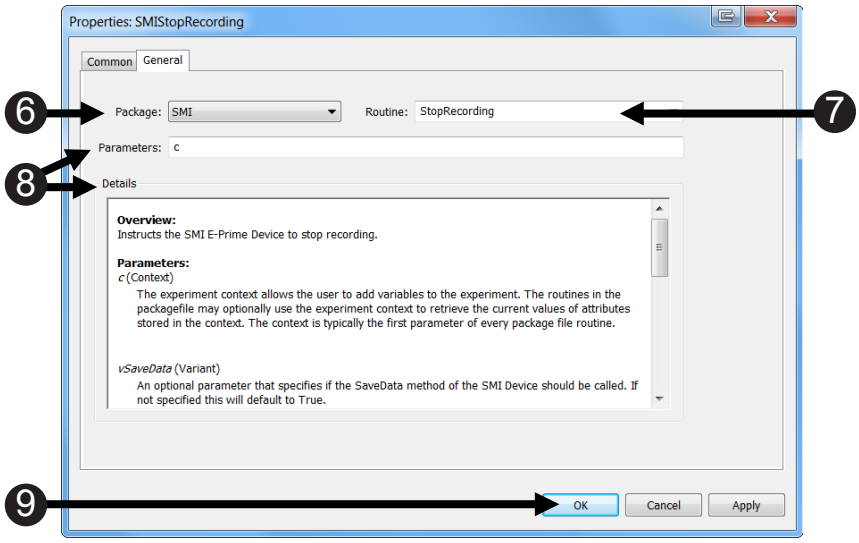
7) ***Select*** **StopRecording** from the **Routine** dropdown list.

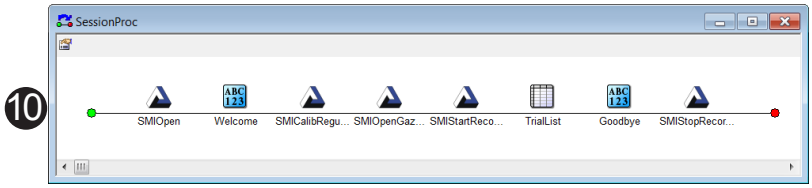
8) ***Review*** the **SMIStopRecording** parameters listed in the **Parameters** and **Details** fields.

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 17: Add the SMICloseGazeDataFile PackageCall to End Data Collection

Add a PackageCall at the end of the SessionProc and Name the PackageCall SMICloseGazeDataFile.

The end of the SessionProc is an appropriate location to add a SMICloseGazeDataFile PackageCall, because at this point we have stopped recording and will no longer need to save any gaze data information.

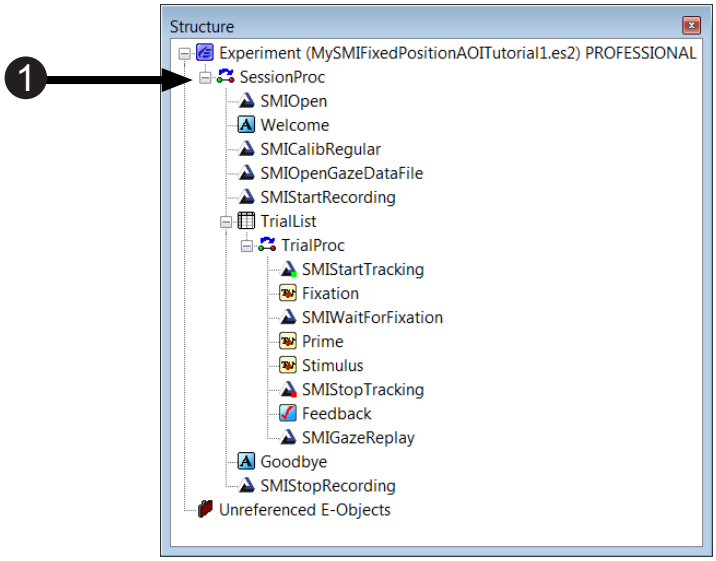
1) ***Double click*** the **SessionProc** Object to open it in the workspace.

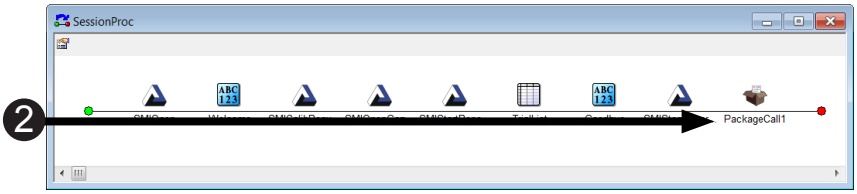
2) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** the **SMIStopRecording** Object in the **SessionProc** procedure. The object will be given a default name of **PackageCall1**.

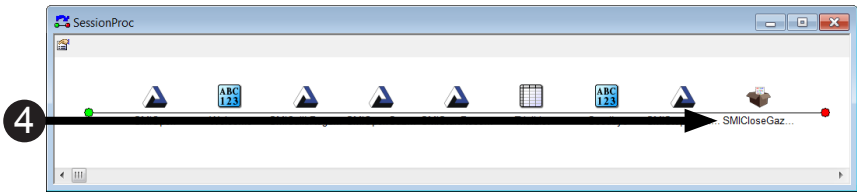
3) ***Click*** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.

   You may alternatively right click on the Object and select Rename from the context menu.

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







Task 17 (continued): Add the SMICloseGazeDataFile PackageCall to End Data Collection

Configure the SMICloseGazeDataFile PackageCall to call the CloseGazeDataFile routine of the SMI package and accept the default properties.

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

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

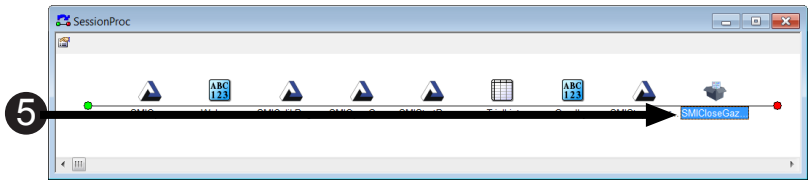
6) ***Select*** **SMI** from the **Package** dropdown list.

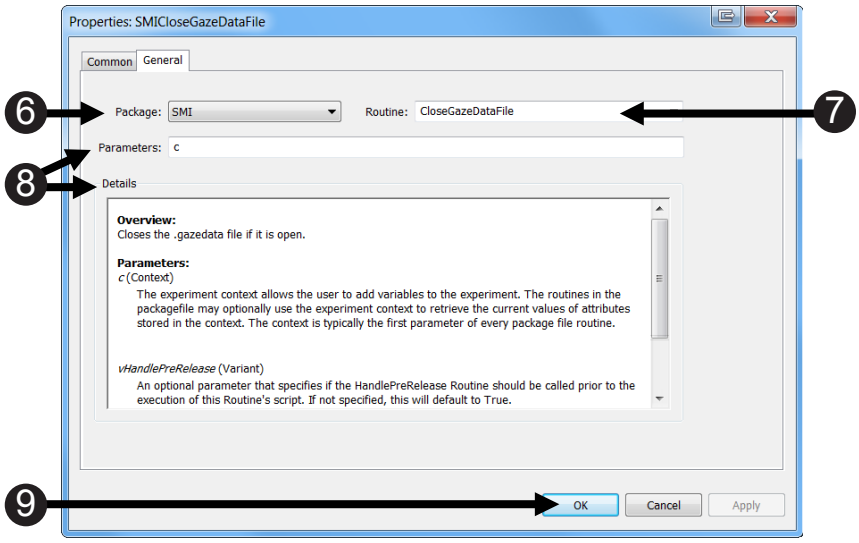
7) ***Select*** **CloseGazeDataFile** from the **Routine** dropdown list.

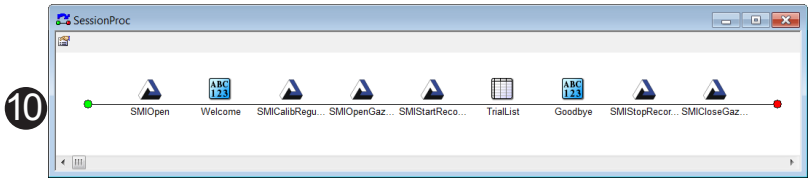
8) ***Review*** the **SMICloseGazeDataFile** parameters listed in the **Parameters** and **Details** fields.

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

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





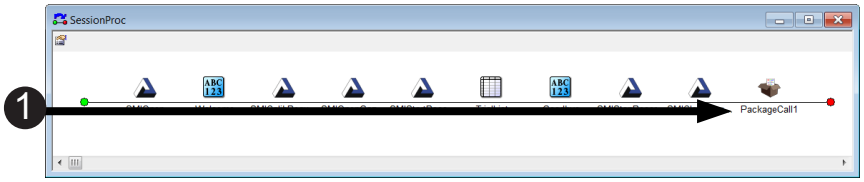


Task 18: Add the SMIClose PackageCall to Close the SMI device

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

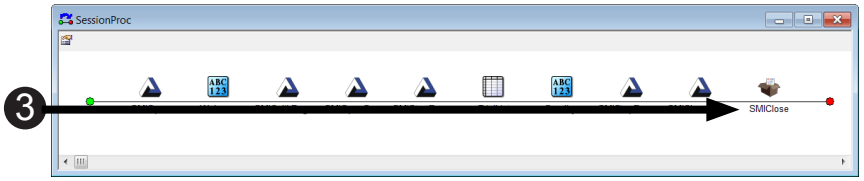
The last necessary PackageCall is **SMIClose**. This call will close the SMI device, and should be implemented once eye tracking is no longer necessary in your experiment.

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



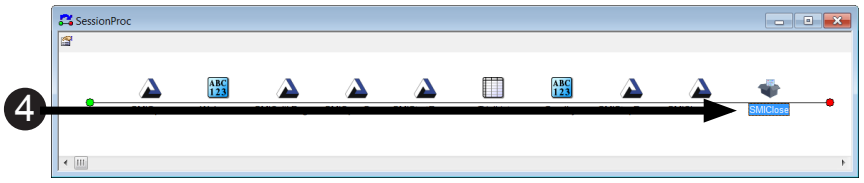
2) **Click** on the **PackageCall1** to select it then ***press*** **F2** to rename the object.

   You may alternatively right click on the object and select Rename from the context menu.



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

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



Task 18 (continued): Add the SMIClose PackageCall to Close the SMI Server

Configure the SMIClose PackageCall to call the Close routine of the SMI package and accept the default parameters.

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

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

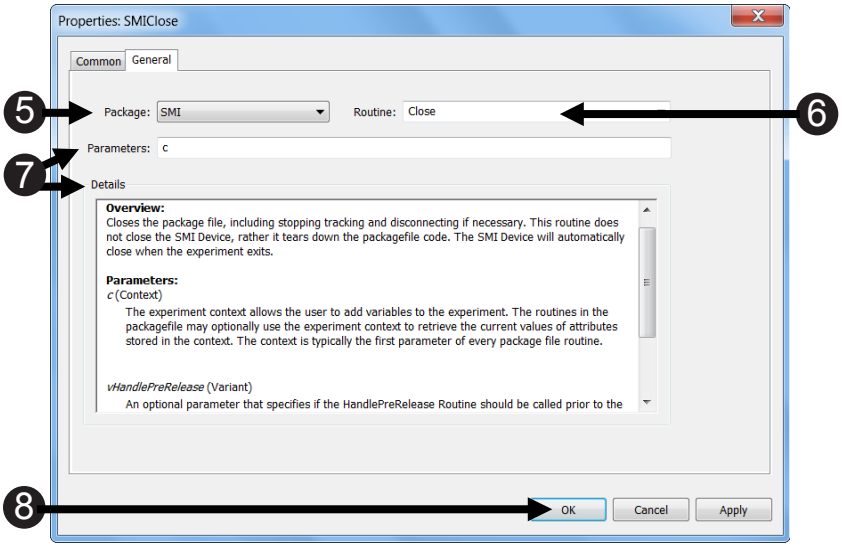
6) ***Select* Close** from the **Routine** dropdown list.

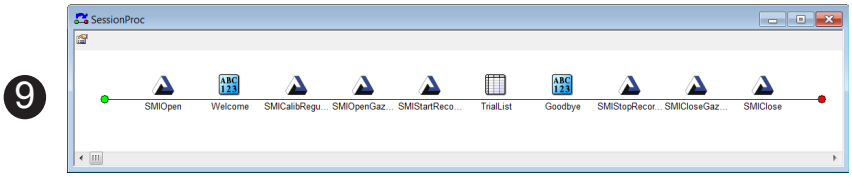
7) ***Review*** the **SMIClose** parameters listed in the **Parameters** and **Details** fields.

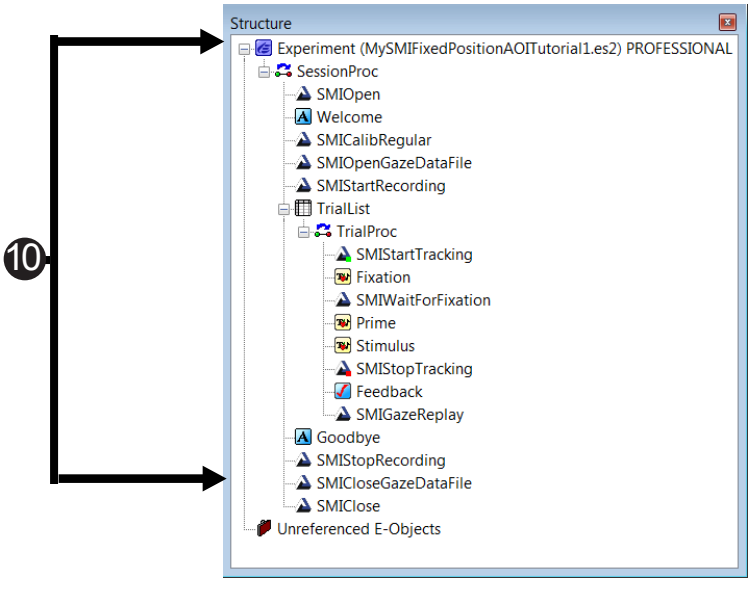
8) ***Click*** the **OK** button to accept the changes and dismiss the **SMIClose Property Pages**.

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.







Task 19: Run the Experiment

Run the experiment to verify that the SMI eye tracker device 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 SMI-enabled paradigm. E-Prime Extensions for SMI-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 experiment task, it is important that you understand how the task works, especially if you did not run through the FixedPositionAOI.es2 experiment prior to working through this tutorial. 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 the sound the 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. You will see feedback on the accuracy of your response. Your response time will also be displayed for correct responses. The experiment presents one block of three trials.

In addition, you need to understand how the track status, calibration, and validation routines that you added operate. Track status shows the position of the eyes with respect to the eye tracker. If the eyes do not appear in the track status window, or if they appear along with arrows, then move your head until the eyes appear or move your head in the direction indicated by the arrows. Once the track status display shows both eyes with no arrows, then press the space bar to continue.

Next, Calibration occurs. This tutorial uses SMI’s semi-automatic calibration method, which means that the first calibration dot remains on the screen until the space bar is pressed. After that, you will progress through the remaining calibration dots automatically once the eye tracker has calibrated your gaze.

Finally, Validation occurs immediately after calibration. You may need to press the space bar after the first validation dot appears, depending upon your specific eye tracker. If the first validation dot (upper left-hand corner of screen) remains on the screen, then press the space bar just as you did for calibration. The remaining three validation dots will appear and disappear automatically.

Task 19: (continued) Run the Experiment

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

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

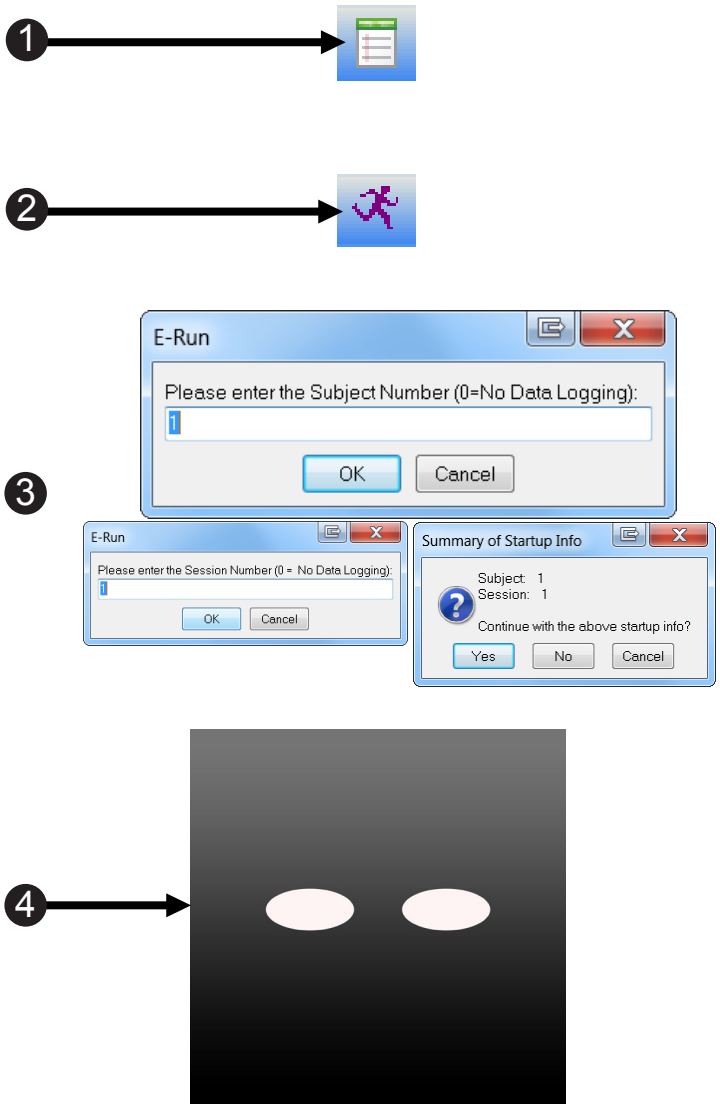
2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

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 **both eyes** are represented in the window and that **no arrows** appear. ***Press* Space** to close the track status window.

5) ***Perform* Calibration** (5 dots) and **Validation** (4 dots). ***Accept*** the **Calibration** results that are displayed, if x and y are each <=2 deg; otherwise, ***select*** the **"recalibrate"** option.

6) ***Perform*** the *three trials* in the experiment.



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

Summary:

Now that you have incorporated SMI support into your E-Prime experiment, the next step is to enable the experiment to write out the eye tracking data that is being collected to a file. E-Prime provides extensive capabilities for collecting, reporting, and viewing experimental data. In most designs, E-Prime is used to collect multiple independent and dependent variables and record these to the E-Prime data file (.edat2). Further, E-Prime data files are in the format of one data row for each trial, and E-Prime data files can be easily output to text files in the same format. For example, the current experiment logs “Prime” (which animal name is displayed in the center of the screen) as one of the independent variables and “Stimulus.RT” (the response time to the Stimulus display) as one of the dependent variables. The variable names can be output as column headings and the trial-level values can be output as rows.

Conversely, eye tracking experiments typically collect and output data for every eye gaze sample, of which there may be several hundred per trial depending on the trial duration. Because the typical eye tracking data set does not map readily to a typical E-Prime data file, EESMI paradigms are enabled to employ an additional data collection and output technique that results in the creation of a custom, tab delimited text file. This file contains data rows that are a combination of eye gaze data received by E-Prime from the SMI eye tracker device and any additional data added by the user via E-Prime. This file is referred to throughout the remainder of this manual as the .gazedata file. This tutorial illustrates how to create such a .gazedata file.

The type and amount of data you choose to log in the eye gaze data file is determined by your experimental design and data analysis needs. In order to allow for maximum flexibility, the methods used in this tutorial are implemented in script sections of the experiment, as opposed to package file calls. This technique exposes both the content and the format of the eye gaze data file, making it easier to customize in E-Studio.

⚠ **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 modify the starting design material to add a new piece of experimental data to the .gazedata file. You would use the same process shown in this tutorial to add your paradigm-specific information to the .gazedata file. (We recommend a conservative, “verbose” 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 SMIFixedPositionAOI.es2 experiment we previously created in the last tutorial.

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

⚠ **NOTE:** *In addition to the .edat2 and .gazedata files, your experiment should also generate an .idf file as long as you include the SMIStartRecording and SMIStopRecording PackageCalls as is done in the EESMI samples and tutorials experiments.*

Summary:

During this tutorial, you will modify an existing EESMI-enabled experiment to generate an output data file that contains both eye gaze information from the SMI eye tracker and experiment-specific information from the E-Prime experiment. E-Basic script is added, both as User Script to declare global variables and helper functions, and as InLine script to write out the data after each trial.

This tutorial assumes that you have some preliminary knowledge of the E-Basic scripting language. For a brief introduction to this topic, see **Chapter 5: Using E-Basic** from the *E-Prime 2.0 User's Guide.* However, in order to complete this tutorial more quickly and easily, we provide the majority of the E-Basic script that needs to be added to the experiment in two text files. You can simply open this script in a text editor, copy it to the clipboard, and then paste it into the appropriate location in the experiment (either User Script or an InLine object). The alternative would involve directly typing E-Basic script commands, which is not recommended, as it can be an error-prone process. The script examples that are provided here can be added to any eye-tracking experiment and then customized as needed.

Goal:

This tutorial illustrates how to write data to a tab delimited .gazedata file. Once you complete this tutorial, you will have a functioning EESMI-enabled E-Prime experiment that saves eye-tracking data to the gaze data file.

Overview of Tasks:

- Open the SMIFixedPositionAOITutorial1Completed.es2 experiment and resave as MySMIFixedPositionAOITutorial2.es2.
- Create User Script, which declares a Global User Defined Data Type named UserEyeGazeData.
- Add a new member to the end of the UserEyeGazeData data structure.
- Edit the script to append the new column label.
- Edit the script to append the new UserEyeGazeData member.
- 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 SMIFixedPositionAOIUserScript.txt into E-Prime User Script

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

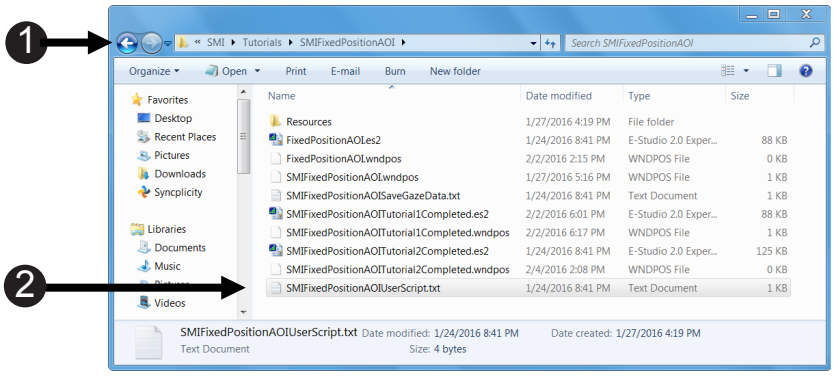
The UserEyeGazeData data structure is used to keep track of information specific to E-Prime conditional data per eye gaze observation. You may use it to help you to track and log any additional experiment related data that you need to associate with the gaze data. The quick and efficient way to get started is to copy the entire contents of the SMIFixedPositionAOIUserScript.txt included in the "…My Experiments\SMI\Tutorials\SMIFixedPositionAOI\" folder and paste it into the User tab of the Script view. Before beginning Task 1 below, open the E-Studio application, load the file "…My Experiments\SMI\Tutorials\SMIFixedPositionAOI\SMIFixedPositionAOITutorial1Completed.es2", and resave as MySMIFixedPositionAOITutorial2.es2.

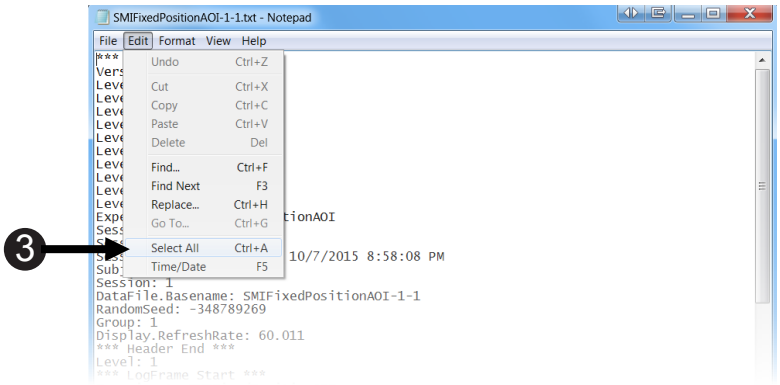
⚠ **NOTE:** *The script provided in the SMIFixedPositionAOIUserScript.txt and the SMIFixedPositionAOISaveGazeData.txt* (**Task 7: Copy Script from SMIFixedPositionAOISaveGazeData.txt to SaveGazeData InLine, Page 59-60**) *files can be copied to any EESMI-enabled experiment that you create. It was designed for use in different types of experiments, not just the FixedPosition paradigm, and contains commonly used variables. For each new experiment that needs to generate a .gazedata file and therefore use the E-Basic script described in this tutorial, you will need to alter some of the variable equivalencies to be specific to that particular experiment. This topic is explained throughout this tutorial, particularly in* **Task 9: Understand the UserEyeGazeData Type Variable Equivalents, see Page 62**.

1) ***Navigate*** to "…My Experiments \SMI\Tutorials\ SMIFixedPositionAOI\."

2) ***Double click*** the **SMIFixedPositionAOIUserScript. 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 SMIFixedPositionAOIUserScript.txt into E-Prime User Script

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

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

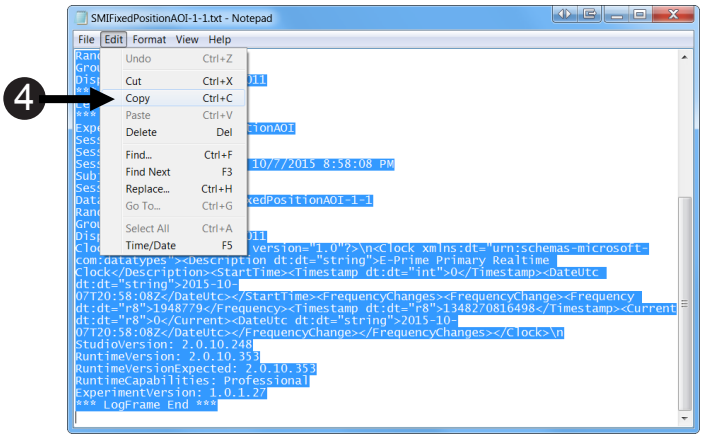
5) ***Return*** to the **E-Studio session.** If the **Script** window is not visible, ***press*** Alt+5 to open it.

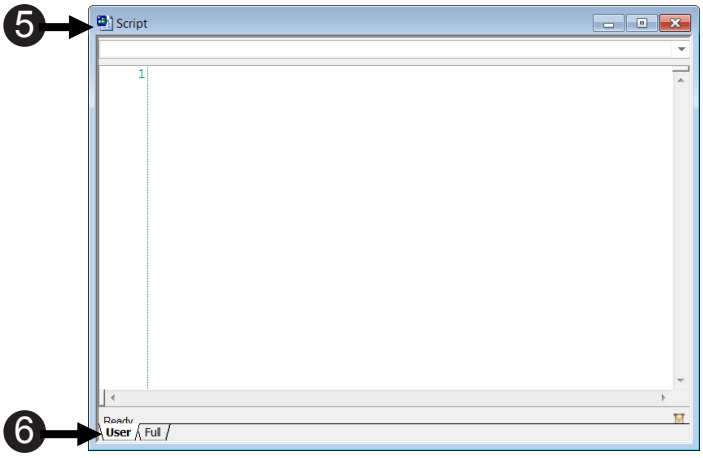
6) ***Select*** the **User tab.**

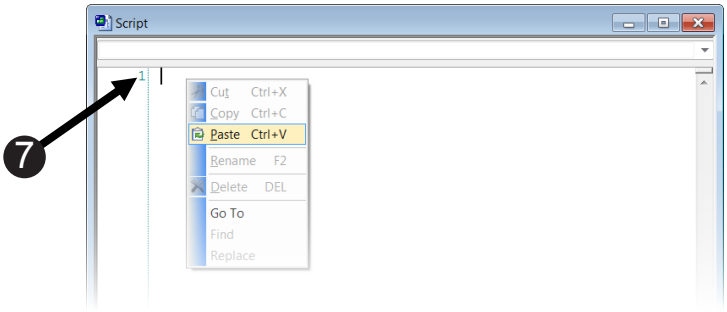
7) ***Select*** **Edit > Paste (Ctrl+V)** to ***paste*** the contents of the file into **E-Studio.**

8) ***Press*** the **Pg Up (Page Up) key** to move back to the top of the object.

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







Task 2: Understanding the UserEyeGazeData Type

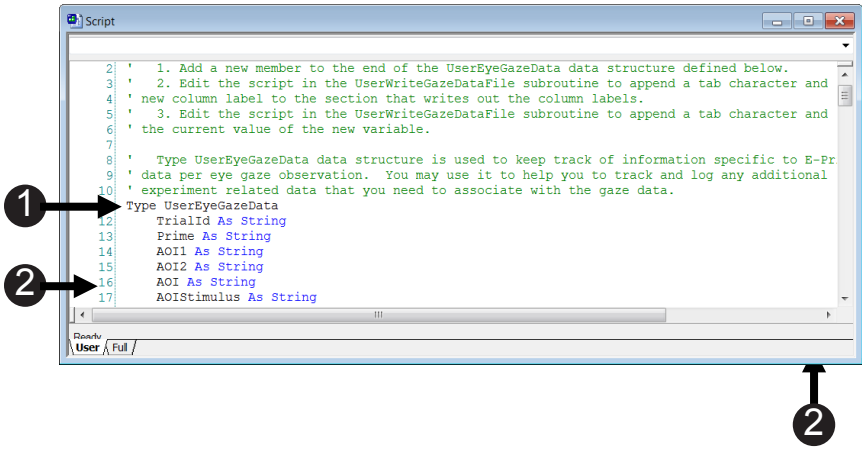
Select the User tab and examine the MySMIFixedPositionAOITutorial2.es2 script.

The UserEyeGazeData data structure defines the experimental data from E-Prime that will be associated with the SMI eye gaze data when the output .gazedata file is created. Before any piece of data is actually written out to the data file, as occurs in **Task 7: Copy Script from SMIFixedPositionAOISaveGazeData.txt to SaveGazeData InLine, see Page 59**, the variable that stores that data must be defined as part of the UserEyeGazeData data structure. 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 could be captured and contextualized with eye gaze data. Defining the UserEyeGazeData data structure is the first step in capturing this data. Refer to E-Prime's documentation for the full range of experimental data values that are logged and therefore available to be output as part of the .gazedata file. As an example, if your experiment displays an image on a colored background, you may wish to log the color of the background. This tutorial shows how to add the BackColor to the template that is provided for the .gazedata file.

Take a moment to become familiar with the template of the data structure which you have just pasted into the User tab of the Script Window. The variables in the data structure represent typical data that would be collected in most studies. For readability, we recommend that any additional values that you want to add to the .gazedata file be entered at the bottom of the structure. The order in which the variables appear is not important **as long the same order is used consistently** throughout the steps, which follow. You can also delete one or more of the variables that we have included. For example, in some designs it may not be necessary to log the independent variable Prime, or the dependent variable RT. If you have determined that you do not need to examine an experimental variable in your analysis, then you may remove values that are not necessary to your data file. Again, though, you must remove them in both the current task (defining the data structure) and in the later task, which writes out the .gazedata file values.

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

2) Each line in the User Script window is numbered in green. In certain E-Studio compile error messages, these numbers will help you locate the location of problematic script.



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

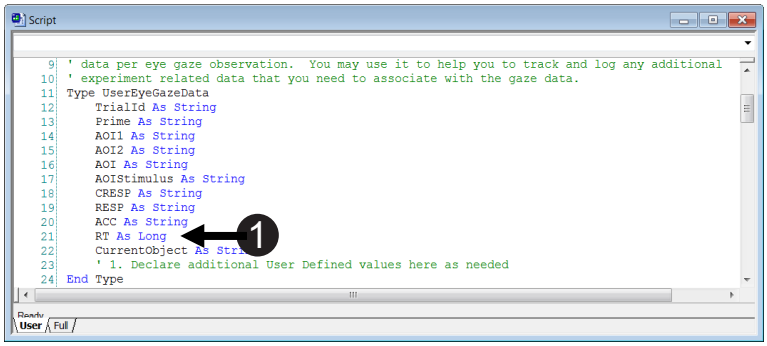
Edit the script at the end of the UserEyeGazeData type and declare the variable BackColor as a string.

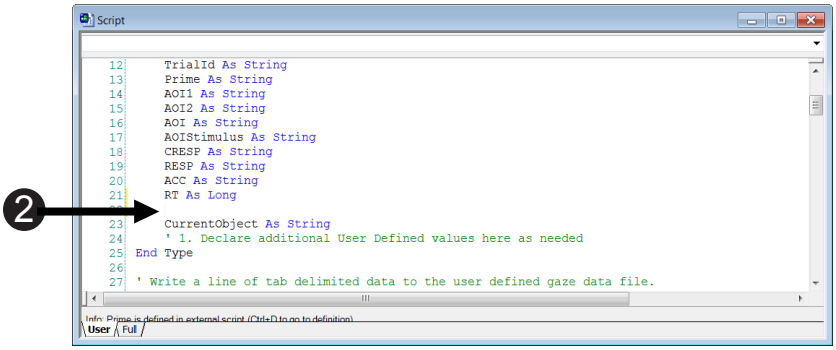
In this step, you will add the background color of the slide as a member of the UserEyeGazeData data structure. E-Prime assigns the value of this variable on every trial. By adding this variable to the data structure here, by making additional changes to the script on the User tab and by modifying InLine script in a later step, E-Prime will associate the background color with every eye gaze sample acquired while an instance of the slide was being presented.

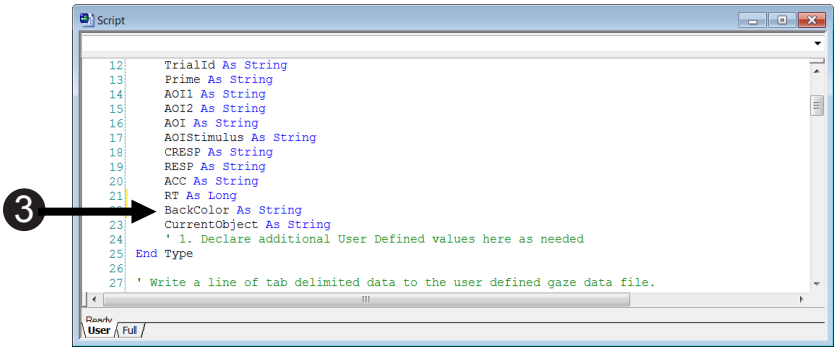
1) ***Position*** your cursor after the **g** on the line that reads "**RT As Long**".

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 entry to the list of column headings that are already defined in the User Script. Changes to the column headings are made in a section of the script labeled "Create columns to append additional user defined data." A comma separates the column headings. To enhance readability, we recommend placing each column name on its own line. With this layout, the underscore character ("_") indicates that the list of column names continues onto the next row. The order in which the columns appear here defines the order in which the columns appear in the .gazedata output file.

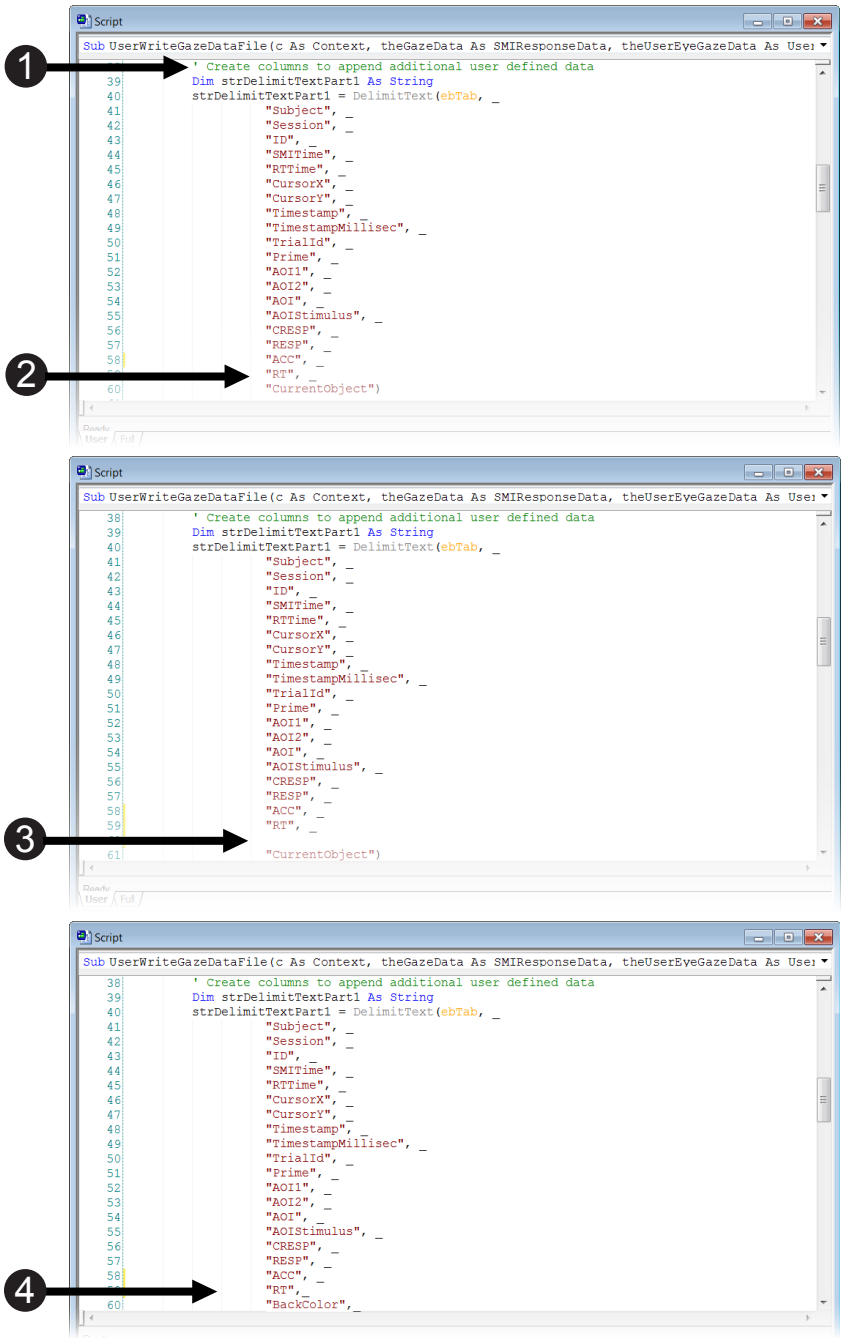
1) ***Locate*** 'Create columns to append additional user defined data in the **User Script** (it can be found in the middle of the script).

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

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

4) ***Type*** **"BackColor", _**

⚠ **NOTE:** *It's critical to include the underscore character ("_") after the comma.*



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.

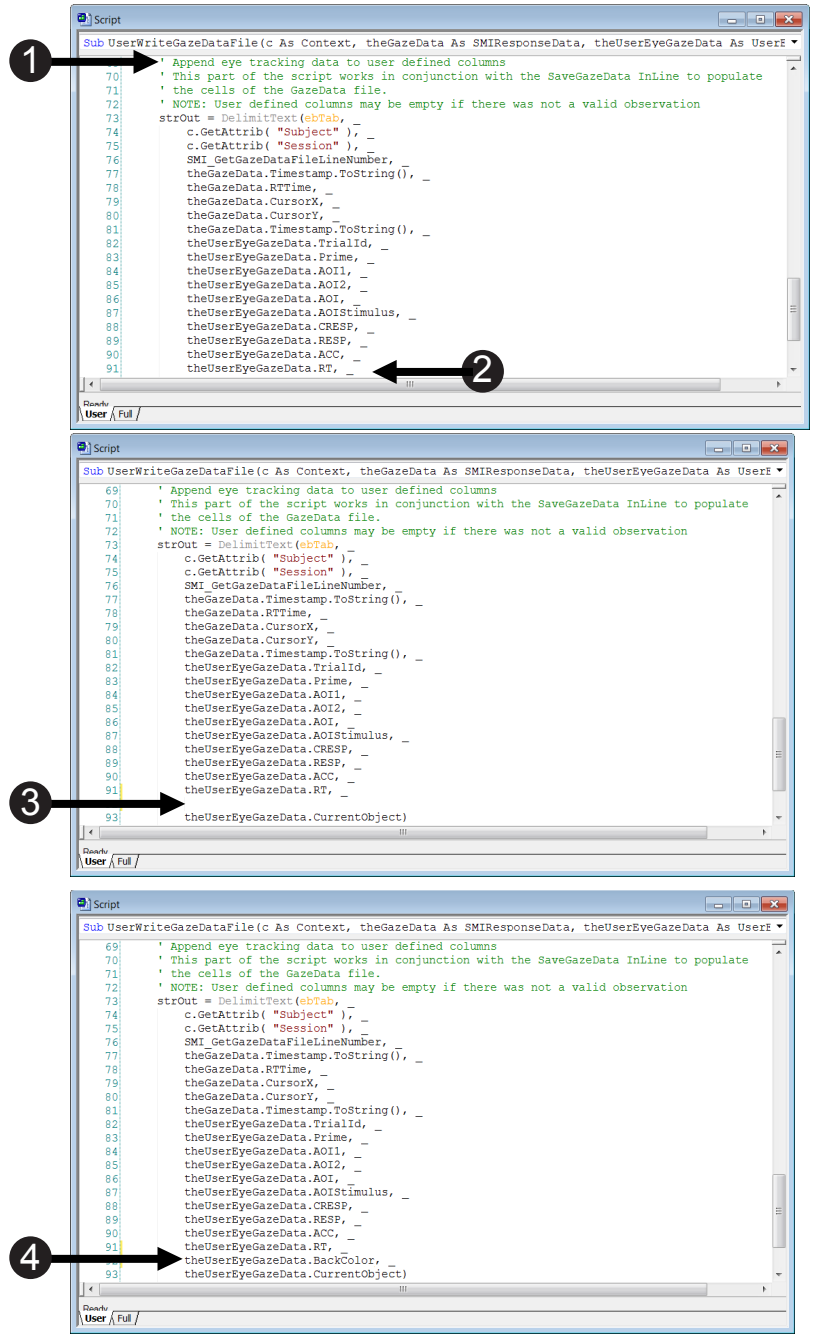
In Task 4, you modified the list of column headings to be output to the .gazedata file. In this task, you will modify the list of actual variable values that correspond to those column headings. The SaveGazeData InLine, which is created in the next Task, refers to the variables defined here to populate the cells in the .gazedata file. The order in which the variables appear in this step is critical and needs to match the order of the column headings from the previous step. 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 that is written out to the data file will not map correctly to the column headings. 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 SMI_WriteGazeDataFile line).

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

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

4) ***Type***: **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 that was just modified to write the data to a delimited .gazedata file. At the conclusion on an experimental run, the resulting .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 at the point at which you have stopped collecting eye tracking data for the current procedure. In this experiment, the InLine should follow the SMIStopTracking PackageCall.

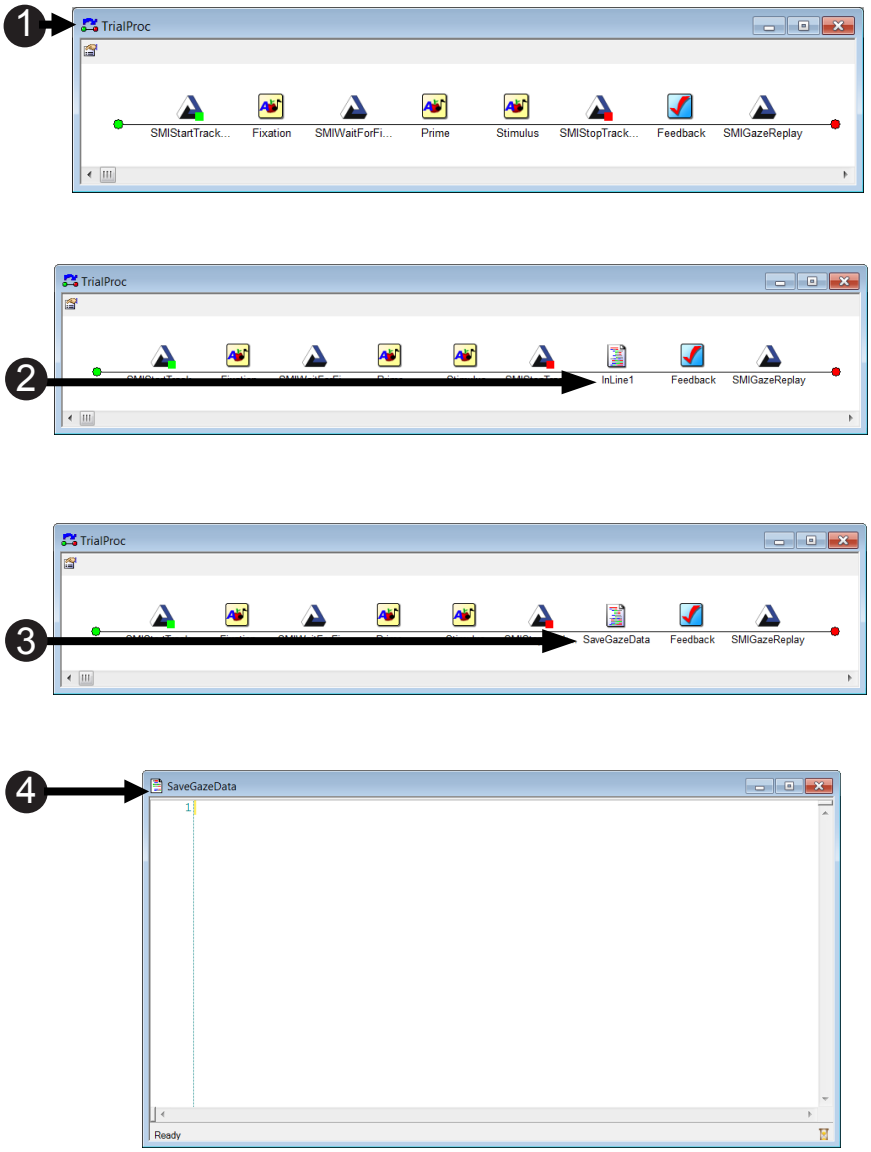
⚠ **NOTE:** *The script provided in the SMIFixedPositionAOIUserScript.txt (from previous Task 2) and the SMIFixedPositionAOISaveGazeData.txt files can be copied to every experiment that you create. It was written to be exported easily and contains commonly used variables. For each new experiment, you 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 **SMIStopTracking** PackageCall.
The **InLine** will be given a default name of **InLine1**.

3) ***Click*** the **InLine1** Object to highlight it and ***press*** **F2** to to ***rename*** it 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 SMIFixedPositionAOISaveGazeData.txt to SaveGazeData InLine

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

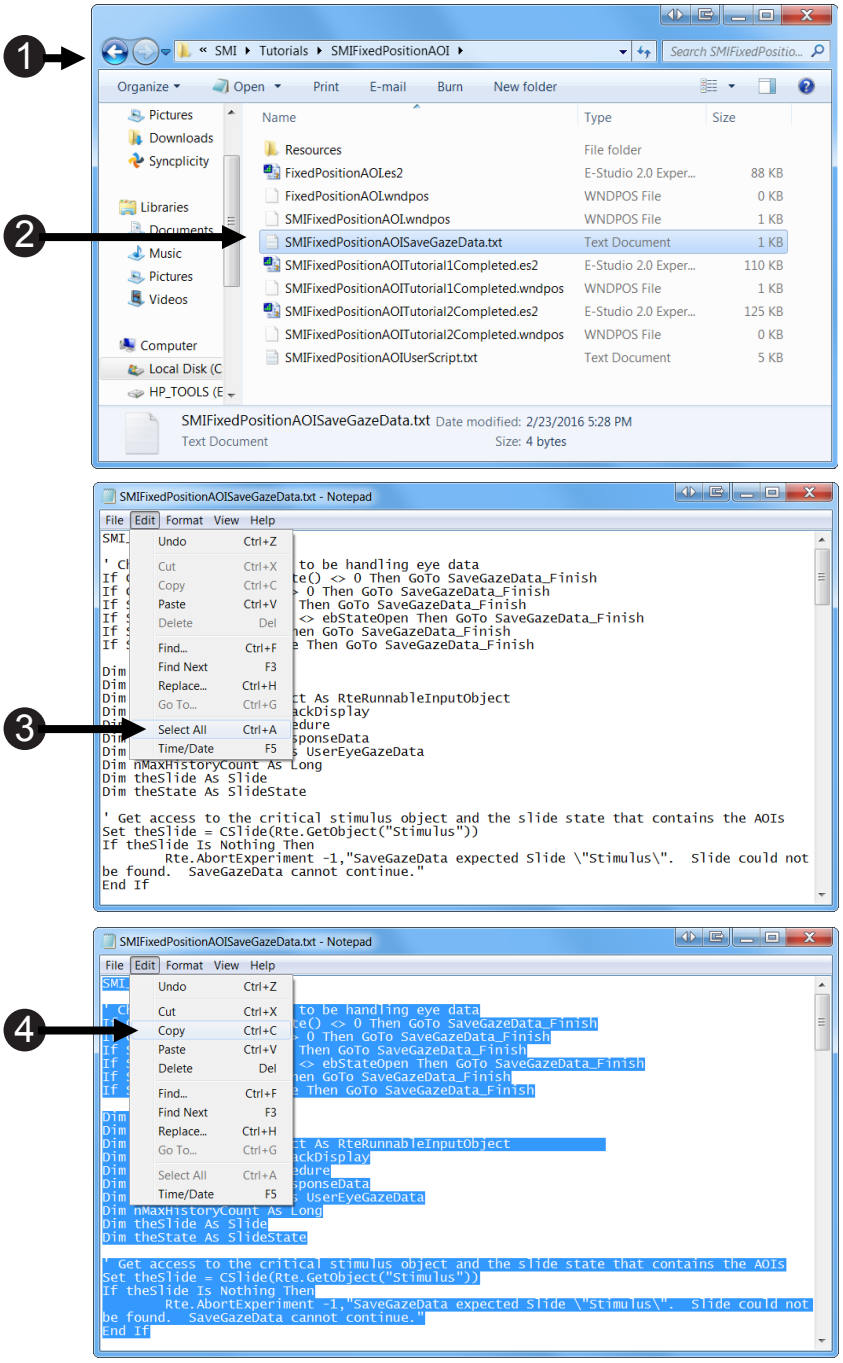
The InLine object that creates the .gazedata file contains over 100 lines of E-Basic script. Rather than require you to type that much script, we provide the necessary script in a text file. Just as you did in **Task 1: Copy the SMIFixedPositionAOIUserScript.txt into, see Page 52**, this task has you copy the script from the text file and place it in the appropriate location in the experiment (in this case, an InLine object). The remainder of this task walks you through the major components of the script and shows you how to add the BackColor variable to the set of data that is already being output to the .gazedata file.

1) ***Navigate*** to "…My Experiments \SMI\Tutorials\SMI\ SMIFixedPositionAOI\."

2) ***Double click*** the **SMIFixedPositionAOISave GazeData.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 to the clipboard.



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

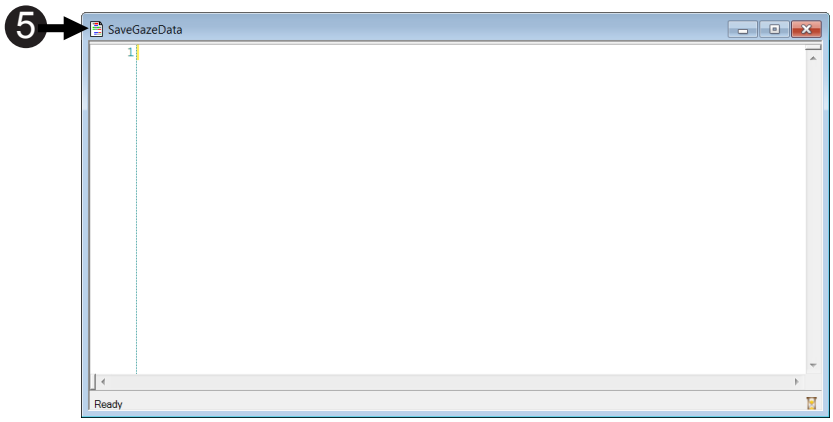
Open the SMIFixedPositionAOISaveGazeData.txt file, copy the contents, and paste it into the SaveGazeData InLine in E-Studio.

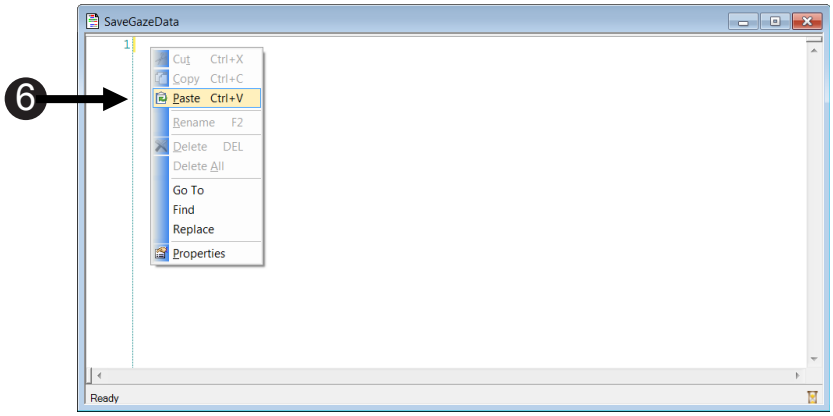
5) ***Click*** the **SaveGazeData InLine** in **E-Studio.**

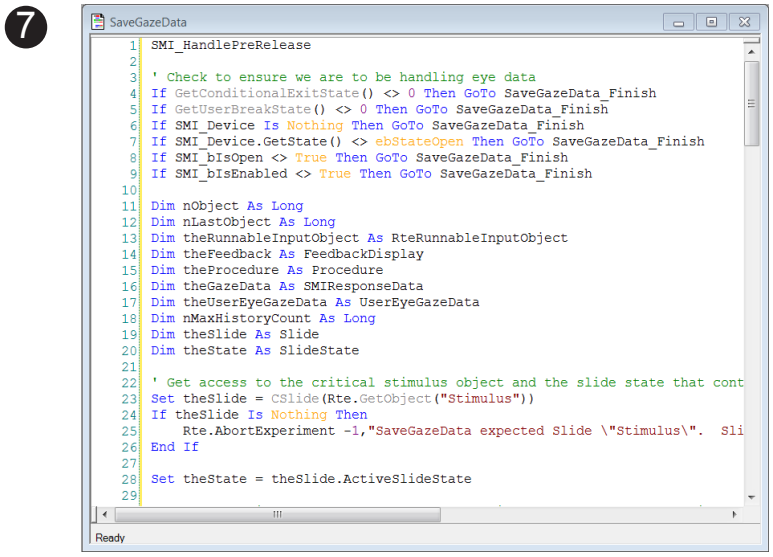
6) ***Select*** **Edit** > **Paste (Ctrl+V)** to ***paste*** the contents of the file into **E-Studio.**

7) ***Press*** the **Pg Up (Page Up) key** to move back to the top of the object.

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







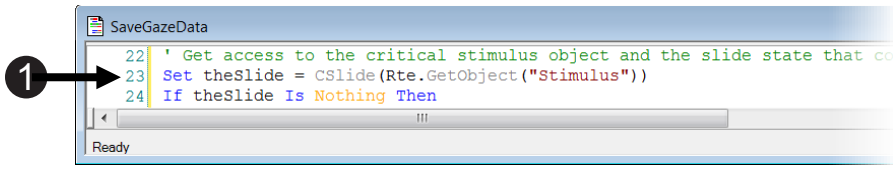
Task 8: Edit the SaveGazeData InLine

Confirm that the correct object ("Stimulus") is identified in the SaveGazeData InLine.

Next, we will confirm that the correct Slide object is identified in the SaveGazeData InLine. The script that was provided in the SMIFixedPositionAOISaveGazeDataFile.txt file was designed to be a general template that could 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, which is the Slide Object named "Stimulus".

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

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



Task 9: Understand the UserEyeGazeData Type Variable Equivalents

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

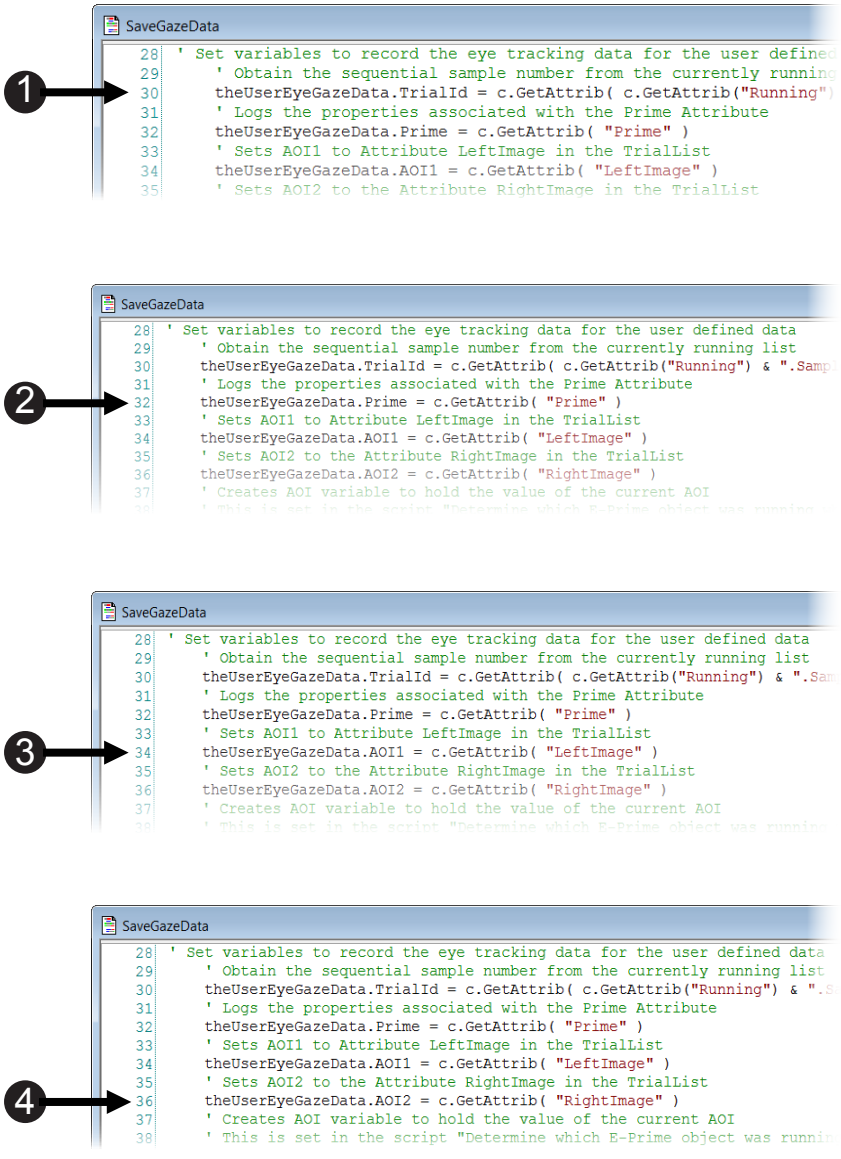
In addition to identifying the critical stimulus object, the SaveGazeData InLine object assigns values to the columns that will be output into the .gazedata file. The User tab on the Script window defined the names of the columns; the SaveGazeData InLine defines the variable values that are placed into those columns. The variables referenced on this InLine 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 **Task 11: Declare AOIs as Slide Sub-objects, Page 66**). Review 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. You are likely to need to customize this list of items for any of your own experiments.

1) ***Locate* theUserEyeGazeData.TrialId**
   This line obtains the sequential sample number from the currently running list of exemplars as defined on the TrialList object.

2) ***Locate* theUserEyeGazeData.Prime**
   This line logs the value of the attribute “Prime”, which will be the animal name that was displayed in the center of the screen.

3) ***Locate* theUserEyeGazeData.AOI1**
   This sets AOI1 to the attribute named “LeftImage” on the TrialList.

4) ***Locate* theUserEyeGazeData.AOI2**
   This sets AOI2 to the attribute named “RightImage” on the TrialList.



Task 9: (continued) Understand the UserEyeGazeData Type Variable Equivalents

Understand how the UserEyeGazeData type variables correspond to other Experiment Objects.

The variables provide information to the .gazedata file concerning which part of the Stimulus display is being gazed upon for the current gazedata observation. The script initializes the AOI and AOIStimulus variables, setting them both to a null string (“”). Later in the script, several lines below what is shown here, the AOI variable is set to an identifier (1 = image to left of fixation, 2 = image to right of fixation, “Fixation” = the fixation cross, and “” = an other area of the screen). The AOIStimulus variable identifies what information is in the current AOI (either name of the animal or “fixation”).

5) ***Locate* theUserEyeGazeData.AOI**
Initialize this variable to a null string. The final variable assignment is beneath the comment, “Determine which E-Prime object was running when this sample was taken”.

6) ***Locate* theUserEyeGazeData.AOIStimulus**
Initialize this variable to a null string. The final variable assignment is beneath the comment, “Determine which object is being viewed if the critical stimulus is on screen”.

7) ***Position*** your cursor at the end of the line that reads **“theUserEyeGazeData.RT = theSlide.RT**

8) ***Press* Enter**.

9) ***Type*** ‘Logs the background color
It is important to start the line with a single quote; this indicates that the characters which follow the quote are a comment and not E-Basic script.

10) ***Press* Enter**.

11) ***Type* theUserEyeGazeData.RT = theState.BackColor**
The background color of the Slide object will now be written out to the .gazedata file.

12) ***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. The variable assignments and output to a text file that occur in the SaveGazeData InLine object are performed between trials, during which no time-critical events are happening. In this tutorial, samples from the eye tracker device 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: The HitTest Method, see Page 67***. 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.

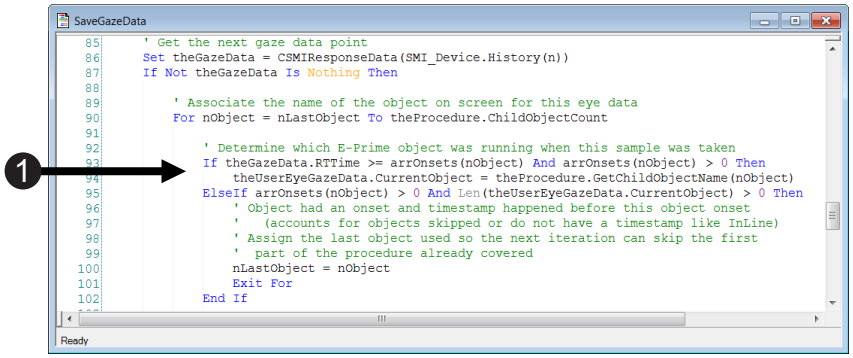
The screen image below shows how E-Basic script associates eye tracking data with the E-Prime object that is currently displayed on the screen. The RTTime property of the data structure named theGazeData provides a timestamp of when the current gaze data point was delivered to E-Prime. This value is then compared to the time at which each object in the TrialProcedure began to execute to determine which object was being displayed when the gaze data point was recorded.

The SaveGazeData InLine object contains script, not shown below, which first loops through all of the objects in the TrialProcedure up to the Feedback object and retrieves their OnsetTime property. The script shown below shows the comparison of this value to the RTTime.

Both sections of script are written generically: if you add, remove, or rename any object on the TrialProc, up to the Feedback object, this script will still operate as intended. You do not need to modify any of this script even if you modify the objects on the TrialProc.

1) ***Compare*** the eye gaze data time (**theGazeData.RTTime**) to the object's onset time (**arrOnsets(nObject)**)".

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



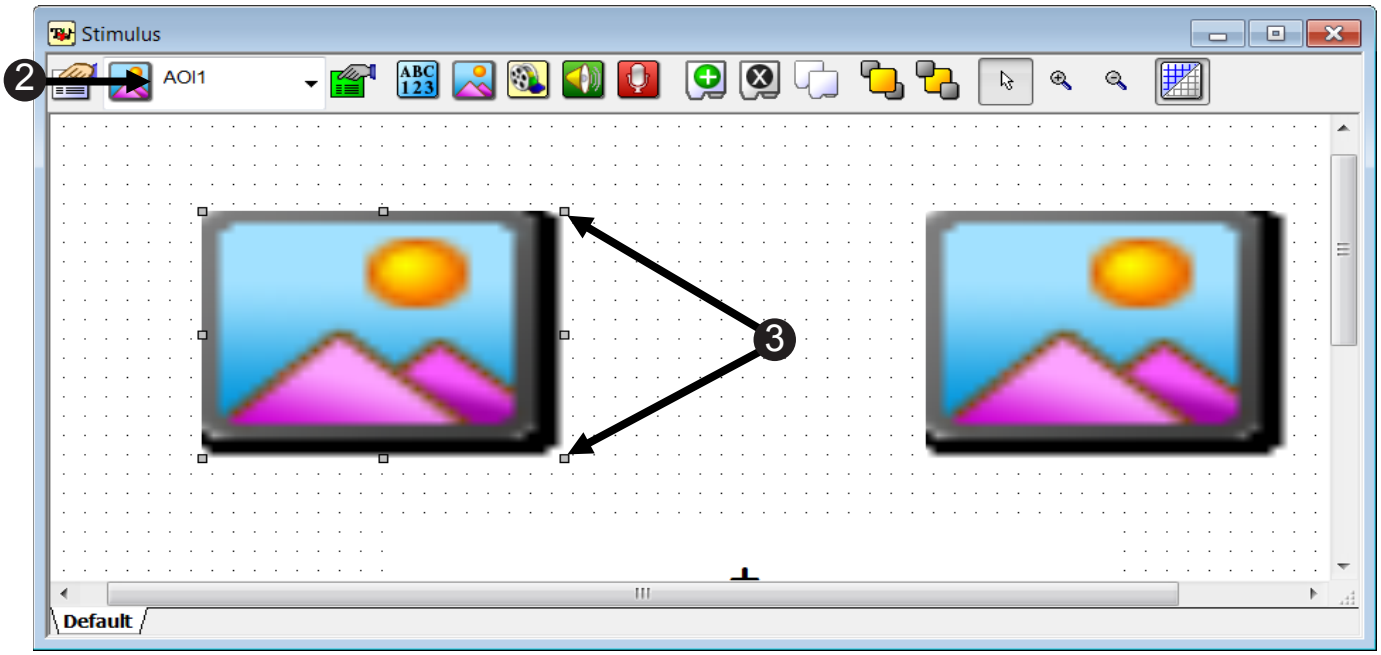
Task 11: Declare AOIs as Slide Sub-objects

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

Recall that in **Task 8: Edit the SaveGazeData InLine, Page 61**, we defined the critical stimulus as the Stimulus Object. Recall further the SaveGazeData InLine calls the HitTest method, which determines if the current eye gaze data falls into one of the pre-defined Areas of Interest on the critical stimulus object. This task explains where the output values of the HitTest method, the name of an AOI (Area of Interest), are defined.

Since the Stimulus Slide Object was already configured at the start of this tutorial, you have not yet reviewed the Stimulus slide sub-objects in detail. However, the images shown below illustrate that each sub-object has a name property. The name was specified when the Stimulus Slide was originally created. The HitTest method takes one parameter -- the name of the critical stimulus slide object, and outputs one value -- the name of the sub-object under which the current gazedata observation falls. The left-most slide sub-object on the Stimulus Display is named “AOI1”; the right-most slide sub-object on the Stimulus Display is named “AOI2”; and the fixation cross sub-object on the Stimulus Display is named “Fixation”. The HitTest method returns one of these names if the current eye gaze data sample falls within one of the Stimulus Slide sub-objects.

1) ***Double click*** the **Stimulus Object** to ***open*** it in the workspace.
2) ***Select*** **AOI1**.
3) 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 gaze data point is within.

E-Prime supports the HitTest Method, which determines if a coordinate pair is located within any of the sub-objects on a Slide. The HitTest method typically uses cursor or mouse coordinates, but for eye tracking studies, it can use the eye position as defined by .CursorX and .CursorY instead. The HitTest method returns the name of the Slide's sub-object within which the coordinates fall. The script below then assigns the name returned by the HitTest to theUserGazeData.AOI element, which is logged in the .gazedata file. The "Else" case in the script logs a null value for the AOI if the eyes are not gazing on one of the sub-objects. This script would need to be updated if you added or removed any sub-objects on the Stimulus Slide. For more information, see the **HitTest topic** in *E-Basic Help*.

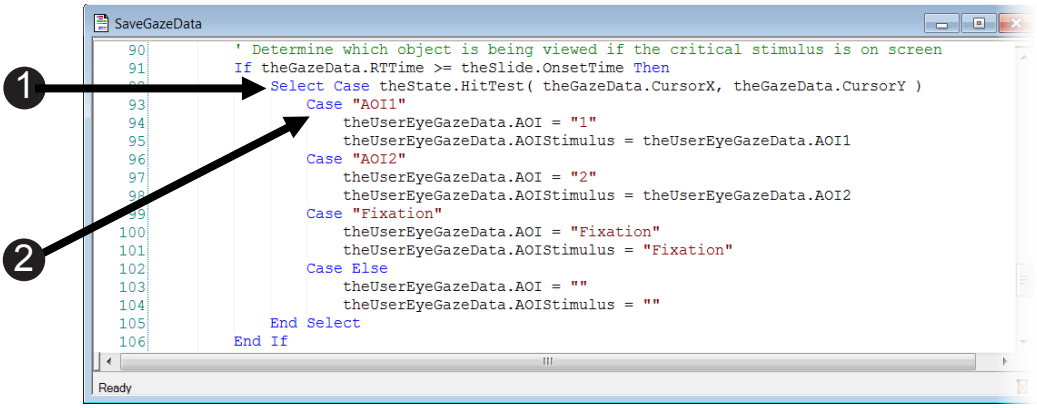
1) Command line that executes **HitTest**.

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.)



3
```
Case "AOI2"
    theUserEyeGazeData.AOI = "2"
    theUserEyeGazeData.AOIStimulus = theUserEyeGazeData.AOI2
```

4
```
Case "Fixation"
    theUserEyeGazeData.AOI = "Fixation"
    theUserEyeGazeData.AOIStimulus = "Fixation"
```

5
```
Case Else
    theUserEyeGazeData.AOI = ""
    theUserEyeGazeData.AOIStimulus = ""
```

Task 13: Run the Experiment

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

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

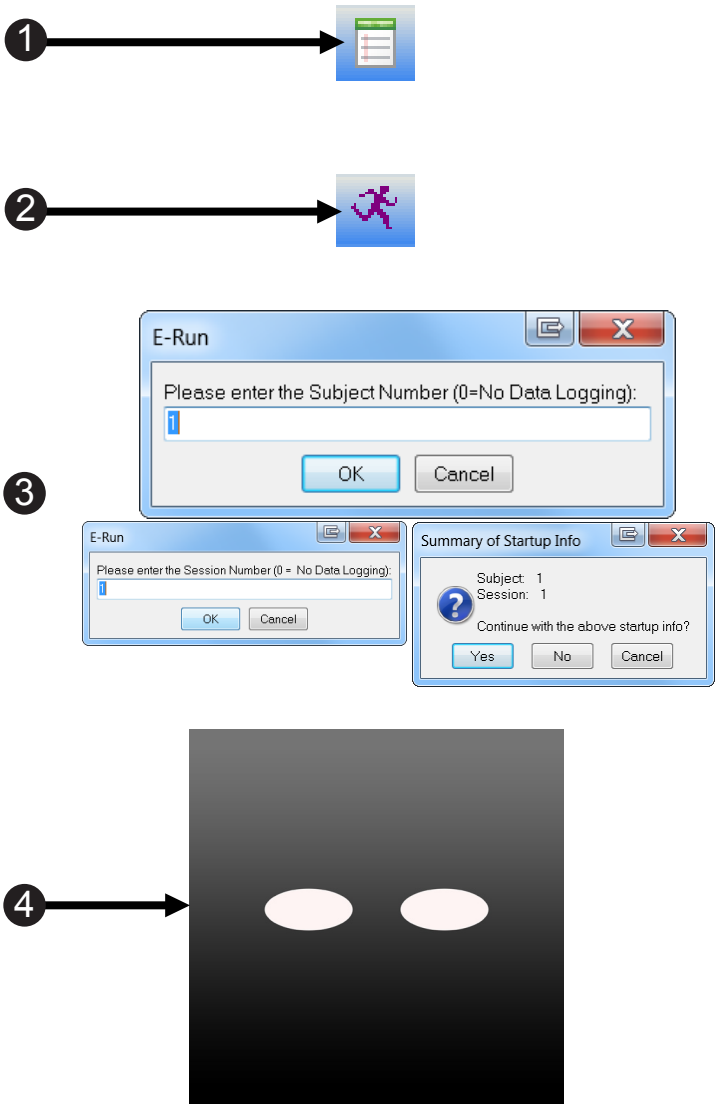
2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

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 **both eyes** are represented in the window and that **no arrows** appear. ***Press* Space** to close the track status window.

5) ***Perform* Calibration** (5 dots) and **Validation** (4 dots). ***Accept*** the **Calibration** results that are displayed, if x and y are each <=2 deg; otherwise, ***select*** the **"recalibrate"** option.

6) ***Perform*** the *three trials* in the experiment.



Task 14: Open the Eye .gazedata File

Verify the .gazedata file exists and can be opened.

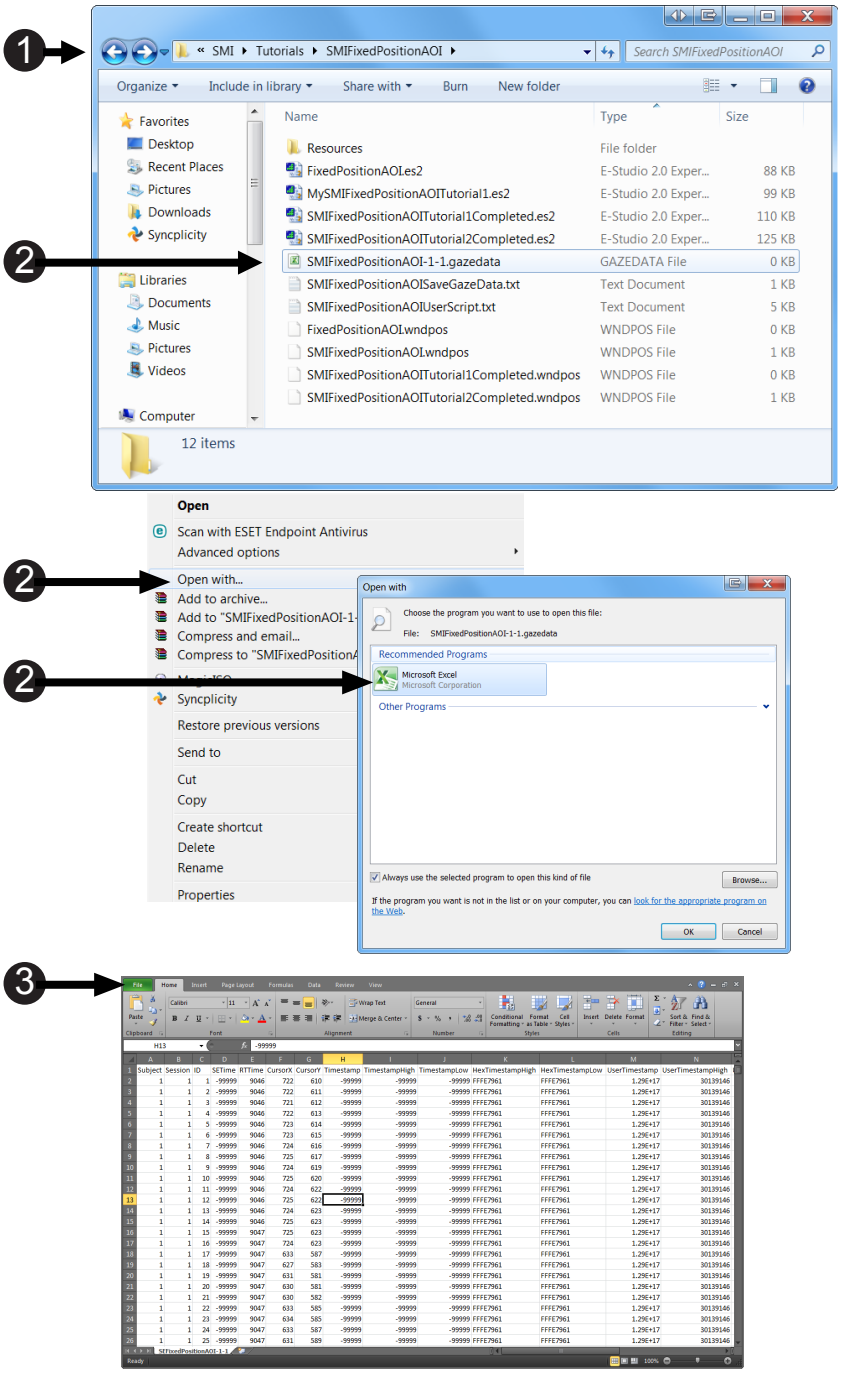
The .gazedata file contains both information about the stimulus presentation, e.g. what was on screen and when it was on screen, and the eye tracking data, e.g. where the eyes were looking. This file will facilitate data analysis. After the experiment is created, it is important that you check this file 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, which defaults to [Experiment Name]-[Subject#]-[Session#].gazedata. In this example, if you followed the directions in Task 1 then the file will be named MySMIFixedPositionAOITutorial2-1-1.gazedata.

1) ***Navigate*** to ...My Experiments\SMI\ Tutorials\SMIFixedPositionAOI.

2) ***Right click*** the **SMIFixedPosition 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 run 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 experiment that was completed in Tutorial 2. Otherwise, you will not have a data file to compare to the screen shots in this manual.*

If you have not completed Tutorial 2, including running the experiment to generate a .gazedata file (**Tutorial 2, Task 13: Run the Experiment, Page 68**), then you need to do so now.
Open the experiment "....My Experiments\SMI\Tutorials\SMIFixedPositionAOI\SMIFixedPositionAOITutorial2Completed.es2". Before you run the experiment, save it under the name, "MySMIFixedPositionAOITutorial2Completed.es2".

Summary:

In the previous tutorial you enabled the experiment to write eye tracking and stimulus data collected during the experimental run to a file. This resulted in the creation of a tab delimited text file containing rows that are a combination of eye gaze data received by E-Prime from the SMI eye tracker device additional data added by the user via E-Prime. E-Basic script, both in the form of User Script and InLine script, was added to the experiment in order to collect and manipulate the format of this output file. This tutorial will offer a review of the resulting .gazedata file that was created from this script.

Since the layout and content of the .gazedata file is controlled by the E-Basic script commands, it is paramount to understand how the script works. It is also helpful to have reviewed the script in order to help you troubleshoot if problems arise during testing.

User Script 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. This tutorial will illustrate specifically which parts of the script output what parts of the .gazedata file.

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. Once you finish the tutorial, you will understand the relationship between the User Script and the SaveGazeData InLine and the ways in which they work together to create the .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 that was created from MySMIFixedPositionAOITutorial2Completed.es2 to verify the data written to the file.
- Learn how the .gazedata file is created.

Estimated Time: 10-15 minutes

⚠**NOTE:** *The name of your .gazedata file will vary depending on the subject- and session- number that were used.*

Task 1: Introduction to .gazedata File

Understand the general output from the .gazedata file.

The .gazedata file contains information about eye movements from the SMI eye tracker device 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. **Appendix C: Gazedata File Content, see Page 136,** 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.

Task 2: Open the .gazedata File and Examine the Structure and Content

View the User-defined variables in the .gazedata file

The Tracking Mode at which your SMI eye tracker is set determines the rate at which eye gaze data is collected. Each time 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 69** for instructions on how to navigate to and open the .gazedata file. The SaveGazeData InLine begins with the SMI_HandlePreRelease line. The purpose of this code is to prevent you from losing any data. For more information, please refer to **Chapter 3: SMI PackageCall Reference, Page 105**.

⚠ **NOTE:** *The line numbers that appear in the SaveGazeData and UserScript images may differ slightly from the line numbers in your copy of the these objects.*

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.

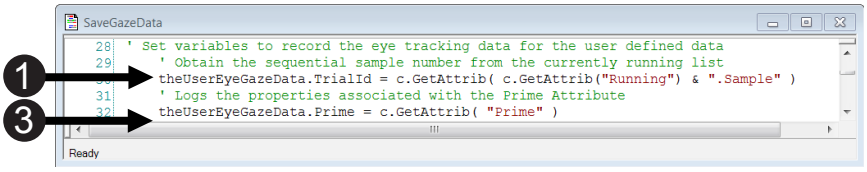
2) The **TrialID column *increases*** for each trial. The **first trial** is labeled **1** and the **second trial** is labeled **2**.

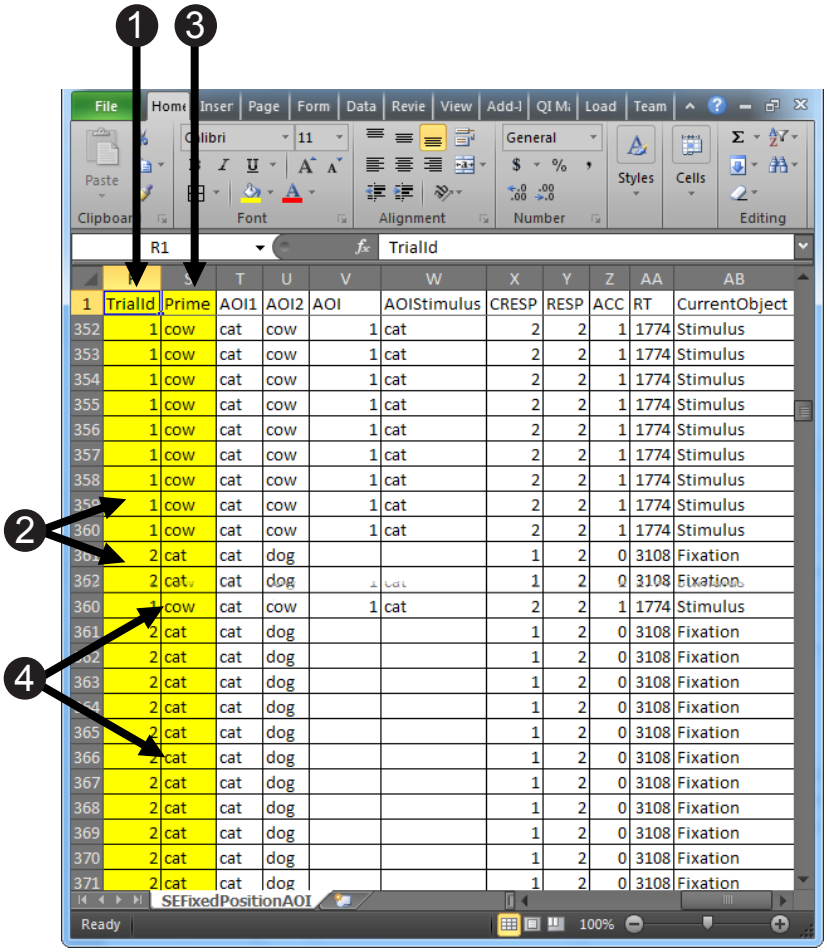
3) **Prime Column**: Script used to assign variable. (location: SaveGazeData InLine):

   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. In this example, the **first Prime** was **cow**, and the **second Prime** was **cat**.





	TrialId	Prime	AOI1	AOI2	AOI	AOIStimulus	CRESP	RESP	ACC	RT	CurrentObject
352	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
353	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
354	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
355	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
356	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
357	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
358	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
359	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
360	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
361	2	cat	cat	dog			1	2	0	3108	Fixation
362	2	cat	cat	dog			1	2	0	3108	Fixation
360	1	cow	cat	cow	1	cat	2	2	1	1774	Stimulus
361	2	cat	cat	dog			1	2	0	3108	Fixation
362	2	cat	cat	dog			1	2	0	3108	Fixation
363	2	cat	cat	dog			1	2	0	3108	Fixation
364	2	cat	cat	dog			1	2	0	3108	Fixation
365	2	cat	cat	dog			1	2	0	3108	Fixation
366	2	cat	cat	dog			1	2	0	3108	Fixation
367	2	cat	cat	dog			1	2	0	3108	Fixation
368	2	cat	cat	dog			1	2	0	3108	Fixation
369	2	cat	cat	dog			1	2	0	3108	Fixation
370	2	cat	cat	dog			1	2	0	3108	Fixation
371	2	cat	cat	dog			1	2	0	3108	Fixation

Task 2 (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 SMIFixedPositionAOI.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.

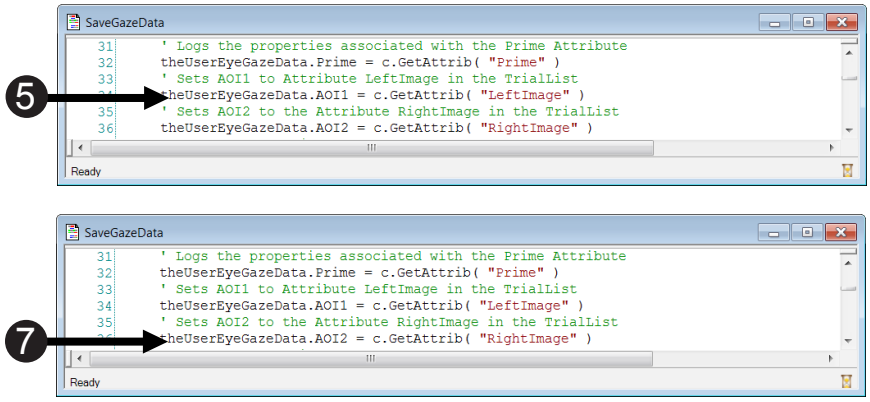
6) The **AOI1** is ***assigned*** the **LeftImage attribute** from the TrialList. The **AOI1 column *shows*** what image was displayed on the left side of the screen on a trial by trial basis. In this example, the LeftImage is a cat.

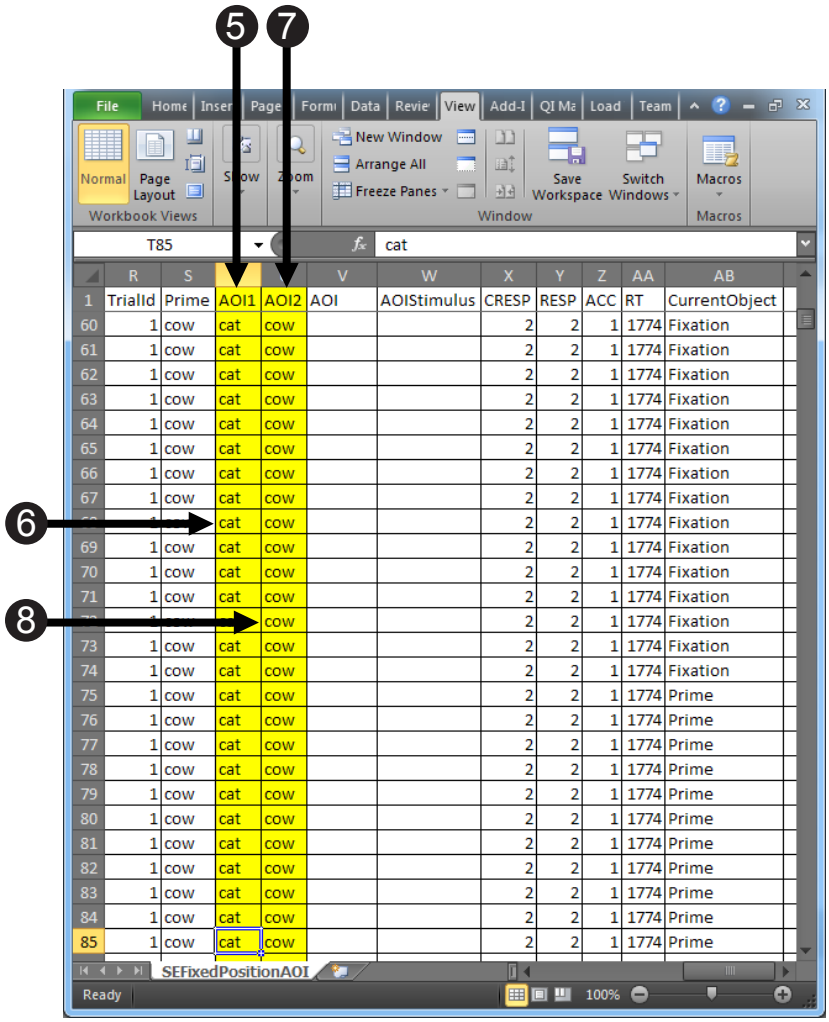
7) **AOI2 Column**: Script used to assign variable.
(location: SaveGazeData InLine):

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 3: 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 which hold information about what was on screen at a certain point in time. This information is determined in the UserScript within the experiment. Below, in the screenshot numbered 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). Like the SaveGazeData InLine the UserScript also begins with the SMI_HandlePreRelease line. The purpose of this code is to prevent you from losing any data. For more information, please refer to **Chapter 3: SMI PackageCall Reference, Page 105**.

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 = " "

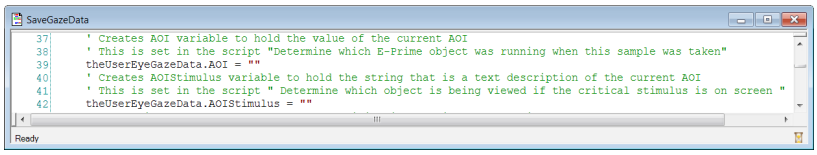
   Purpose: Creates AOI variable to hold the value of the AOI the participant is currently looking at.

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

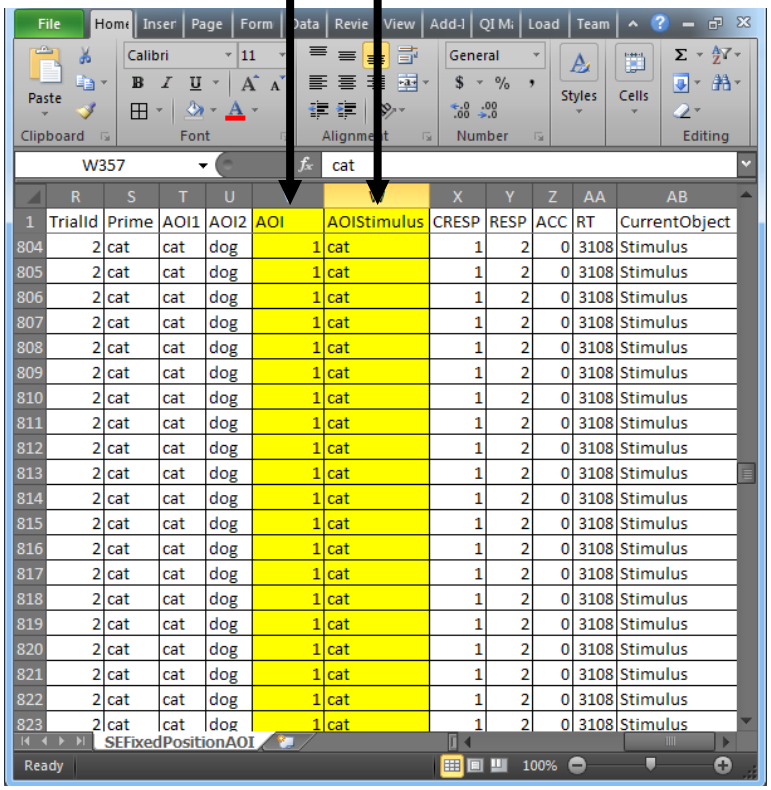
   theUserEyeGazeData.
   AOIStimulus = " "

   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 3: (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 code 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.

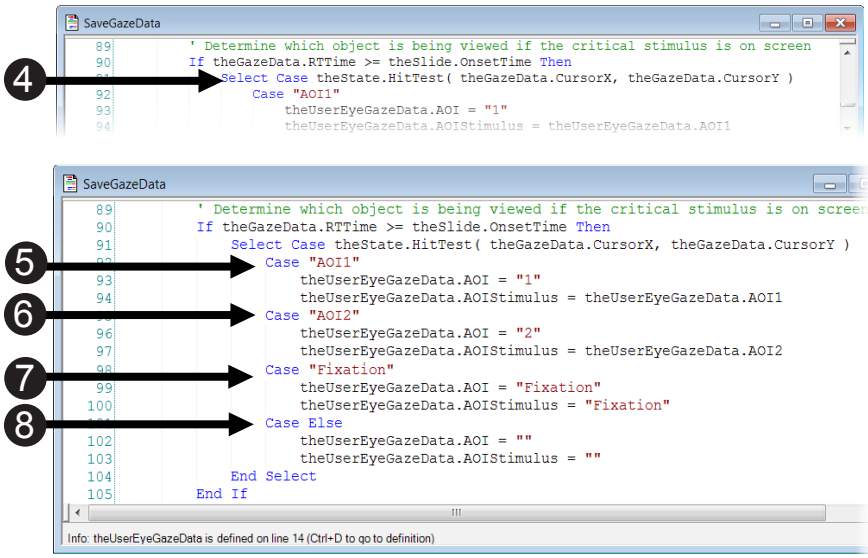
7) **Case "Fixation"** is ***true*** if the **hit test coordinates *correspond*** with the location of the **Stimulus Slide sub-object** Fixation.

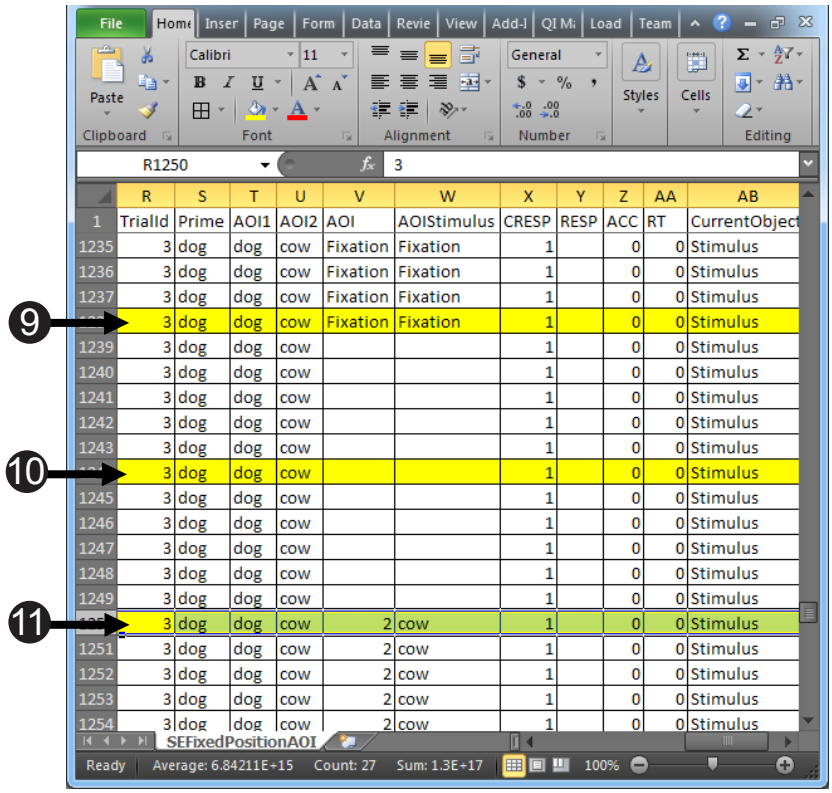
8) **Case "Else"** is ***true*** if the **hit test coordinates *correspond*** to a location on the **Stimulus Slide** that does not have a **sub-object** associated with it.

9) Example of when Case = "Fixation"

10) Example of when Case = "Else"

11) Example of when Case = "AOI2"





Task 4: 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):

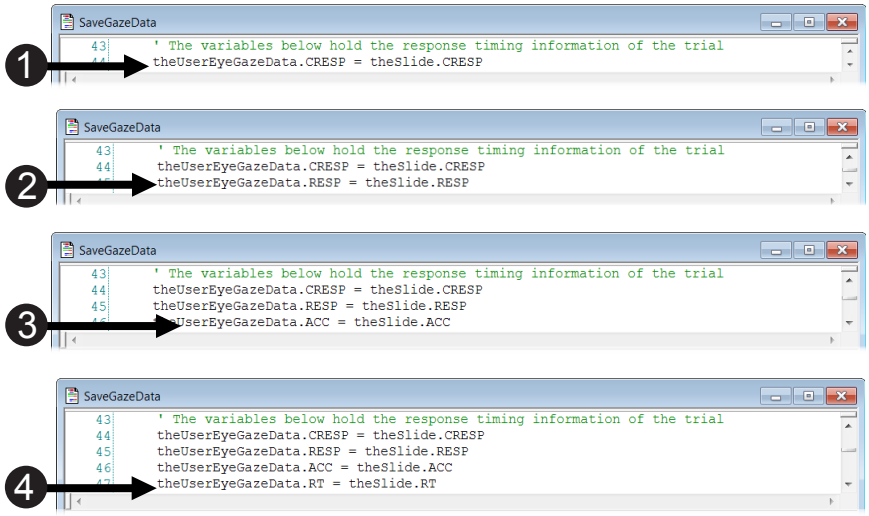
   theUserEyeGazeData.ACC = Stimulus.ACC

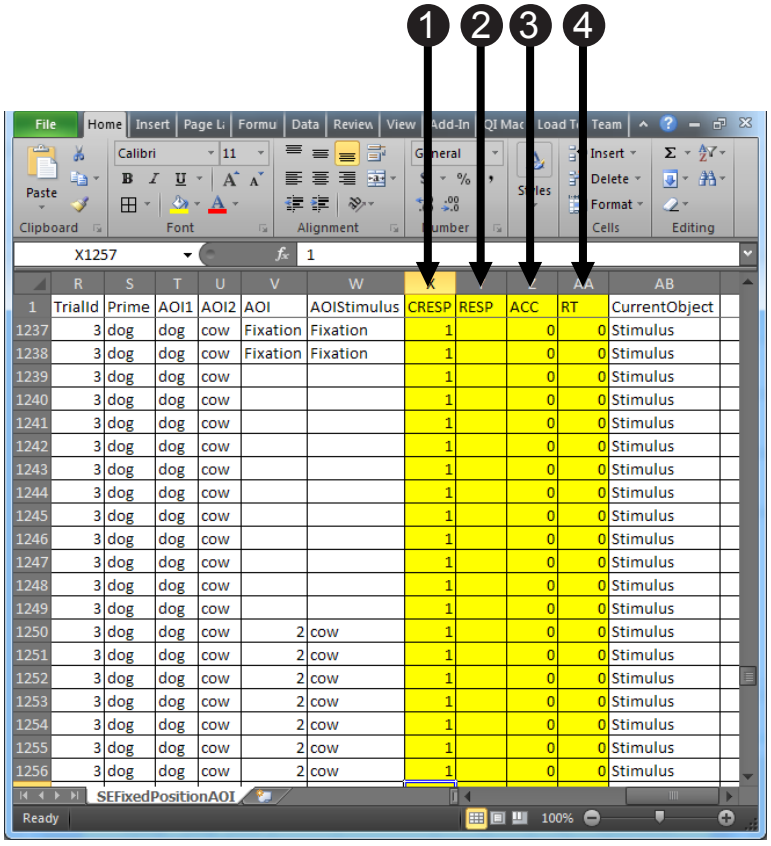
   Purpose: Scores accuracy of response to Stimulus Object for the trial.

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.





	TrialId	Prime	AOI1	AOI2	AOI	AOIStimulus	CRESP	RESP	ACC	RT	CurrentObject
1237	3	dog	dog	cow	Fixation	Fixation	1		0	0	Stimulus
1238	3	dog	dog	cow	Fixation	Fixation	1		0	0	Stimulus
1239	3	dog	dog	cow			1		0	0	Stimulus
1240	3	dog	dog	cow			1		0	0	Stimulus
1241	3	dog	dog	cow			1		0	0	Stimulus
1242	3	dog	dog	cow			1		0	0	Stimulus
1243	3	dog	dog	cow			1		0	0	Stimulus
1244	3	dog	dog	cow			1		0	0	Stimulus
1245	3	dog	dog	cow			1		0	0	Stimulus
1246	3	dog	dog	cow			1		0	0	Stimulus
1247	3	dog	dog	cow			1		0	0	Stimulus
1248	3	dog	dog	cow			1		0	0	Stimulus
1249	3	dog	dog	cow			1		0	0	Stimulus
1250	3	dog	dog	cow	2	cow	1		0	0	Stimulus
1251	3	dog	dog	cow	2	cow	1		0	0	Stimulus
1252	3	dog	dog	cow	2	cow	1		0	0	Stimulus
1253	3	dog	dog	cow	2	cow	1		0	0	Stimulus
1254	3	dog	dog	cow	2	cow	1		0	0	Stimulus
1255	3	dog	dog	cow	2	cow	1		0	0	Stimulus
1256	3	dog	dog	cow	2	cow	1		0	0	Stimulus

Task 4: (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 variables, theUserEyeGazeData.AOI and theUserEyeGazeData.AOIStimulus, identify which area of the screen was being fixated upon the when the Stimulus object is on screen. The HitTest method is used to identify which E-Prime AOI is being fixated upon. The AOI property is set to “1” when the gaze data point falls within the first AOI (left of the fixation point), to “2” when the gaze data point falls within the second AOI (right of the fixation point), to “fixation” when within the Fixation AOI, and to “” when it falls outside of these AOIs. The AOIStimulus property is set to the name of the animal displayed in the left AOI or to the name of the animal displayed in the right AOI if the participant is looking at one of these AOIs, to “Fixation” when looking at the fixation AOI, and to “” if looking elsewhere.

5) ***Recall*** that the CurrentObject being displayed for the current eye gaze sample is assigned from script on the SaveGazeData InLine, for details,see **Task 10: Log the Objects on Screen Via .OnsetTime, see Page 65**.

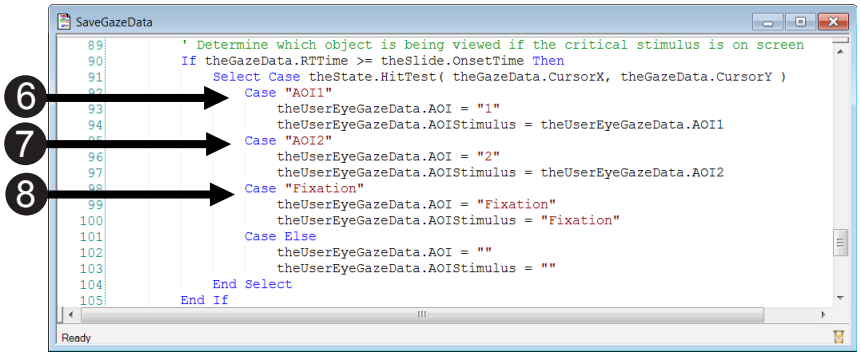
6) If...Case statement used to determine if the participant is looking at **AOI1**. If the statement is **true**, then ***set*** the **AOI** and **AOIStimulus** properties to the **first AOI**.

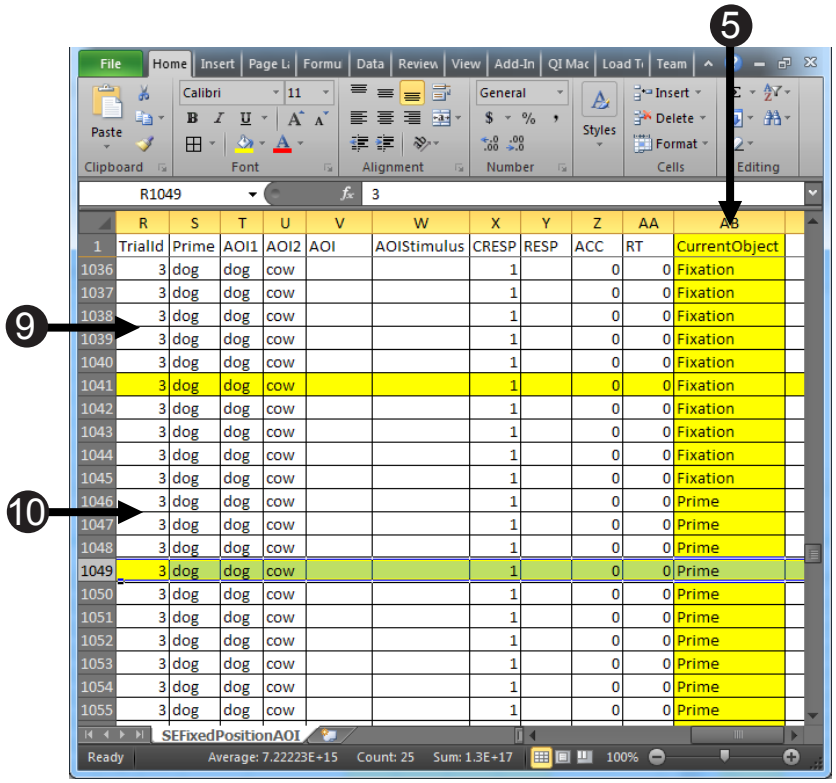
7) If...Case statement used to determine if the participant is looking at **AOI2**. If the statement is **true**, then ***set*** the **AOI** and **AOIStimulus** properties to the **second AOI**.

8) If...Case statement used to determine if the participant is looking at the **fixation AOI**. If the statement is **true**, then ***set*** the **AOI** and **AOIStimulus** properties to the **fixation AOI**.

9) Example when the **CurrentObject** is **Fixation**. Note that the AOI and AOIStimulus columns are blank, since the CurrentObject is not the Stimulus object.

10) Example when the **CurrentObject** is **Prime**. Note that the AOI and AOIStimulus columns are blank, since the CurrentObject is not the Stimulus object.





Task 5: Run the Experiment

Run the experiment to verify that the SMI eye tracker device 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, **Task 16: Add the SMIClose PackageCall to Close the SMI device, Page 46-47)** 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**.

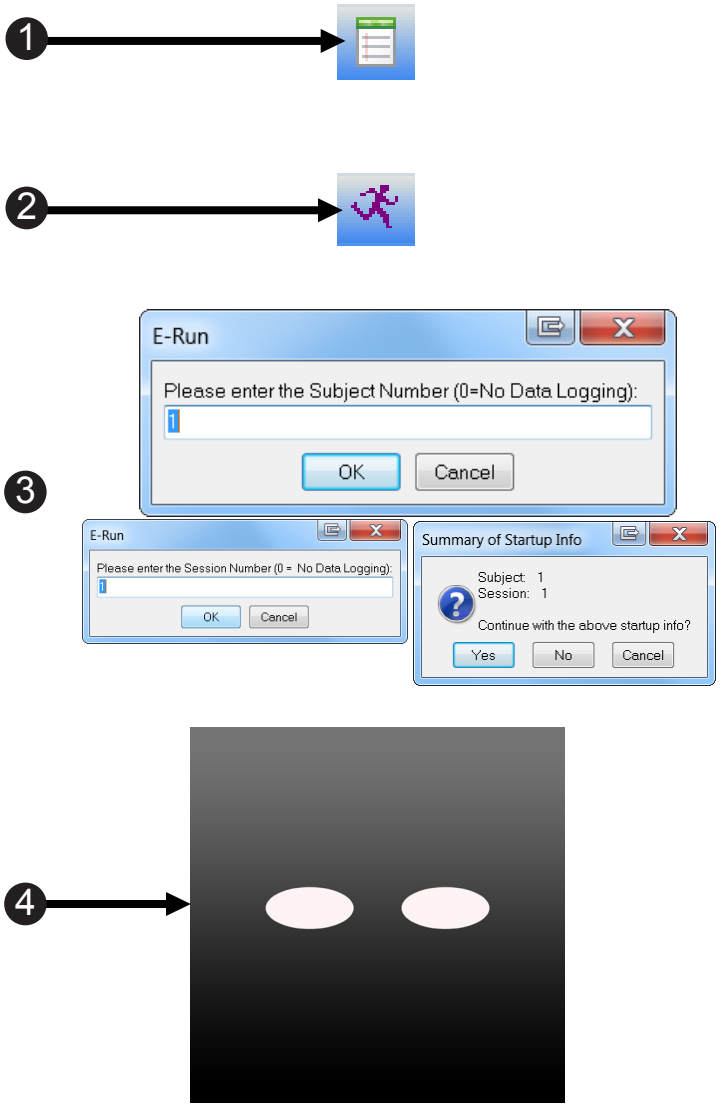
2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

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

4) ***Look*** at the **screen** to ***verify*** the **eyes** are **stable** in the track status window. ***Ensure*** that **both eyes** are represented in the window and that **no arrows** appear. ***Press* Space** to close the track status window.

5) ***Perform* Calibration** (5 dots) and **Validation** (4 dots). ***Accept*** the **Calibration** results that are displayed, if x and y are each <=2 deg; otherwise, ***select*** the **"recalibrate"** option.

6) ***Perform*** the *three trials* in the experiment.



Tutorial 4: **Adding Markers for BeGaze**

Summary:

Tutorials 1 through 3 focus on the SMI package file routines that support the capture and replay of gaze data at the conclusion of each trial, along with the creation of the .gazedata file which can be examined in Excel after the completion of the experiment. To review, when working with the gazedata-related calls, 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 SMI, 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 SMI 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 4 introduces a new area of functionality that is supported in the SMI package file for users who need to utilize SMI's BeGaze data analysis package to perform more visually sophisticated eye tracking data analysis than can be performed with the .gazedata file. Specifically, E-Prime's Task Events feature will be utilized to send markers to the eye tracking data stream. These markers are incorporated into an SMI .idf file, which can then be used to construct a variety of data analyses within BeGaze.

The Task Events feature in E-Prime 2.0 Professional is a powerful tool, which enables one or more 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 in this tutorial to send a marker with the SendImageMessage task action; this marker flags a specific point in the eye tracking data stream, namely when the display object beings to execute. This marker can be then be viewed within BeGaze to construct more meaningful and complex eye tracking data analysis.

The marker that is sent with SendImageMessage can be either a text string to denote a specific event, such as "Fixation" to mark when the Fixation display occurs, or the name of an image file which can illustrate the current display. The E-Prime marker appears in the list of BeGaze User Events. These markers also appear as Bookmarks, which illustrate the key events when the eye gaze is replayed with the BeGaze application. This tutorial illustrates sending both text and image file markers with Task Events, and introduces how to access this information within BeGaze.

Goal:

By the end of this tutorial, you will have created a basic EESMI-enabled E-Prime paradigm that uses Task Events to send event markers to the .idf file for enhanced analysis in BeGaze. This tutorial sends four different markers for each trial, to flag when the following events occur: StartTracking begins, Fixation is displayed, Prime is displayed, and the Target image is displayed. In practice, you may wish to flag different events. However, the techniques shown in the tutorial can be applied to any of the SMI PackageCalls or display objects, so you can easily apply Task Events to your own paradigm.

Tutorial 4: **Adding Markers for BeGaze**

Overview of Tasks:

- Load SMIFixedPositionAOITutorial2Completed and resave it as MySMIFixedPositionAOIMarkersTutorial4Completed.es2.
- Add the Task Events to mark when tracking begins, when the Fixation and Prime displays appear, and when the Stimulus display appears.
- Verify the overall experiment structure and run the experiment.
- Load the .idf file into BeGaze and construct an AOI analysis.

Estimated Time: 15-20 minutes

Task 1: Open the SMIFixedPositionAOITutorial2Completed.es2 Experiment and save to a new location

Locate the E-Studio icon in the Start>All Programs>E-Prime 2.0 menu and launch the application by selecting it. Load the SMIFixedPositionAOITutorial2Completed.es2 experiment.

This tutorial begins with the E-Prime experiment that is created in Tutorial 2. Open the E-Studio application, navigate to the "…My Experiments\SMI\Tutorials\SMIFixedPositionAOI" folder, load the completed tutorial, and save the experiment file to the "…My Experiments\SMI\Tutorials\SMIFixedPositionAOIMarkers" subfolder.

⚠ **NOTE:** *The .es2 file being saved to a different folder.*

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.

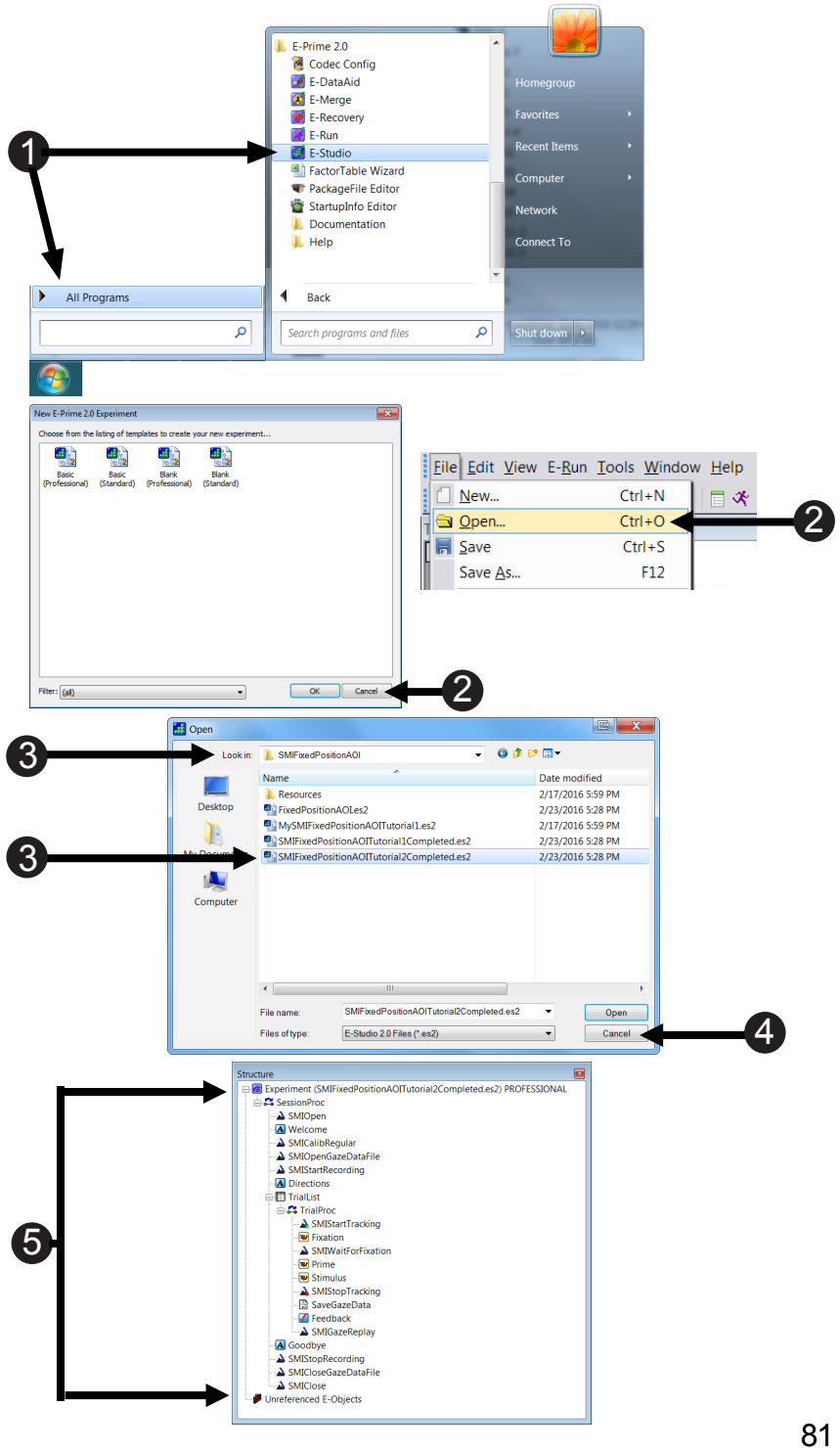
2) ***Click*** the **Cancel** button. ***Select*** **File** > **Open**.

3) ***Navigate*** to the "**…\My Experiments\SMI\Tutorials\ SMIFixedPositionAOI\"** folder.

4) ***Select*** the "**SMIFixedPositionAOITutorial2 Completed.es2**" file and load the paradigm into E-Studio.
If you cannot find the 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.

6) ***Save*** the experiment with a new name in a **new** folder: "**…My Experiments\SMI\Tutorial\ SMIFixedPositionAOIMarkers**" subfolder, and enter "**MySMIFixed PositionAOIMarkersTutorial4 Completed.es2**" in the File name field.



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

Add an Attribute named "Target" to the TrialList.

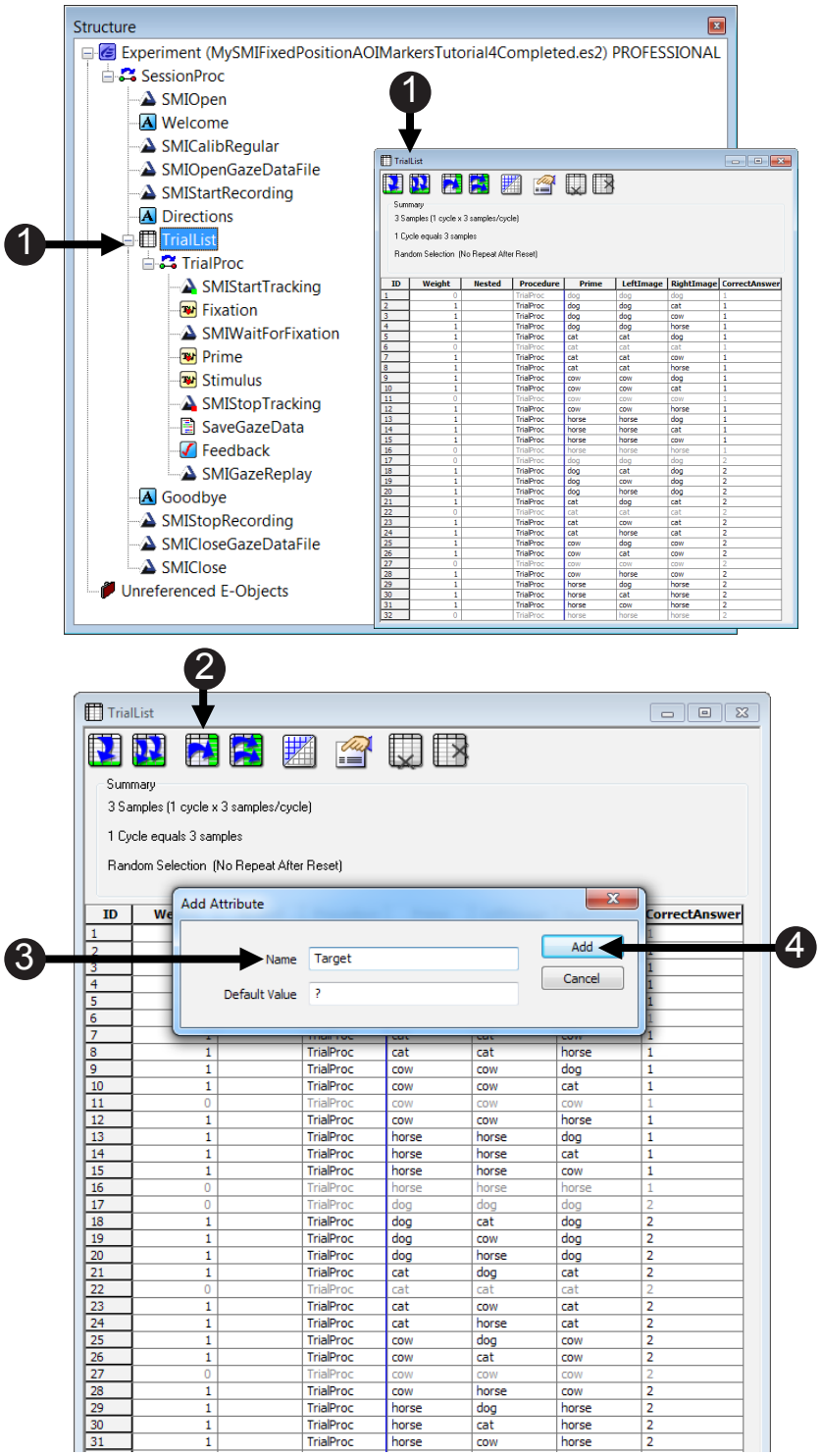
Currently, the SMIFixedPositionAOIMarkers.es2 experiment does not explicitly encode the AOI for each trial (left or right). This information would be useful to have when analyzing the gaze data in BeGaze. In order to send this information to BeGaze, it must first be added to the experiment. In this step, 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 BeGaze.

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.

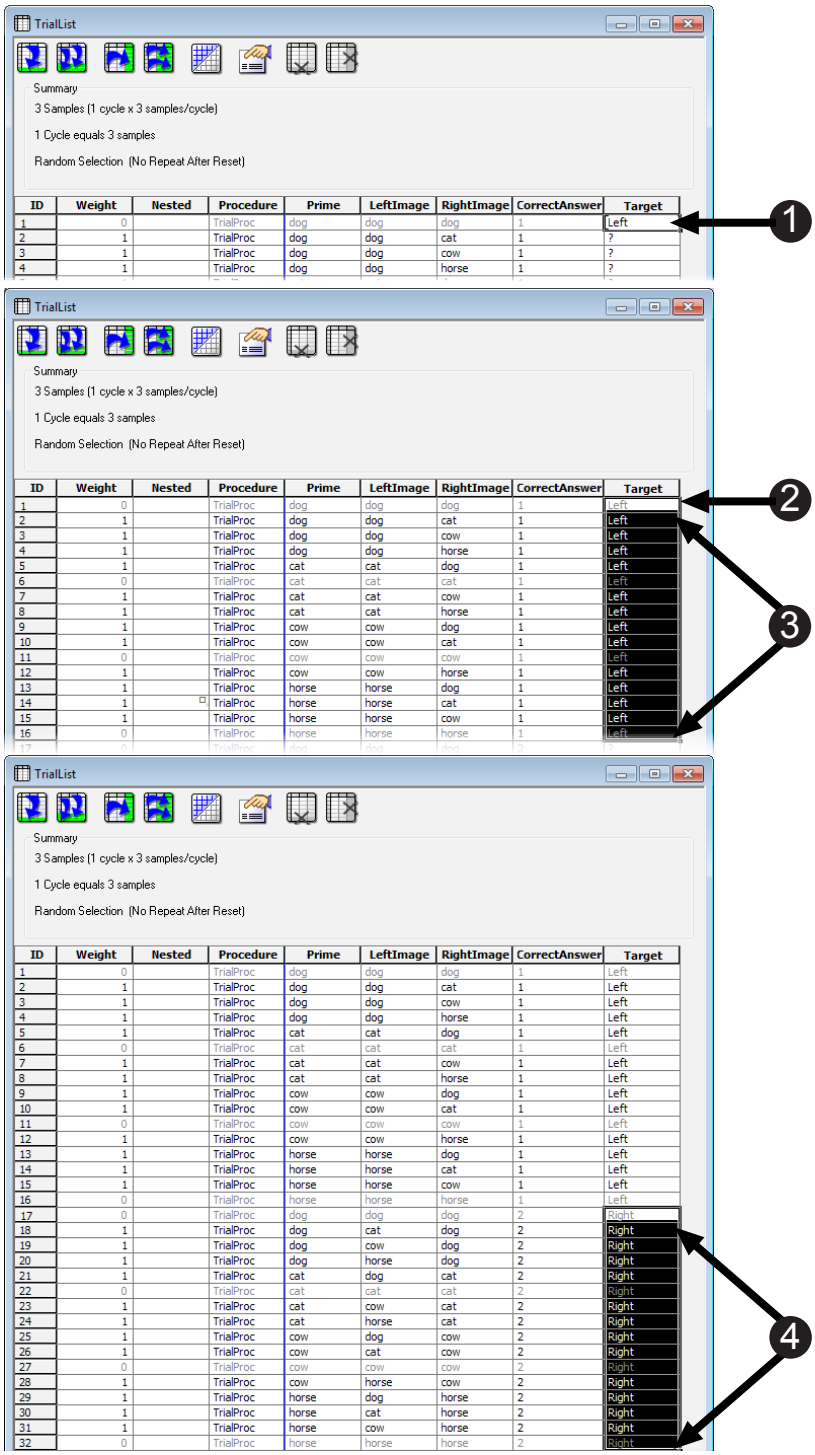
The Target Attribute will be used in BeGaze to identify where the target image was displayed relative to the fixation point. The default value assigned to the Target Attribute in Task 3 ("?") is changed in this task to either "Left" or "Right". Although the CorrectAnswer attribute could be used to infer 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 BeGaze ("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 4: Review the default EventMessage on the StartTracking PackageCall

Examine the StartTracking PackageCall to confirm that it sends a StartTracking.bmp message by default, which increments the trial counter in the .idf file.

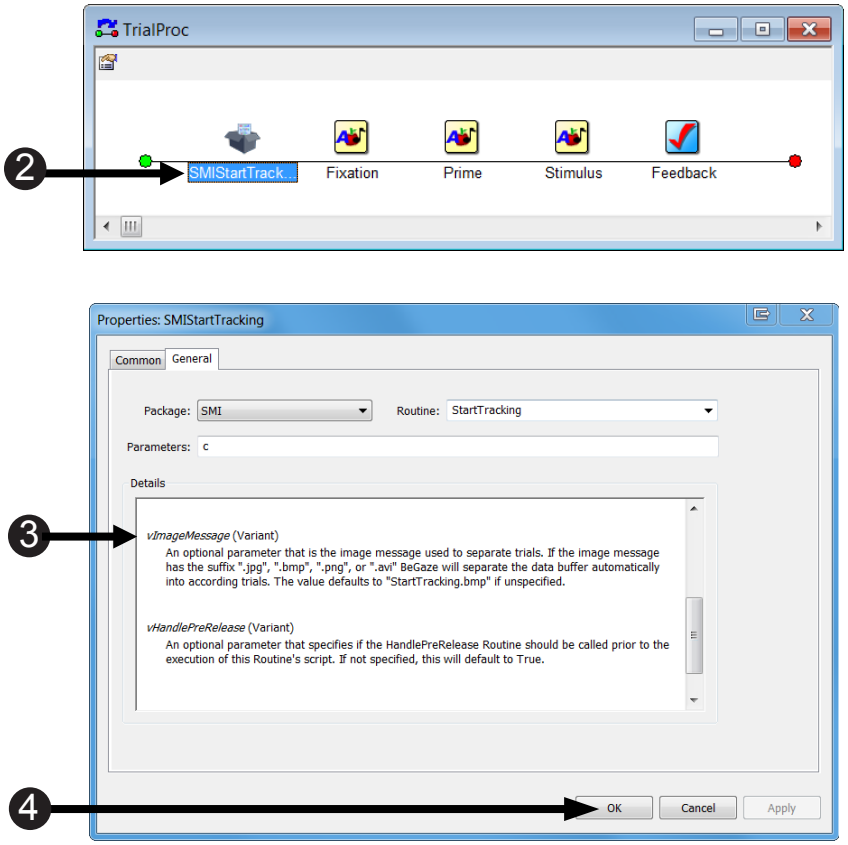
The first event in the trial sequence for which we want to send a marker is when eye tracking begins for the trial. This occurs with the StartTracking PackageCall. The StartTracking PackageCall sends an Event marker automatically. The default message that is sent is the “StartTracking.bmp” message, which does not need to be changed. Note that we are specifying a graphics file name that does not exist. If the file does not exist, then it obviously cannot be displayed within BeGaze; however, the user event will still appear as a bookmark and will still denote the event. (You can create this image later for use in BeGaze for visualization purposes.) In addition to generating the user event, however, including a graphics file extension (.bmp in this case) as part of the marker name also results in the trial counter being incremented in BeGaze, which is key to identifying the start of each E-Prime trial.

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

2) ***Double click*** the **SMIStartTracking** PackageCall on the **TrialProc** to display its **Property Pages**.

3) ***Scroll down*** through the **Details** section until the **vImageMessage** parameter is visible, and ***read*** the text description.

4) ***Click*** the **OK** button.



Task 5: 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.

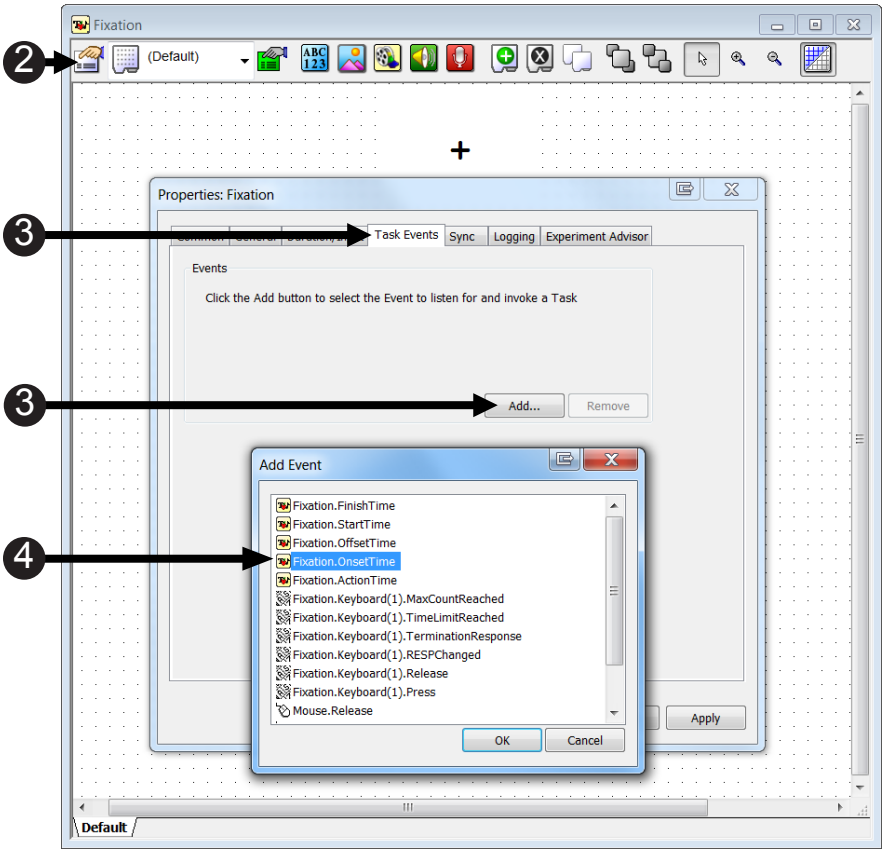
Most markers are not sent to the .idf files via setting parameters on package file call, as is done with the StartTracking PackageCall, but instead are defined on the Task Events tab of a Display object. The first object that we will create a Task Event for in this manner is the Fixation object. The Fixation Object's Onset Time, or the time when the Fixation object beings its action of being displayed, is used as the triggering event. For this Task Event, we always send the text string "Fixation"; there are no experiment attributes that vary from trial to trial with respect to the fixation, so sending a fixed string is sufficient to mark this event in the trial sequence. As you will see later, we can send references to Attributes on the List object as messages also.

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 6: Edit the Fixation.StartOnsetTime Task Event

Define the "Task" component of the Task Event by specifying "Fixation" as the marker to be sent.

The next task is to configure the Task Event to send a "Fixation" marker to the .idf file. We want to send a string that identifies when the Fixation object begins to display.

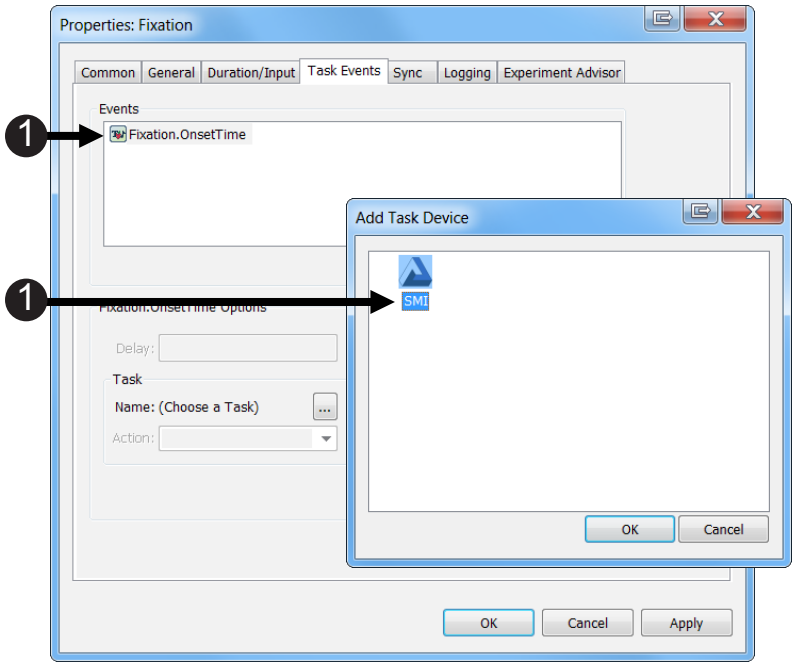
1) ***Select*** the **Fixation.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **SMI** icon.

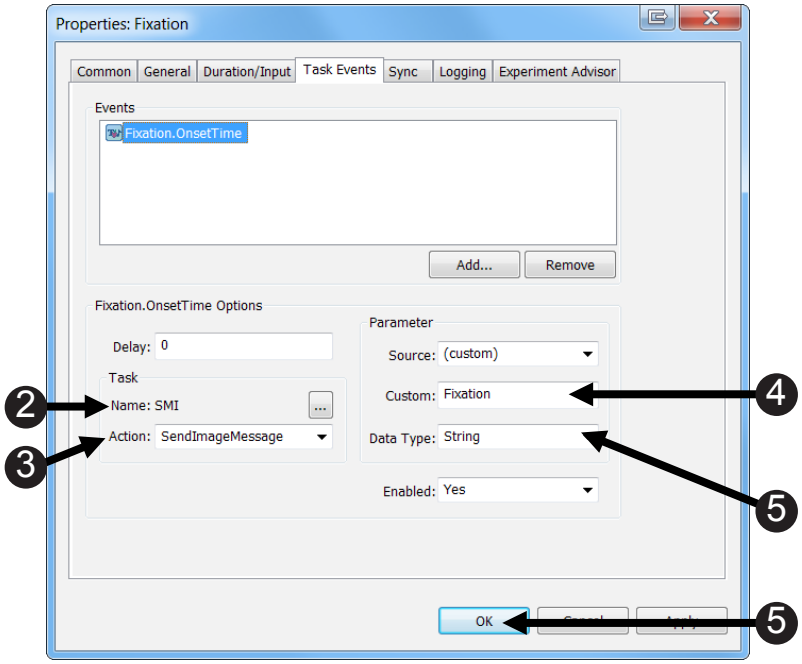
2) ***Note*** the **Name** field ***reads*** **SMI**, indicating that the **SMI device** will ***receive*** the **Task information**.

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

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

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





Task 7: Add a Task Event to The Prime Object

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

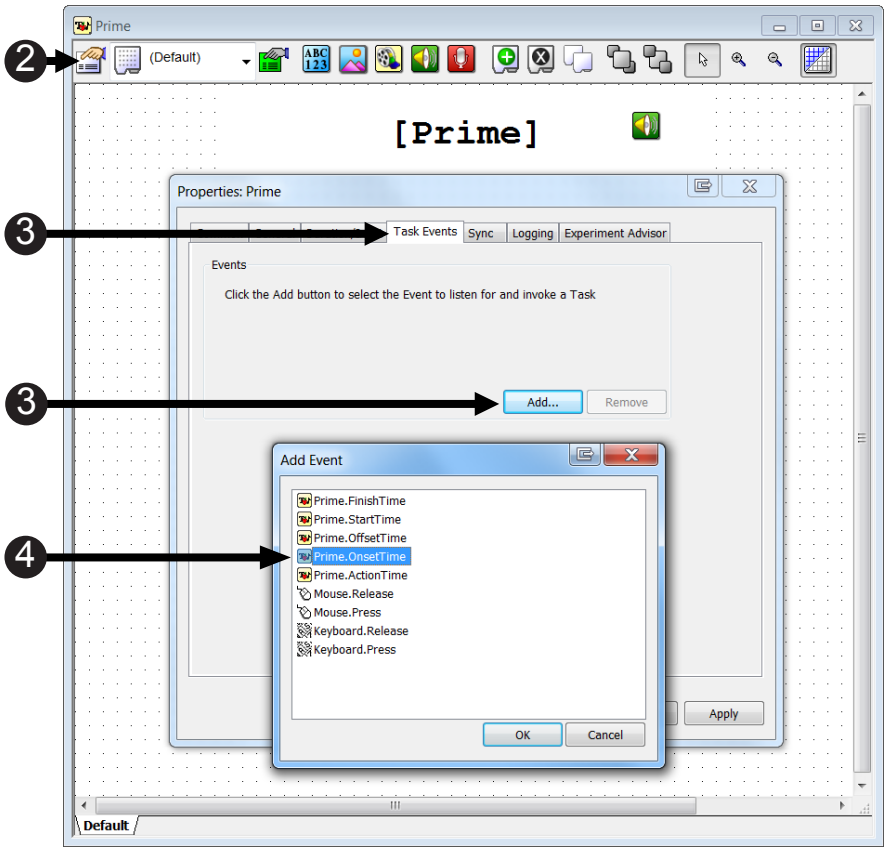
This step adds an .OnsetTime task event to The Prime Object. This task event will send the onset time of the Prime Object from E-Prime to the .idf file.

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 8: Edit the Prime Task Event

Define the "Task" component of the Task Event by specifying the Prime word as the marker to be sent.

The next step is to edit the Prime.OnsetTime Task Event so E-Prime knows what information to send to the .idf file. 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 User Events that are labeled "dog", "cat", "cow", or "horse".

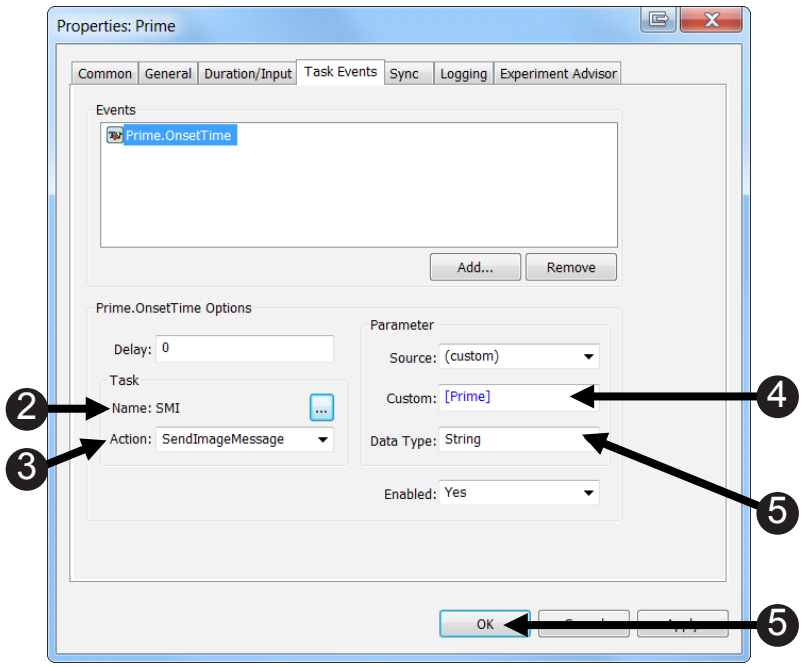
1) ***Select*** the second **Prime.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **SMI** icon.

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

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

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

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



Task 9: Add a Task Event to the Stimulus object

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

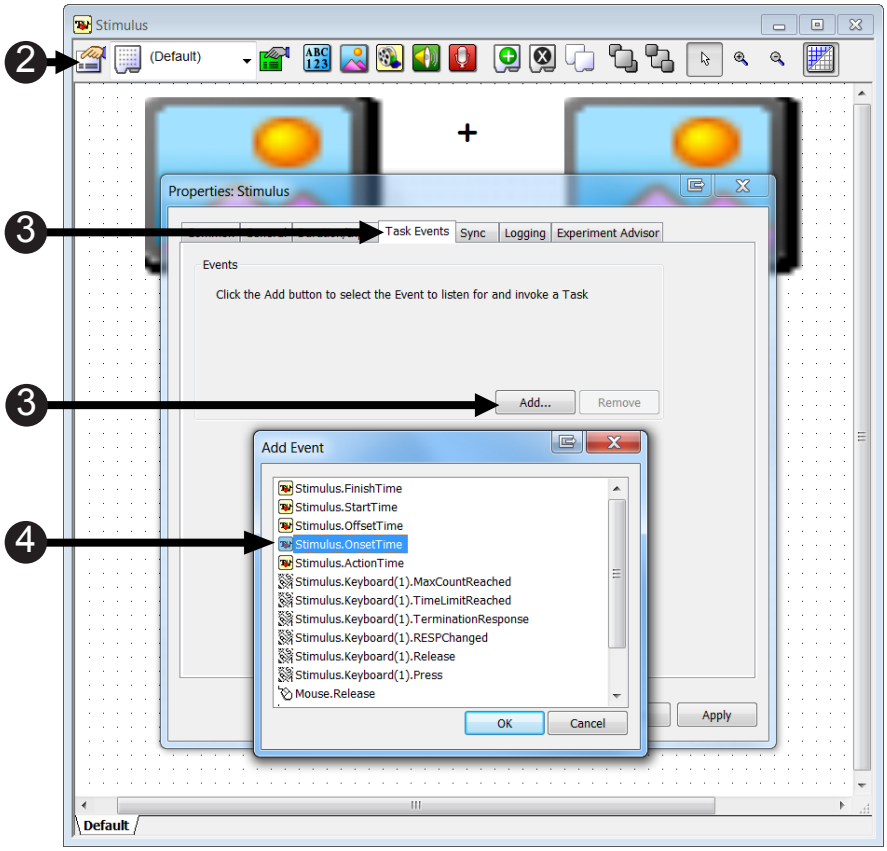
This step adds an .OnsetTime task event to the Stimulus object. This task event will be triggered at the onset time of the Prime Object.

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 10: Edit the Stimulus Task Event

Define the "Task" component of the Task Event by specifying where the target appears as the marker to be sent.

The next step is to edit the Stimulus.OnsetTime Task Event so E-Prime knows what information to send to the .idf file. In this we want to identify where the target animal appears on the trial. The target animal identifies where the animal who matches the animal sound played when the Prime is displayed appears. Recall that the Target attribute Recall that the Prime attribute is assigned to the Prime word that appears on each trial. Therefore, sending the [Target].bmp attribute will result in sending a "left. bmp" or "right.bmp" marker.

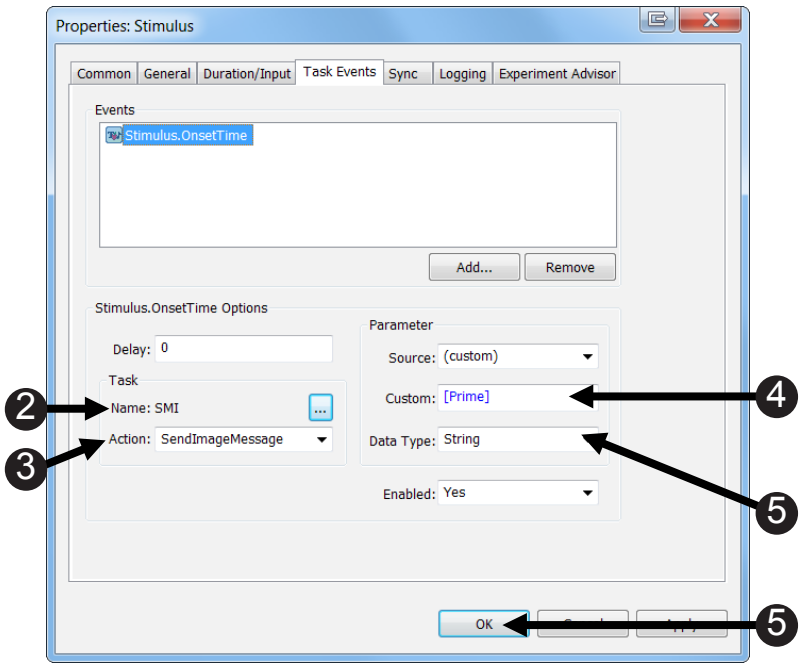
1) ***Select*** the **Stimulus.OnsetTime** task event. ***Click*** the **...** button next to the **Name** field. ***Double click*** the **SMI** icon.

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

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

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

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



Task 11: Run the experiment

Run the experiment to verify that the SMI eye tracker device is working and to generate an .idf file.

Now you should run the experiment to generate an .idf file. The next section of this tutorial provides an overview of how to utilize the markers that have been inserted into the .idf file when analyzing the data in BeGaze.

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**.

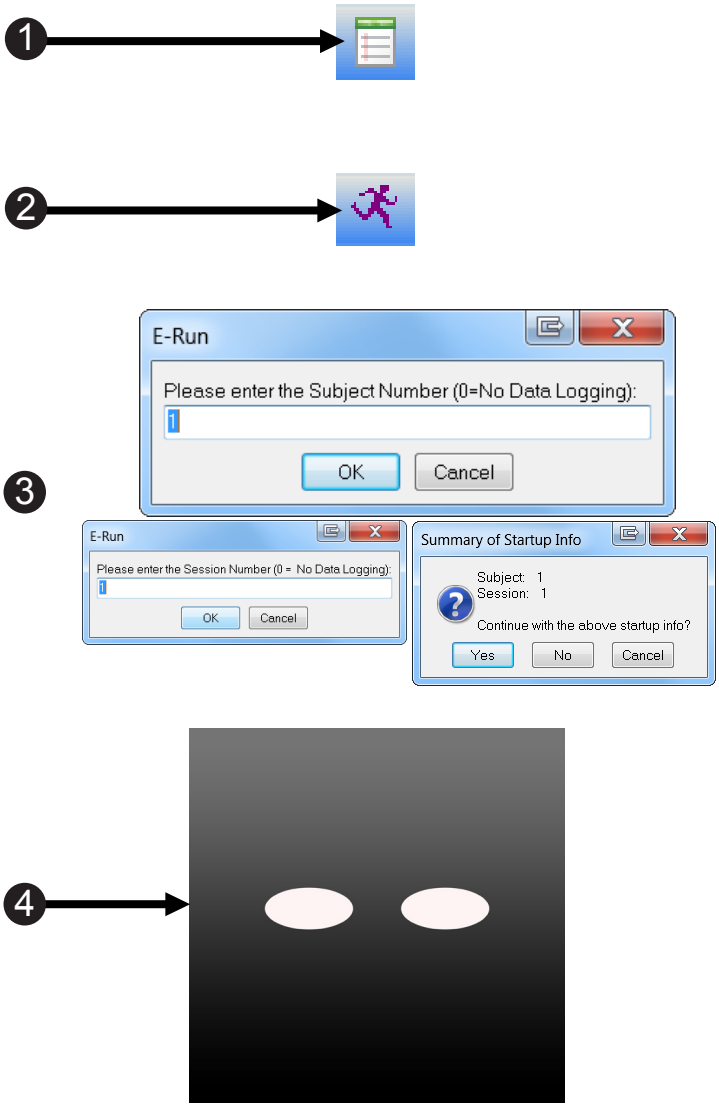
2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

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 **both eyes** are represented in the window and that **no arrows** appear. ***Press* Space** to close the track status window.

5) ***Perform* Calibration** (5 dots) and **Validation** (4 dots). ***Accept*** the **Calibration** results that are displayed, if x and y are each <=2 deg; otherwise, ***select*** the **"recalibrate"** option.

6) ***Perform*** the *three trials* in the experiment.



Analysis of E-Prime markers in BeGaze

This section provides an overview of the steps needed to perform a meaningful analysis on the data that is created from the completed Tutorial 4. Unlike the previous sections of this Tutorial, we do not present a step-by-step guide to using BeGaze. Instead, this section is intended to provide a high-level overview of analyzing gaze data in BeGaze. 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. For specific information on data analysis in BeGaze, please refer to your SMI documentation.

Step 1: Load the .idf file into BeGaze

Gather all files in the same folder

The location of the .idf file varies depending on your specific SMI eye tracking configuration. For a one-computer setup, where the SMI server is running on the same computer that is running the E-Prime experiment, the .idf file is written to the same folder as the .edat2 file. (A one-computer setup is typical for any of the mobile eye trackers, such as the RED250mobile and the REDn Scientific). For a two- computer setup, where E-Prime is running on one computer and the SMI server is running on another computer, the .idf file is written to one of the following locations on the SMI server computer:

- The current user's Temp subfolder
- The SMI subfolder within Program Files (e.g. …Program Files(x86)\SMI\).

If necessary, gather the .idf file or files to be analyzed along with any graphics files (.bmp), if any, that were referenced in a SendImageMessage Task Event. All of these files should be placed in the same subfolder on the computer to which you installed BeGaze (as part of the Experiment Center installation).

Load .idf file into BeGaze

Open the BeGaze application, select the New Experiment icon (upper left-hand corner), and select the folder that contains the .idf file(s) of interest. BeGaze will automatically load all .idf and graphics files that if finds in the folder.

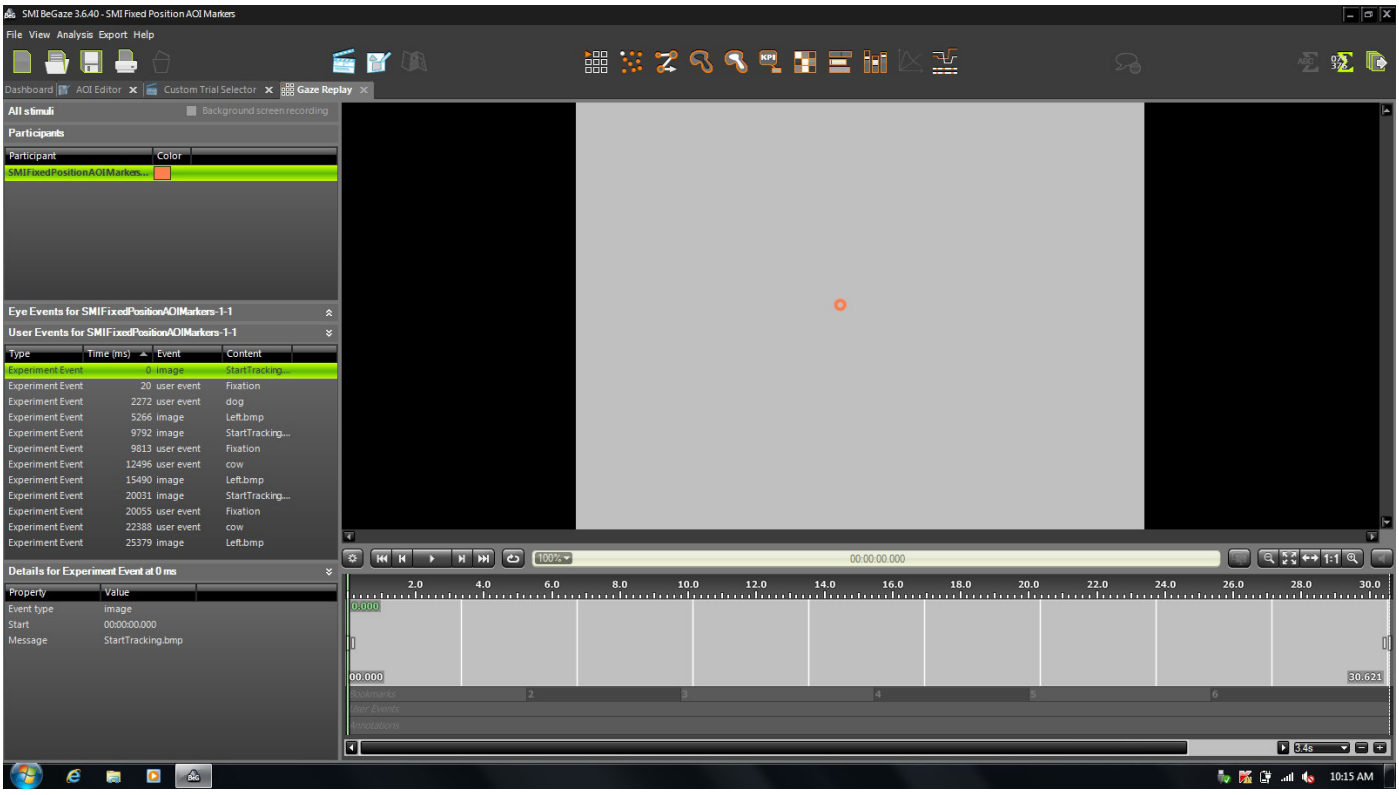
Step 2: Review the markers sent from E-Prime

Click on the Gaze Replay icon located at the top of the screen, and then click on the User Events pane that now appears on the left-hand side of the screen. The User Events pane displays a list of all of the markers that were sent from E-Prime with a SendImageMessage Task Event. You should see the following entries for each trial:

- StartTracking.bmp (recall that this was sent via the StartTracking PackageCall)
- Fixation
- dog, cat, cat, or horse (name of the Prime)
- Left.bmp or Right.bmp (determined by the value of the Target attribute on the trial)

You should see 12 entries in the User Events listing, four entries each for the three trials that were run.

In addition, for any Image Message that was sent as the name of an image file, the Image Message will appear in the Bookmarks section under the gaze data stream. It is important to understand that every time that an event message is received that references a graphics file, (in this example, .bmp) BeGaze creates a new trial. For this experiment, Trial001 starts when StartTracking.bmp is sent. Trial002 beings when Left.bmp/Right.bmp is sent, even though these events are generated from the same "trial" in E-Prime. You should see something similar to the following:



Step 3: Define AOIs

To create AOIs for your BeGaze experiment, do the following:

1) Click on the AOI Editor icon at the top of the screen.
2) Click on the tool that most closely resembles the shape of the AOI that you would like to draw (circle, square, or free hand). For the SMIFixedPositionAOIMarkers experiment, select the square.
3) Draw an AOI in the area of interest, and rename it to something descriptive (such as "LeftStimulus").
4) Repeat steps 2 – 4 for each AOI in the experiment.

After creating the AOIs, you can select the 'run the experiment' option to replay the eye gaze, or select other BeGaze analysis tools to assess more in-depth information about the user's gaze in relation to the AOIs.

See **Appendix D: AOI Analysis in BeGaze, see Page 137** for some examples of the types of analyses that can be constructed.

Tutorial 5: **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 is done with the WaitForFixation PackageCall). The current tutorial further illustrates techniques for using eye gaze data interactively.

During this tutorial, you will modify an EESMI-enabled experiment to process the gazedata information that is being provided actively by the eye tracker during a trial. The paradigm for Tutorial 5 is different from the paradigm used in the previous tutorials. It is the SMIVaryingPositionAOI Tracking experiment. In this experiment, you are shown a target and asked to identify it later. The target is a letter displayed in a specific orientation, e.g. an upside down ‘P’. You are then shown a fixation cross, followed by a randomly generated screen with the target letter displayed in several different orientations. The participant’s task is to identify the original target in the same orientation that was first shown. Once you identify the letter, the participant presses a key.

Because of the many edits required in order to implement this paradigm, we have already programmed it for you in SMIVaryingPositionAOITrackingTutorial5Start.es2. As a result, this tutorial focuses on the reviewing the key aspects of the experiment design, which is to interactively outline the letter that is currently being focused on in a red box.

We recommend that you run the fully programmed sample version of this experiment, found in ...My Experiments\SMI\Samples\SMIVaryingPositionAOITracking\SMIVaryingPositionAOITracking.es2 prior to beginning Task 1 of this tutorial. This will provide you with a clear understanding of the task prior to reviewing its implementation.

Goal:

This tutorial will teach you the concepts behind working interactively with gazedata to modify the stimulus display.

Overview of Tasks:

- Open the file SMIVaryingPositionAOITrackingTutorial5Start.es2 file and save as MySMIVaryingPositionAOITrackingTutorial5Completed.es2.
- Set the Stimulus Slide Object’s Property Pages.
- Confirm and review the necessary variable declarations.
- Review the script, which is used to interactively highlight Areas of Interest (AOIs).
- 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 SMIVaryingPositionAOITracking.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 SMIVaryingPositionAOITrackingTutorial5Start.es2 experiment from the SMI\Tutorials\SMIVaryingPositionAOITracking folder.

Open the E-Studio application, navigate to the appropriate folder, and load the SMIVaryingPositionAOI TrackingTutorial5Start.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.

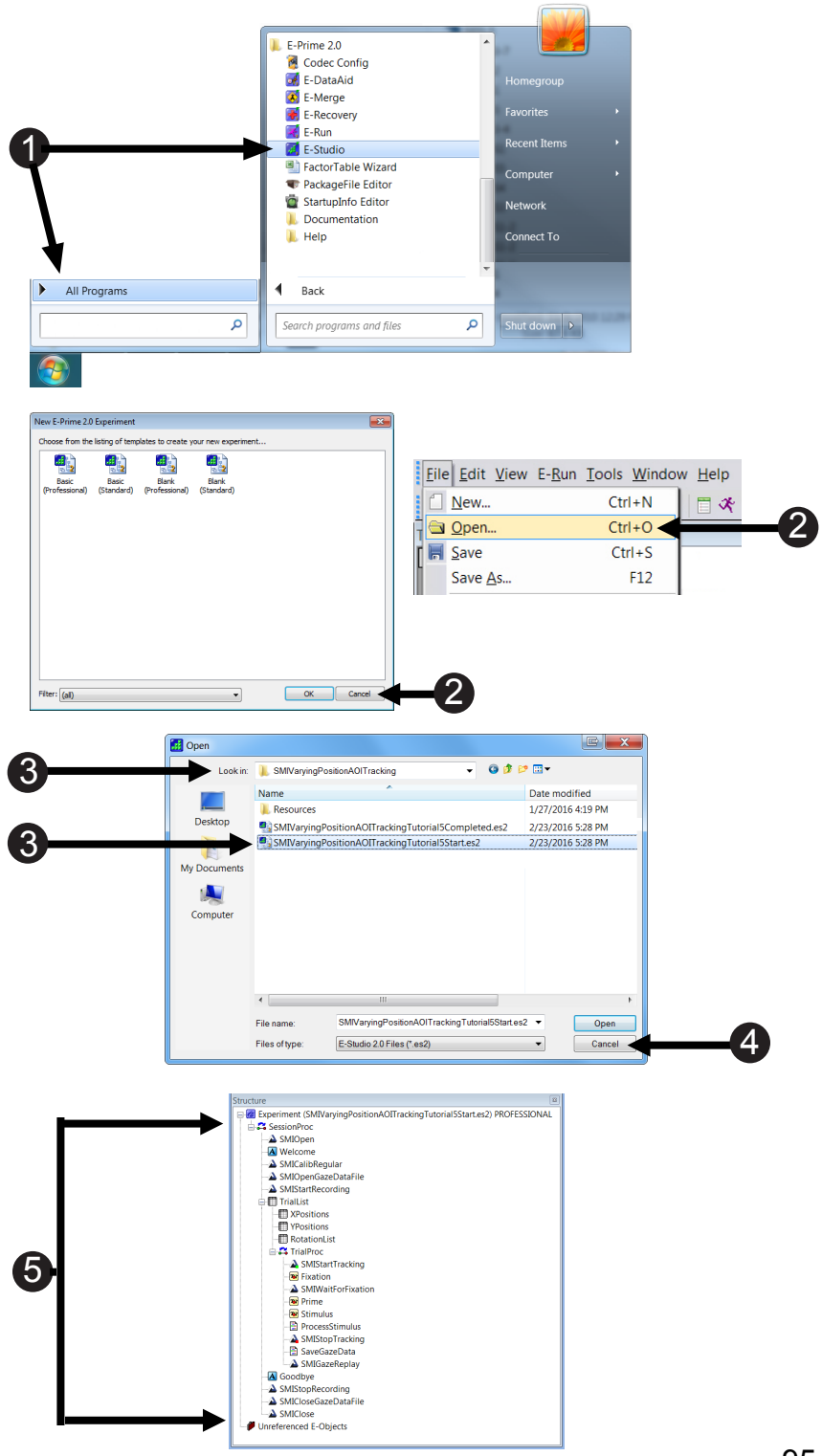
2) ***Click*** the **Cancel** button. ***Select*** **File** > **Open**.

3) ***Navigate*** to the "**…\My Experiments\SMI\Tutorials\ SMIVaryingPositionAOITracking**" folder to load the paradigm.

4) ***Select*** the "**SMIVaryingPositionAOITracking Tutorial5Start.es2**" file and then ***click*** the **Open** button to load the paradigm into E-Studio.
If you cannot find the 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.

6) ***Save*** the experiment as **MySMIVaryingPositionAOI TrackingTutorial5Completed. es2**.



Task 2: Set Stimulus Slide Object Property Pages

Open the Stimulus Slide Object Property Pages and confirm PreRelease and Duration settings.

This experiment tracks the participant's gaze on screen in real time. In order to accomplish this, specific things need to happen in the experiment. The first thing to note is on the Stimulus Slide Object. In order to track the participant gaze in real time, the PreRelease property on the Stimulus Slide Object must be the same value 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 allows time spent waiting for the Stimulus Slide Object to be released and made available for the AOITracking InLine. This gives the AOITracking InLine the ability to alter dynamically 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.

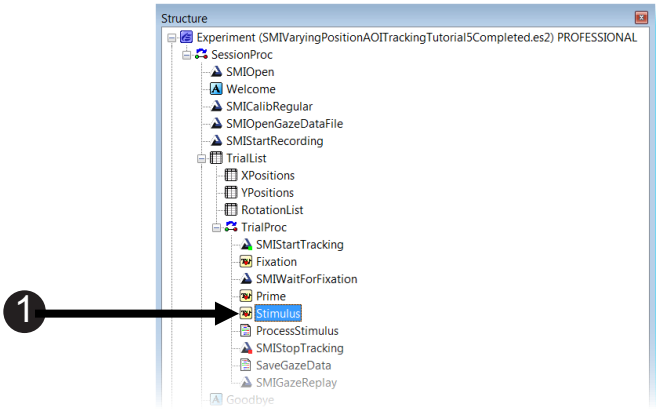
3) ***Select*** the **Duration/Input** tab.

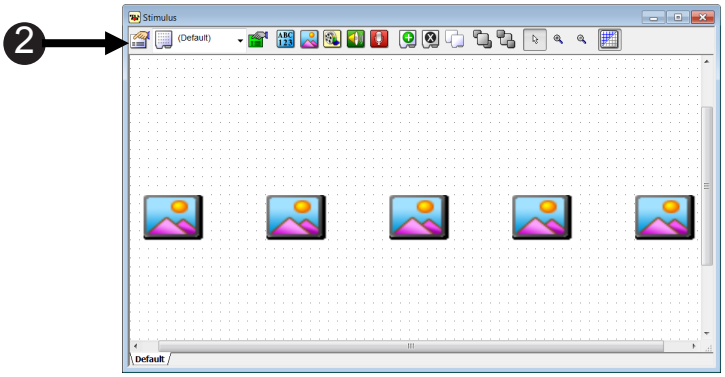
4) ***Confirm*** the **Duration** is set to **5000.**

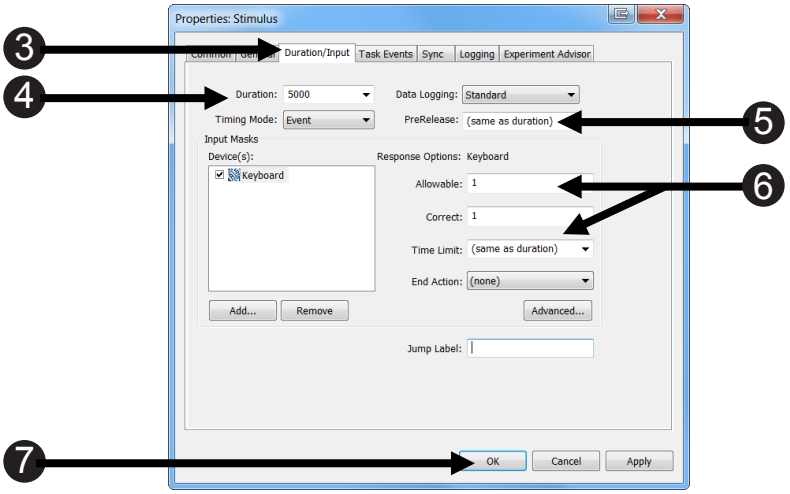
5) ***Confirm*** the **PreRelease** is set to **(same as Duration)**.

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

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







Task 3: Understanding the Variable Declarations

Verify variable declarations.

The script on the ProcessStimulus InLine object deals with a special E-Prime data type used to store individual eye gaze observations as reported by the SMI eye tracker device.

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

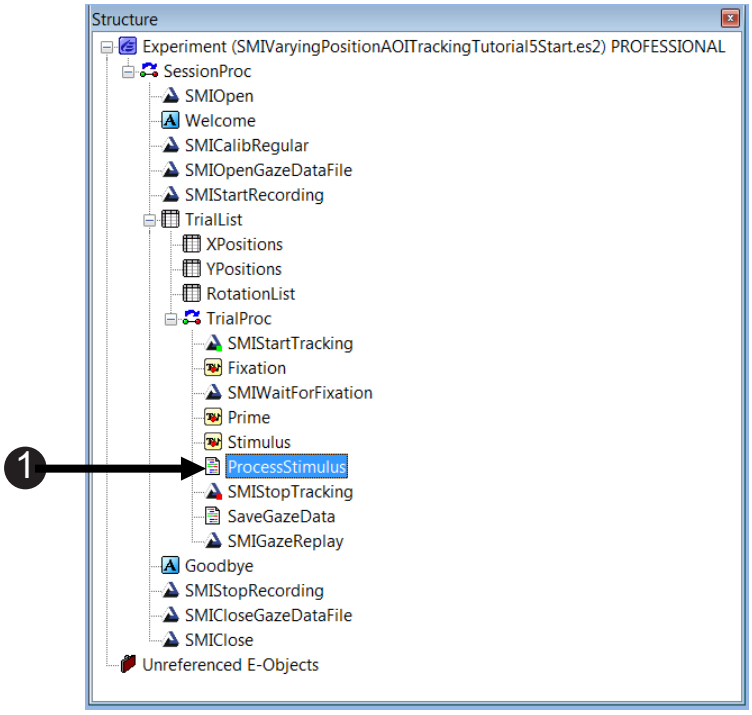
2) ***Confirm*** **Dim theGazeData As SMIResponseData**
Declares variable to store the gaze data sent from SMI.

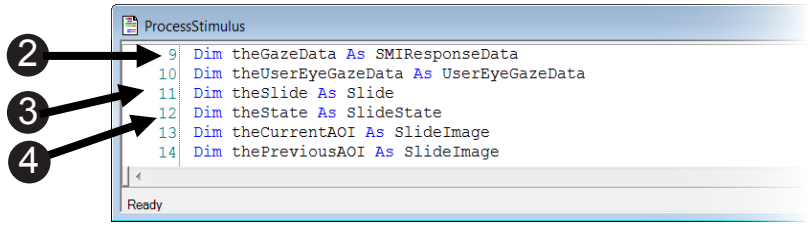
3) ***Confirm*** **Dim theSlide As Slide**
Declares variable for the slide.

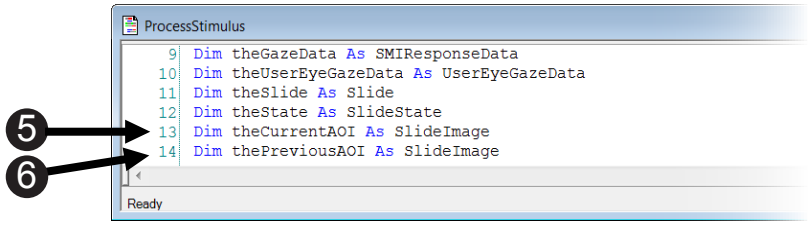
4) ***Confirm*** **Dim theState As SlideState**
Declares variable for the slide state.

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.

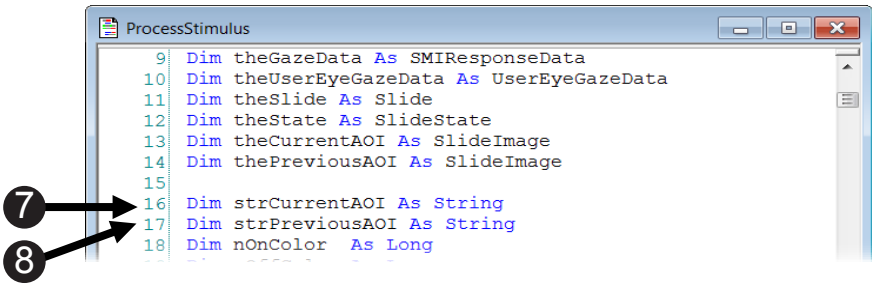
7) ***Confirm*** **Dim strCurrentAOI As String**
This variable will save the current AOI data.

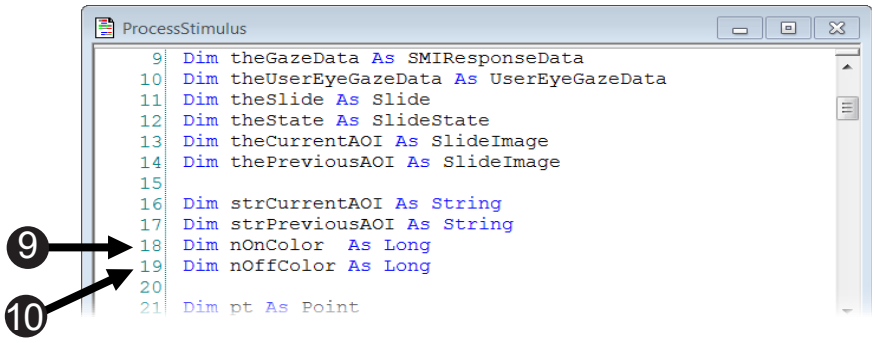
8) ***Confirm*** **Dim strPreviousAOI As String**
This variable will save the previous AOI data.

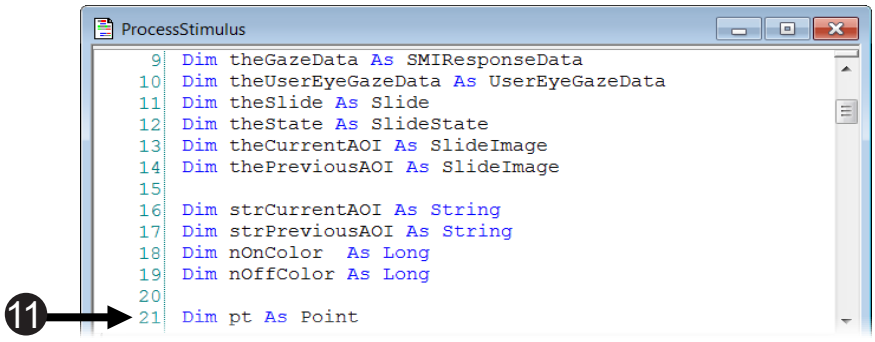
9) ***Confirm*** **Dim nOnColor As Long**
This variable will set the “on“ color for the border.

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

11) ***Confirm*** **Dim pt As Point**
This variable will save the point coordinates.







Task 3 (continued): Understanding the Variable Declarations

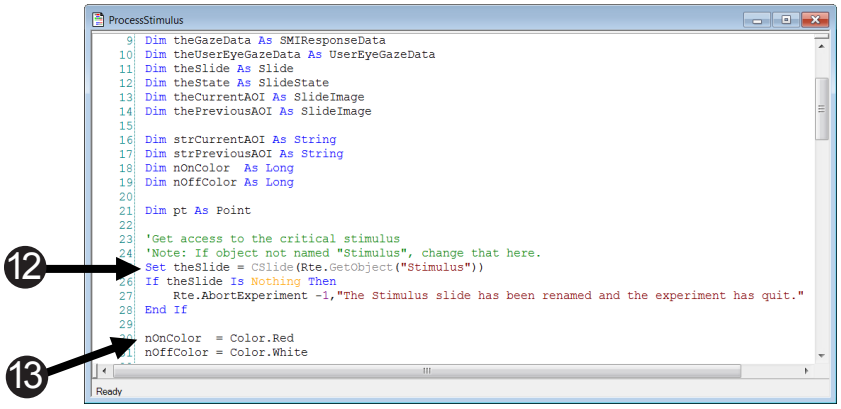
The following steps allow you to confirm the declaration of each of the necessary variables.

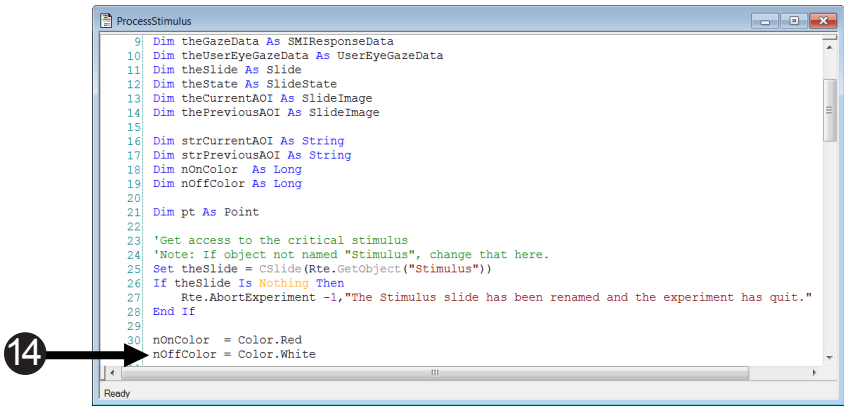
The next set of declarations sets theSlide as Stimulus. It also tells E-Prime what color to make the border while an on or off state is assigned.

12) ***Confirm*** **Set theSlide = Stimulus**

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.





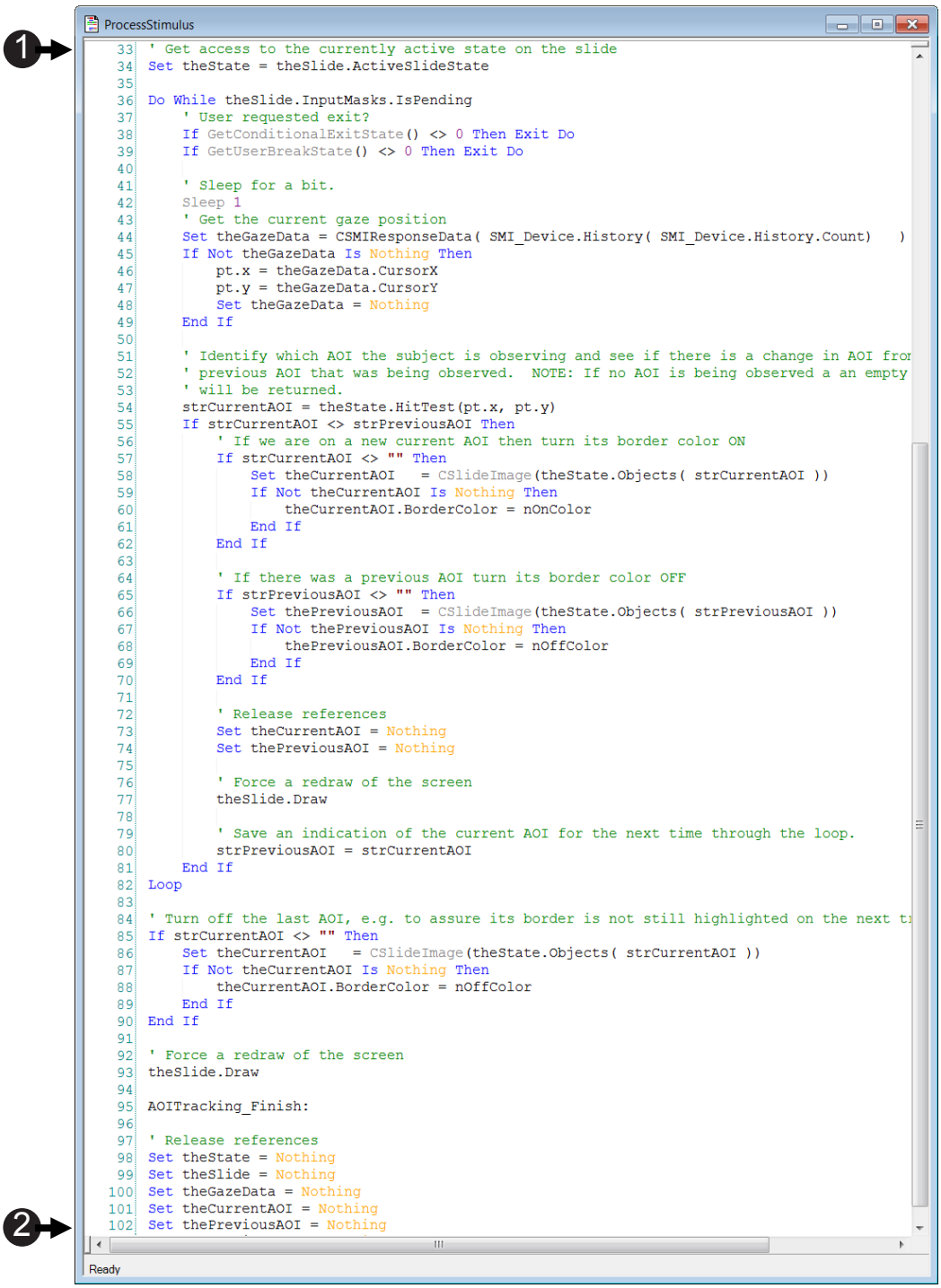
Task 4: The Do…Loop

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) ***Locate*** the beginning of the loop, and read through the comments (in green) and the E-Basic commands.

2) ***Identify*** the end of loop.




```
33  ' Get access to the currently active state on the slide
34  Set theState = theSlide.ActiveSlideState
35
36  Do While theSlide.InputMasks.IsPending
37      ' User requested exit?
38      If GetConditionalExitState() <> 0 Then Exit Do
39      If GetUserBreakState() <> 0 Then Exit Do
40
41      ' Sleep for a bit.
42      Sleep 1
43      ' Get the current gaze position
44      Set theGazeData = CSMIResponseData( SMI_Device.History( SMI_Device.History.Count)   )
45      If Not theGazeData Is Nothing Then
46          pt.x = theGazeData.CursorX
47          pt.y = theGazeData.CursorY
48          Set theGazeData = Nothing
49      End If
50
51      ' Identify which AOI the subject is observing and see if there is a change in AOI fro
52      ' previous AOI that was being observed.  NOTE: If no AOI is being observed a an empty
53      ' will be returned.
54      strCurrentAOI = theState.HitTest(pt.x, pt.y)
55      If strCurrentAOI <> strPreviousAOI Then
56          ' If we are on a new current AOI then turn its border color ON
57          If strCurrentAOI <> "" Then
58              Set theCurrentAOI   = CSlideImage(theState.Objects( strCurrentAOI ))
59              If Not theCurrentAOI Is Nothing Then
60                  theCurrentAOI.BorderColor = nOnColor
61              End If
62          End If
63
64          ' If there was a previous AOI turn its border color OFF
65          If strPreviousAOI <> "" Then
66              Set thePreviousAOI  = CSlideImage(theState.Objects( strPreviousAOI ))
67              If Not thePreviousAOI Is Nothing Then
68                  thePreviousAOI.BorderColor = nOffColor
69              End If
70          End If
71
72          ' Release references
73          Set theCurrentAOI = Nothing
74          Set thePreviousAOI = Nothing
75
76          ' Force a redraw of the screen
77          theSlide.Draw
78
79          ' Save an indication of the current AOI for the next time through the loop.
80          strPreviousAOI = strCurrentAOI
81      End If
82  Loop
83
84  ' Turn off the last AOI, e.g. to assure its border is not still highlighted on the next t
85  If strCurrentAOI <> "" Then
86      Set theCurrentAOI   = CSlideImage(theState.Objects( strCurrentAOI ))
87      If Not theCurrentAOI Is Nothing Then
88          theCurrentAOI.BorderColor = nOffColor
89      End If
90  End If
91
92  ' Force a redraw of the screen
93  theSlide.Draw
94
95  AOITracking_Finish:
96
97  ' Release references
98  Set theState = Nothing
99  Set theSlide = Nothing
100 Set theGazeData = Nothing
101 Set theCurrentAOI = Nothing
102 Set thePreviousAOI = Nothing
```

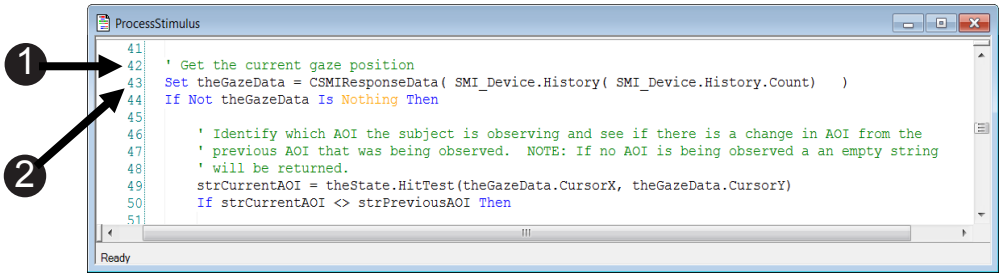
Task 5: Get the current Gaze Data from the Eye Tracker

Review the script labeled 'Get the current gaze position.

In this step, you will review the script that obtains the gaze data, determines the X Y coordinates, and associates the X Y coordinates with the active slide.

1) ***Locate ' Get the current gaze position*** in the **ProcessStimulus InLine** (it will be at the beginning of the **Do…Loop**).

2) **The "*Set* theGazeData ="** line retrieves a copy of the most recent eye tracking gaze data from the eye tracking device.



Task 6: AOI Tracking

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

In the previous step, CursorX and CursorY are set to the coordinates of the participant's current gaze when the current eye gaze data is retrieved. 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** to ***determine*** if the **current gaze point** is on an **AOI**. If so, the name of that **AOI** will be ***assigned*** to **strCurrentAOI**.

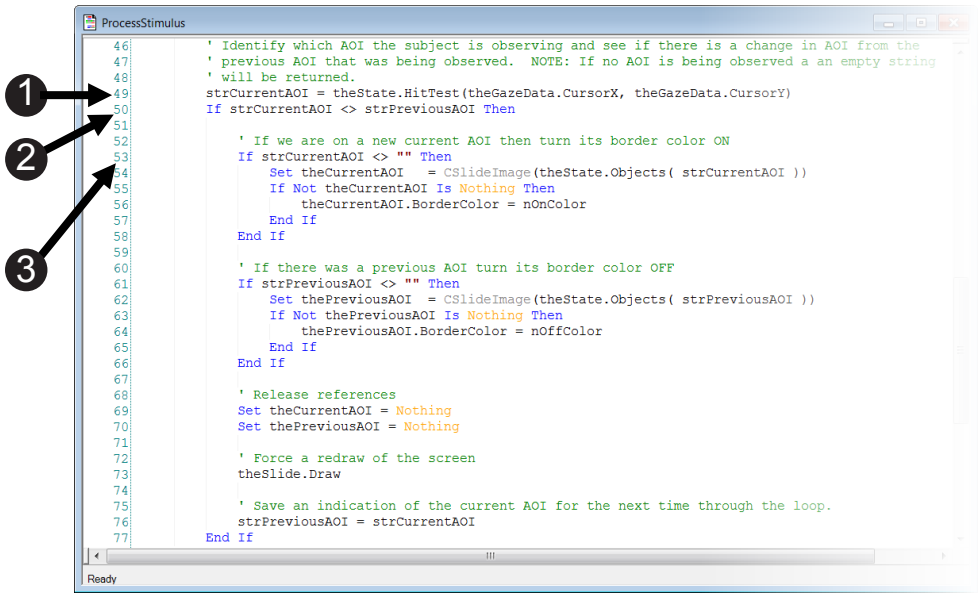
2) ***Use*** a **conditional statement** to determine if **strCurrentAOI= strPreviousAOI.**

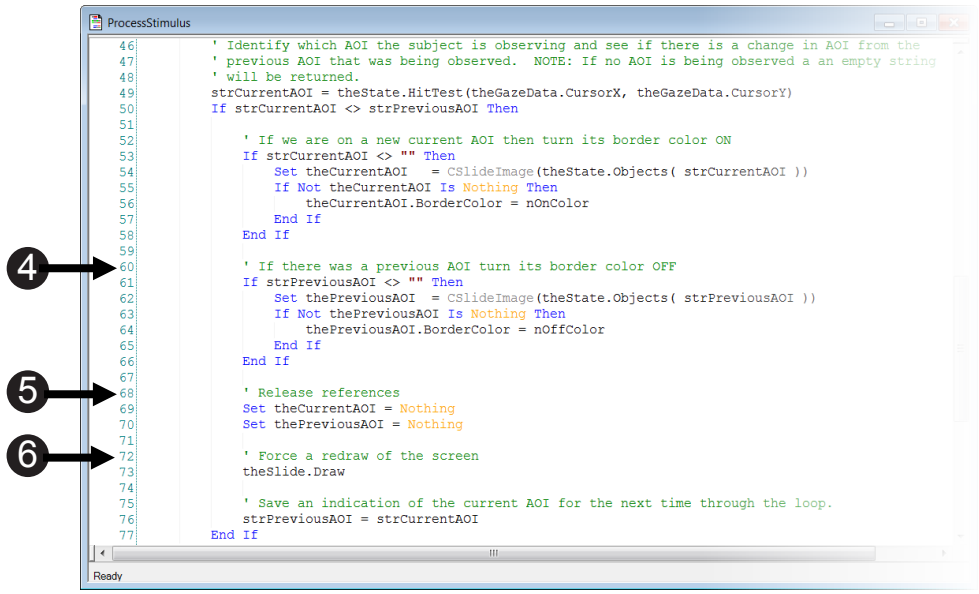
3) ***Use*** a **conditional statement** to ***switch*** **ON** the **BorderColor** of the **AOI** that is *being viewed.*

4) ***Use*** a **conditional statement** to ***switch*** **OFF** the **BorderColor** of the **AOI** that is *not longer being viewed.*

5) ***Release*** **theCurrentAOI** and **thePreviousAOI** references.

6) ***Redraw*** the **screen.**





Task 7: Set the Values for the Next Trial

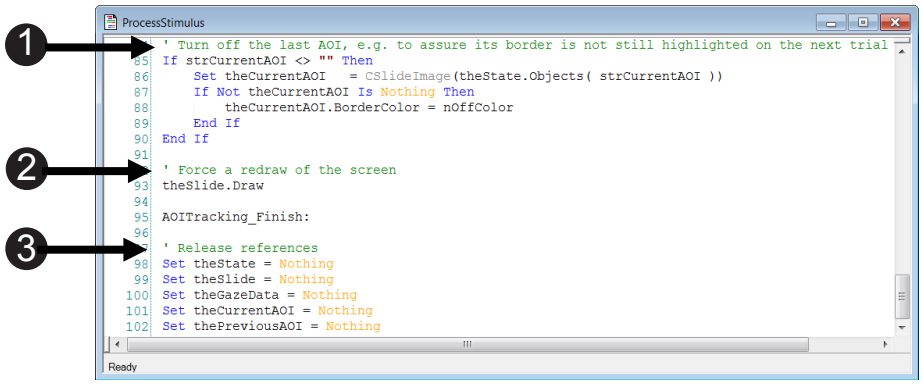
Clear the references and redraw the border for the next trial.

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.

2) ***Redraw*** the **screen.**

3) ***Release*** **References**.



Task 8: Run the Experiment

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

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

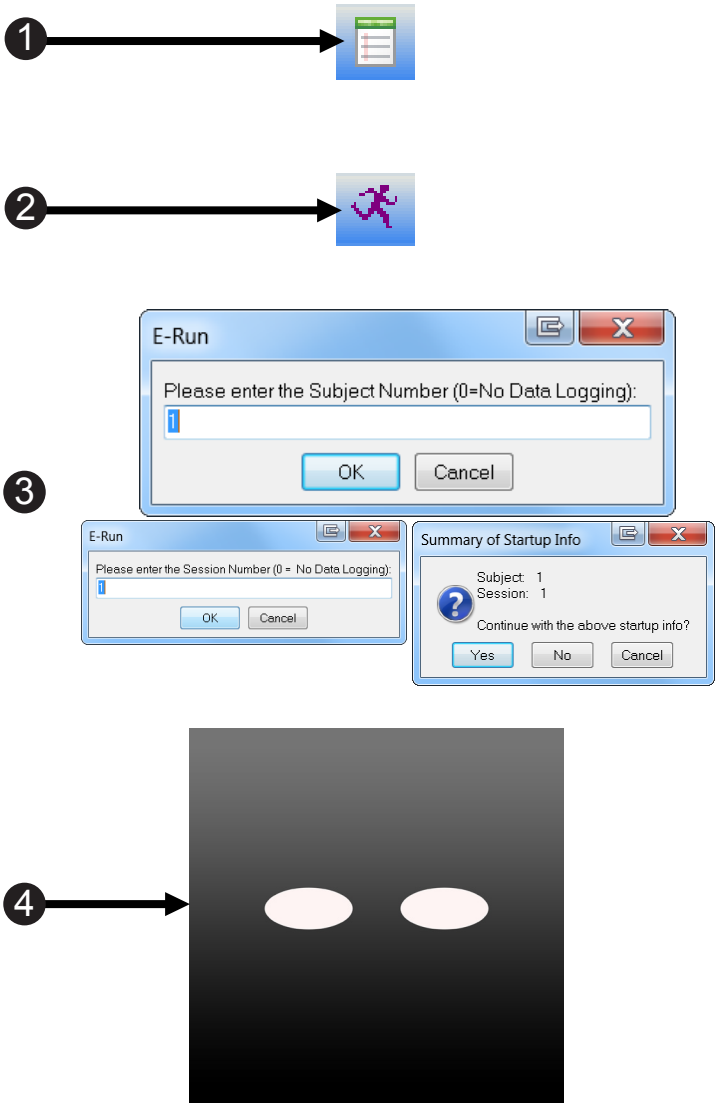
2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

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 **both eyes** are represented in the window and that **no arrows** appear. ***Press* Space** to close the track status window.

5) ***Perform* Calibration** (5 dots) and **Validation** (4 dots). ***Accept*** the **Calibration** results that are displayed, if x and y are each <=2 deg; otherwise, ***select*** the **"recalibrate"** option.

6) ***Perform*** the *three trials* in the experiment.



Chapter 3: **SMI PackageCall Reference**

3.1 Introduction

This section explains the functioning of each PackageCall and how to use the PackageCall parameters.

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.

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 EESMI installation.

First, you will need to make sure the SMI_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 SMIOpen PackageCall when testing your experiment without the availability of the SMI eye tracker your experiment will generate a runtime 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 SMI_Device Is Nothing Then GoTo SaveGazeData_Finish
If SMI_Device.GetState() <> ebStateOpen Then GoTo SaveGazeData_Finish
If SMI_bIsOpen <> True Then GoTo SaveGazeData_Finish
If SMI_bIsEnabled <> True Then GoTo SaveGazeData_Finish
```

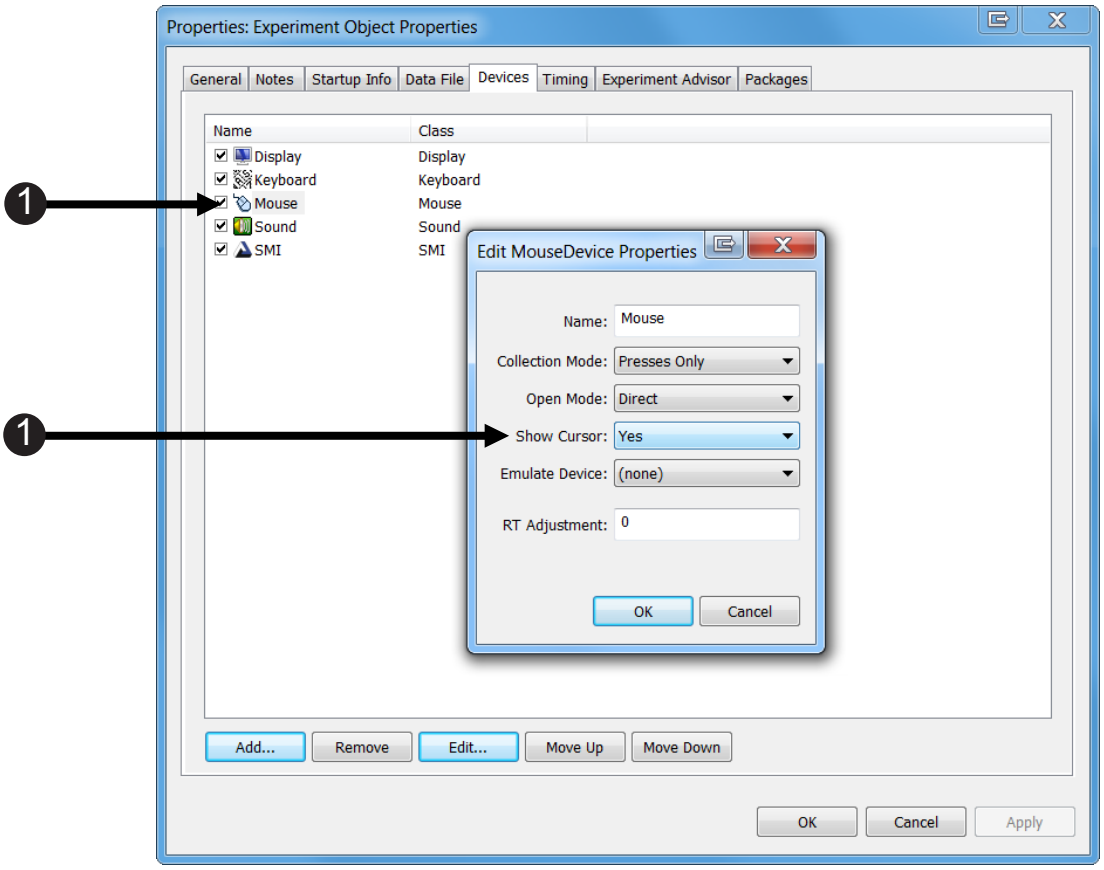
The second scenario you will need to account for is the use of Ctrl + Alt + Backspace or Ctrl + Shift. When the Ctrl + Alt + Backspace key combination is pressed during an experiment, the experiment terminates, but still produces an .edat2 file. The Ctrl + Shift key combination exits the experiment without 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 *SMIVaryingPositionAOI.es2* experiment for use in a Do loop.

```
' User requested exit?
If GetConditionalExitState() <> 0 Then Exit For
If  GetUserBreakState() <> 0 Then Exit For
```

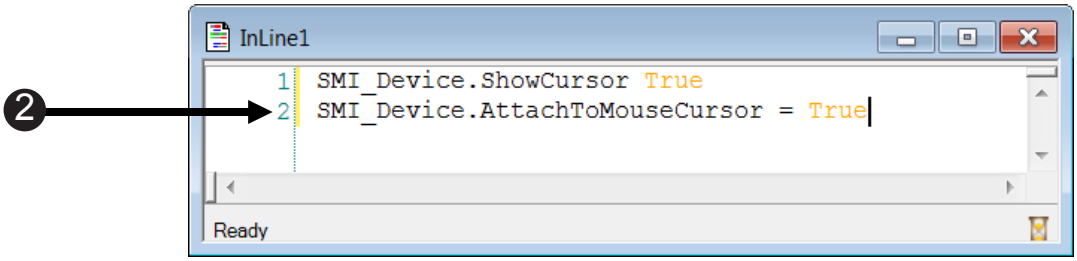
3.2 Calls to the SMI Device

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, make the cursor visible by either setting the MouseDevice.ShowCursor property to True via the Devices tab in the Experiment Property Pages (see image 1 below), or by setting the SMI Device's ShowCursor Property to True via InLine script (see image 2 below). Second, make the cursor active from the eye tracker by setting SMI_Device.AttachToMouseCursor = True in an InLine script (also shown in image 2 below).

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 **SMI_Device.AttachToMouseCursor = True**.



SMI_CalibInfant

Description
Performs a two-point infant calibration.

Syntax
```
SMI_CalibInfant c [,vValidate] [,vReserved1] [,vReserved2]
[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional vValidate As Variant

An optional parameter that specifies if validation should occur after calibration. If not specified, this will default to “True.”

Optional vReserved1 As Variant

An optional parameter reserved for future use.

Optional vReserved2 As Variant

An optional parameter reserved for future use.

Optional 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.”

SMI_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 accepts calibration point. Escape key aborts calibration.

Syntax

```
SMI_CalibManual c [,vValidate] [,vReserved1] [,vReserved2]
[,vHandlePreRelease]
```

Default Parameters

```
c
```

Parameters

c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional vValidate As Variant

An optional parameter that specifies if validation should occur after calibration. If not specified, this will default to “True.”

Optional vReserved1 As Variant

An optional parameter reserved for future use.

Optional vReserved2 As Variant

An optional parameter reserved for future use.

Optional 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.”

SMI_CalibRegular

Description

Performs the typical calibration, similar to that which is seen in the iView software. A colored ball moves across the screen to predetermined points where calibration data is collected automatically. The number of points used for calibration can be set via script using the SMI_ CalibSetDefaultNumPoints call. Escape key aborts calibration, Space key accepts calibration points (only the first if in semi-automatic mode), ‘a’ key aborts the current calibration point.

Syntax

```
SMI_CalibRegular c, nAutoAccept [,vValidate] [,vReserved1] [,vReserved2]
[,vHandlePreRelease]
```

Default Parameters

```
c, 1
```

Parameters

c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

nAutoAccept As Long

Sets calibration/validation point acceptance [2: full-automatic, 1:semi-automatic (default), 0: manual]. Defaults to ‘1’.

Optional vValidate As Variant

An optional parameter that specifies if validation should occur after calibration. If not specified, this will default to “True.”

Optional vReserved1 As Variant

An optional parameter reserved for future use.

Optional vReserved2 As Variant

An optional parameter reserved for future use.

Optional 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.”

SMI_ClearHistory

Description
This method calls RemoveAll on the History, which clears the history of accumulated eye tracking samples.

Syntax
```
SMI_ClearHistory c [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional 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."

SMI_Close

Description

Closes the package file, including stopping tracking and disconnecting if necessary. This routine does not close the SMI Device, rather it tears down the package file code. The SMI Device will automatically close when the experiment exits.

Syntax

```
SMI_Close c [,vHandlePreRelease]
```

Default Parameters

```
c
```

Parameters

c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional 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."

SMI_ CloseGazeDataFile

Description
Closes the .gazedata file if it is open.

Syntax
```
SMI_CloseGazeDataFile c [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional 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."

SMI_Connect

Description
Instructs the SMI E-Prime Device to establish a connection with the SMI hardware.

Syntax
```
SMI_Connect c [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional 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."

SMI_ContinueRecording

Description
Instructs the SMI Device to continue recording if recording is paused.

Syntax
```
SMI_ContinueRecording c [,vEtMessage] [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional vEtMessage As Variant

Text message that will be written to .idf data file. Defaults to “ContinueRecording”.

Optional 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.”

SMI_Disconnect

Description
Instructs the SMI E-Prime Device to disconnect from the SMI hardware and places the package file in the disconnected state.

Syntax
```
SMI_Disconnect c [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional 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."

SMI_GazeReplay

Description
Visually replays the history of samples over top of the given Slide.

Syntax
```
SMI_GazeReplay c, theSlide [,vStartTime] [,vStopTime] [,vHandlePreRelease]
```

Default Parameters
```
c,
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

theSlide As Slide

A Slide object that will be drawn on screen before replay begins.

Optional vStartTime As Variant

An optional parameter that creates 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.

Optional vStopTime As Variant

An optional parameter that creates 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.

Optional 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."

SMI_Open

Description
Opens the SMI package file.

Syntax
```
SMI_Open c [,vState] [,vConnect] [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional vState As Variant

An optional parameter for the requested state of SMI communications (“on”, “off”). If not specified this will default to “on”.

Optional vConnect As Variant

Optional parameter to specify whether or not to immediately connect to the SMI hardware. If not specified this will default to, “True.”

Optional 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.”

SMI_OpenGazeDataFile

Description
Opens the tab delimited .gazedata file.

Syntax
```
SMI_OpenGazeDataFile c [,vFilename] [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional 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 of the form: [ExperimentName]-[Subject#]-[Session#]. gazedata.

Optional 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
1) If the file exists any data in the file will be destroyed.
2) This function will use the next available file number, e.g. if this is the first file to be opened in an experiment it may use file #1 and then the user would have to use file #2 in their experiment.

SMI_ PauseRecording

Description
Instructs the SMI E-Prime Device to pause recording.

Syntax
```
SMI_PauseRecording c [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional 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."

SMI_ SetKeyboardDefault

Description
Sets the Keyboard Device to be used for default SMI package file operations.

Syntax
```
SMI_SetKeyboardDefault strMyKeyboard, bThrowErrorIfNonexistent
[,vHandlePreRelease]
```

Default Parameters
```
, True
```

Parameters
strMyKeyboard As String

The name of the InputDevice that handles the default input operations of the package file.

bThrowErrorIfNonexistent As Boolean

A default parameter that aborts the experiment if an InputDevice with name strMyKeyboard is not found. Otherwise, the Routine will return regardless of whether or not the InputDevice was found and any error will be suppressed.

Optional 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."

SMI_SetMonitorCalibration

Description
Sets the DisplayDevice used for the SMI calibration and validation screen.

Syntax
```
SMI_SetMonitorCalibration strMyMonitor, bThrowErrorIfNonexistent
[,vHandlePreRelease]
```

Default Parameters
```
, True
```

Parameters
strMyMonitor As String

The monitor to use for the calibration screen.

bThrowErrorIfNonexistent As Boolean

A default parameter that aborts the experiment if an DisplayDevice with Name strMyMonitor is not found. Otherwise, the Routine will return regardless of whether or not the DisplayDevice was found and any error will be suppressed.

Optional 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."

SMI_SetMonitorCalibrationResult

Description
Sets the DisplayDevice used for the SMI validation result screen.

Syntax
```
SMI_SetMonitorCalibrationResult strMyMonitor, bThrowErrorIfNonexistent
```

Default Parameters
```
, True
```

Parameters
strMyMonitor As String

The monitor to use for the calibration results screen.

bThrowErrorIfNonexistent As Boolean

A default parameter that aborts the experiment if an DisplayDevice with Name strMyMonitor is not found. Otherwise, the Routine will return regardless of whether or not the DisplayDevice was found and any error will be suppressed.

Optional 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."

SMI_SetMonitorDefault

Description
Sets the DisplayDevice to be used for default SMI package file operations.

Syntax
```
SMI_SetMonitorDefault strMyMonitor, bThrowErrorIfNonexistent
```

Default Parameters
```
, True
```

Parameters
strMyMonitor As String

The Name of the DisplayDevice that handles the default drawing operations of the package file.

bThrowErrorIfNonexistent As Boolean

A default parameter that aborts the experiment if an DisplayDevice with Name strMyMonitor is not found. Otherwise, the Routine will return regardless of whether or not the DisplayDevice was found and any error will be suppressed.

Optional 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
- If you are using multiple monitors, you will need to use SetDefaultMonitor to set the monitor, and use this call early in SessionProc.

SMI_SetMonitorGazeReplay

Description
Sets the DisplayDevice used for the SMI gaze replay screen.

Syntax
```
SMI_SetMonitorGazeReplay strMyMonitor, bThrowErrorIfNonexistent
```

Default Parameters
```
, True
```

Parameters
strMyMonitor As String

The monitor to use for the gaze replay screen.

bThrowErrorIfNonexistent As Boolean

A default parameter that aborts the experiment if an DisplayDevice with Name strMyMonitor is not found. Otherwise, the Routine will return regardless of whether or not the DisplayDevice was found and any error will be suppressed.

Optional 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."

SMI_SetMonitorTrackStatus

Description
Sets the DisplayDevice used for the SMI track status screen.

Syntax
```
SMI_SetMonitorTrackStatus strMyMonitor, bThrowErrorIfNonexistent
```

Default Parameters
```
, True
```

Parameters
strMyMonitor As String

The monitor to use for the track status screen.

bThrowErrorIfNonexistent As Boolean

A default parameter that aborts the experiment if an DisplayDevice with Name strMyMonitor is not found. Otherwise, the Routine will return regardless of whether or not the DisplayDevice was found and any error will be suppressed.

Optional 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."

SMI_SetMouseDefault

Description
Sets the Mouse Device to be used for default SMI package file operations.

Syntax
```
SMI_SetMouse strMyMouse, bThrowErrorIfNonexistent
```

Default Parameters
```
, True
```

Parameters
strMyMouse As String

The Name of the Mouse Device that handles the mouse operations of the package file.

bThrowErrorIfNonexistent As Boolean

A default parameter that aborts the experiment if a Mouse Device with Name strMyMouse is not found. Otherwise, the Routine will return regardless of whether or not the Mouse Device was found and any error will be suppressed.

Optional 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."

SMI_StartRecording

Description
Instructs the SMI E-Prime Device to start recording.

Syntax
```
SMI_StartRecording c [,vClearRecordingBuffer] [,vPauseRecording]
[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional vClearRecordingBufferAs Variant

An optional parameter that specifies if the ClearRecordingBuffer method of the SMI Device should be called. If not specified this will default to “True.”

Optional vPauseRecording As Variant

An optional parameter that specifies if the recording should be paused immediately after it is started. If not specified, this will default to True.

Optional 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.”

SMI_StartTracking

Description
Instructs the SMI E-Prime Device to start processing eye tracker samples.

Syntax
```
SMI_StartTracking c [,vClearHistory] [,vEtMessage] [vContinueRecording]
[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional vClearHistory As Variant

An optional parameter to specify whether or not the data in the History should be cleared when tracking is started. If not specified this will default to “True.”

Optional vEtMessage As Variant

An optional parameter that is the image message used to separate trials. If the image message has the suffix “.jpg”, “.bmp”, “.png”, or “.avi” BeGaze will separate the data buffer automatically into according trials. The value defaults to “StartTracking.bmp” if unspecified.

Optional vContinueRecording As Variant

An optional parameter that specifies if the recording should be continued. If not specified, this will default to True.

Optional 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.”

SMI_StopRecording

Description
Instructs the SMI E-Prime Device to stop recording.

Syntax
```
SMI_StopRecording c [,vSaveData] [,vDirectory] [,vFilename] [,vReserved1]
[,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional vSaveData As Variant

An optional parameter that specifies if the SaveData method of the SMI Device should be called. If not specified this will default to “True.”

Optional vDirectory As Variant

An optional parameter that is the file directory to be used if SaveData is called. If specified the value will be combined with vFilename to produce a full path. If left unspecified, SaveData will be provided only vFilename.

Optional vFilename As Variant

An optional parameter that is the filename to be used if SaveData is called. If not specified and SaveData is called, “[DataFile.Filename].idf” will be used.

Optional vReserved1 As Variant

An optional parameter reserved for future use.

Optional 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.”

SMI_StopTracking

Description

Instructs the SMI E-Prime Device to stop processing eye tracker samples. In order to save the eye tracking data OpenGazeDataFile must be called before StartTracking and CloseGazeDataFile must be called after StopTracking. Otherwise no eye tracking data will be saved.

Syntax

```
SMI_StopTracking c [,vPauseRecording] [,vHandlePreRelease]
```

Default Parameters

```
c
```

Parameters

c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional vPauseRecording As Variant

An optional parameter that specifies if the recording should be paused immediately after it is started. If not specified, this will default to True.

Optional 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."

SMI_TrackStatus

Description
Displays the track status screen.

Syntax
```
SMI_TrackStatus c [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional 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."

SMI_ Validate

Description

Performs the typical validation, similar to that which is seen in the iView software. An Accuracy report is shown after successful validation. Escape key aborts validation, Space key accepts validation points (only the first if in semi-automatic mode), ‘a’ key aborts the current validation point. Validation behavior may be contingent on the eye tracker model in use.

Syntax

```
SMI_Validate  c [,vAccuracyReportDuration] [,vReserved1]
[,vHandlePreRelease]
```

Default Parameters

```
c
```

Parameters

c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

Optional AccuracyReportDuration As Variant

An optional parameter to specify the duration (in milliseconds) the Accuracy report is shown after a successful validation. If not specified, the value defaults to **5000** ms. To display the Accuracy report until the Space key is pressed, specify ‘-1’.

Optional vReserved1 As Variant

An optional parameter reserved for future use.

Optional 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

- We recommend that one of the included calibration routines be called at least once prior to performing validation. This ensures that the same configuration settings that were applied to the calibration will also be used for validation within the same experimental session.

SMI_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

```
SMI_WaitForFixation c, theSlide, nMinFixationDuration
[,vVisualFeedbackColor] [,vKey]
```

Default Parameters

```
c, ,
```

Parameters

c As Context

The experiment context allows the user to add variables to the experiment. The routines in the 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.

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.

Optional 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.

Optional 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.

Appendix A: **Locate Your IP Address and Ping Another Computer**

The typical EESMI experiment is run in a single machine environment. This means that the same computer that is running E-Prime is also running the iView server and is receiving the eye tracking data from the eye tracker. The single machine model is usually used for SMI's mobile eye trackers. When an SMI Device is added to an E-Prime experiment, the default IP address for the SMI and the E-Prime machine are the same. If you are running on a two-computer setup, then the SMI and E-Prime computer's IP addresses will be different. Follow the steps below to identify the IP address for the computer that is running the iView server.

1) To ***find*** the **IP address** of the **SMI machine** on a **Windows machine**, go to **Start Menu** > **Run**.

2) In the **Run** dialog, ***type*** "**cmd**" and ***click*** **OK**.

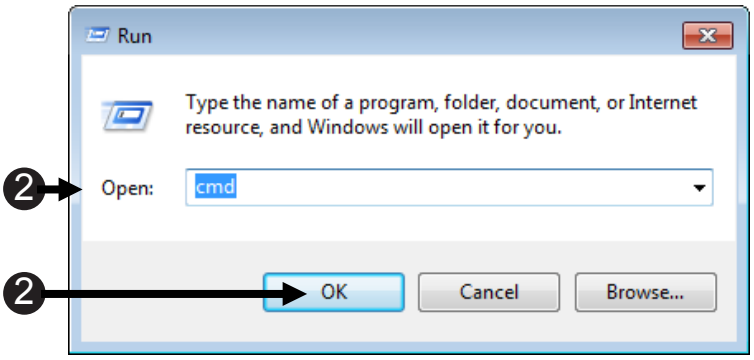
3) A **command prompt** (a dialog box with a black background and white text) will ***open***.

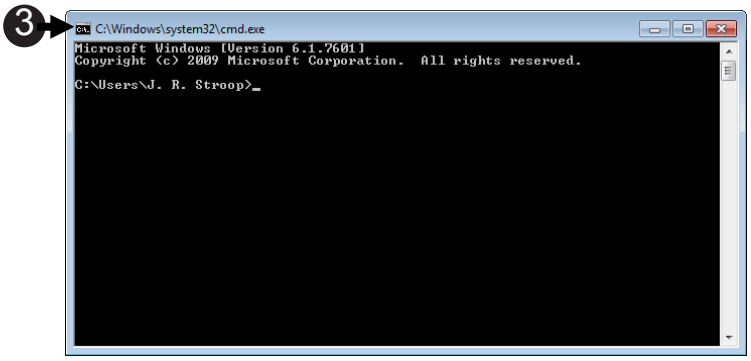
4) The **cmd.exe** will ***display*** a prompt, ***type*** "**ipconfig**", ***press*** **Enter**.

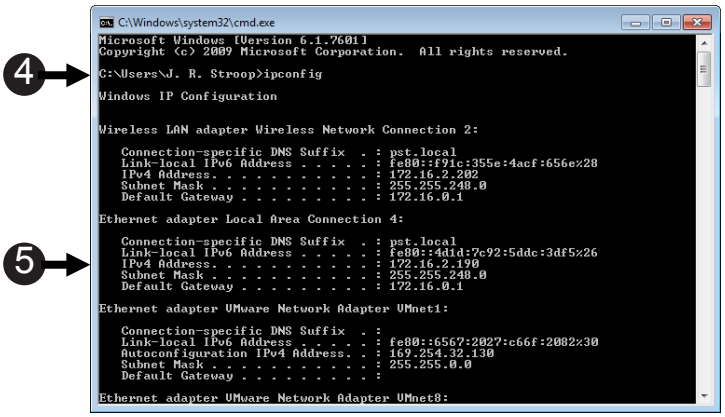
5) The **IP Address** will be labeled in the **command prompt window**.

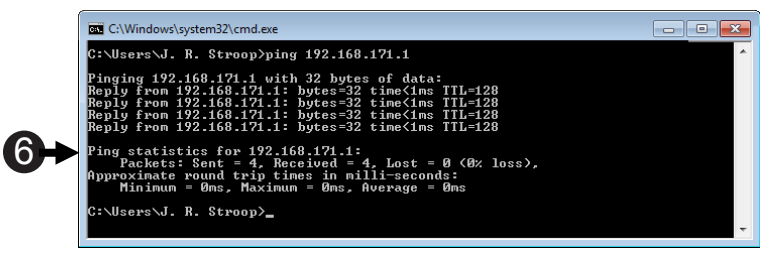
⚠ **NOTE:** *Not all IP Addresses will look the same. In this example, the **IP address** is **IPv4 Address.***

6) If the **machines** are able to ***communicate*** you will get **reply messages**. If the **machines** cannot ***communicate*** you will get **error messages**.









Appendix B: **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 tutorials that are installed with your EESMI 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 SMIStartTracking PackageCall and stop filling the buffer when you call SMIStopTracking PackageCall. Contained in the SMIStartTracking 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 SMIStartTracking 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 SMIClearHistory 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.

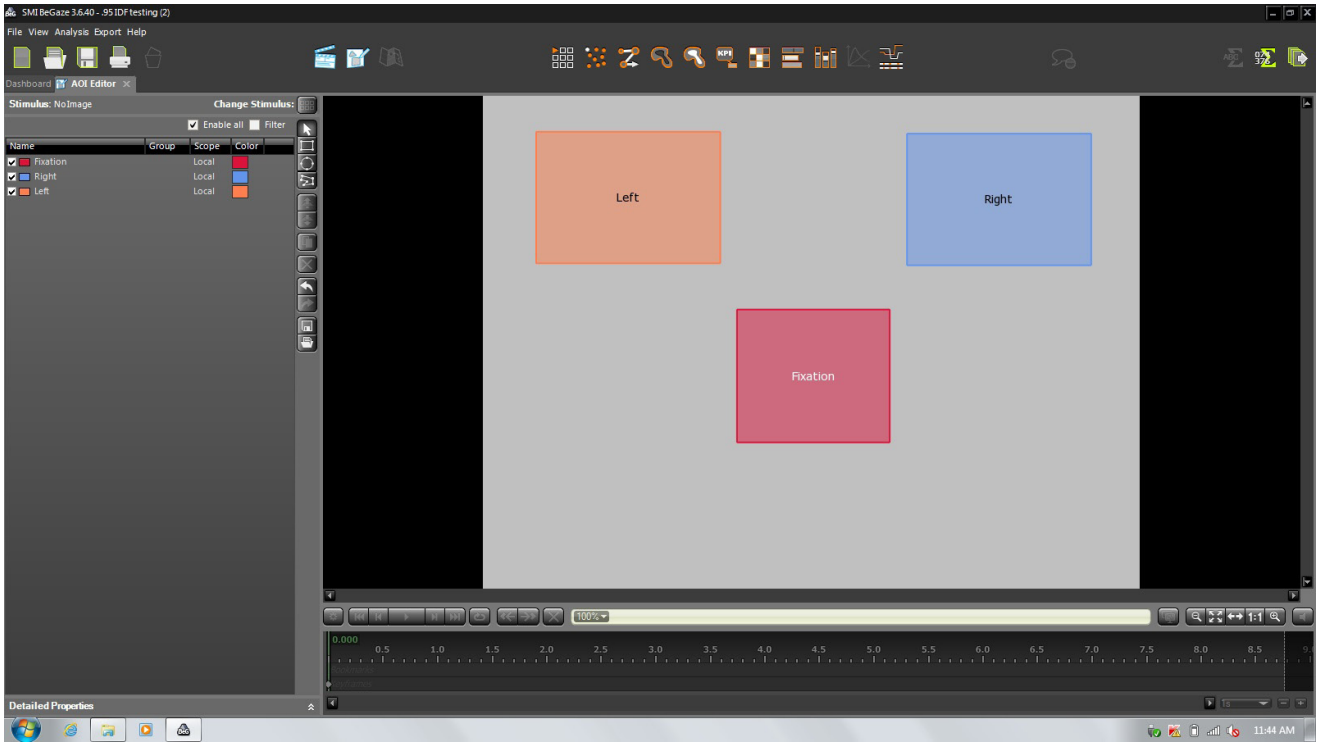
		Time to Fill Buffer (secs)							
		Eye Tracker Speed (Hertz)							
		30Hz	50Hz	60Hz	120Hz	300Hz	500Hz	1000Hz	
“Buffer Size (Samples)”	1024	34.13	20.48	17.07	8.53	3.41	2.05	1.02	
	2048	68.27	40.96	34.13	17.07	6.83	4.10	2.05	
	4096	**136.53**	**81.92**	**68.27**	**34.13**	**13.65**	**8.19**	**4.10**	**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	

Appendix C: **Gazedata File Content**

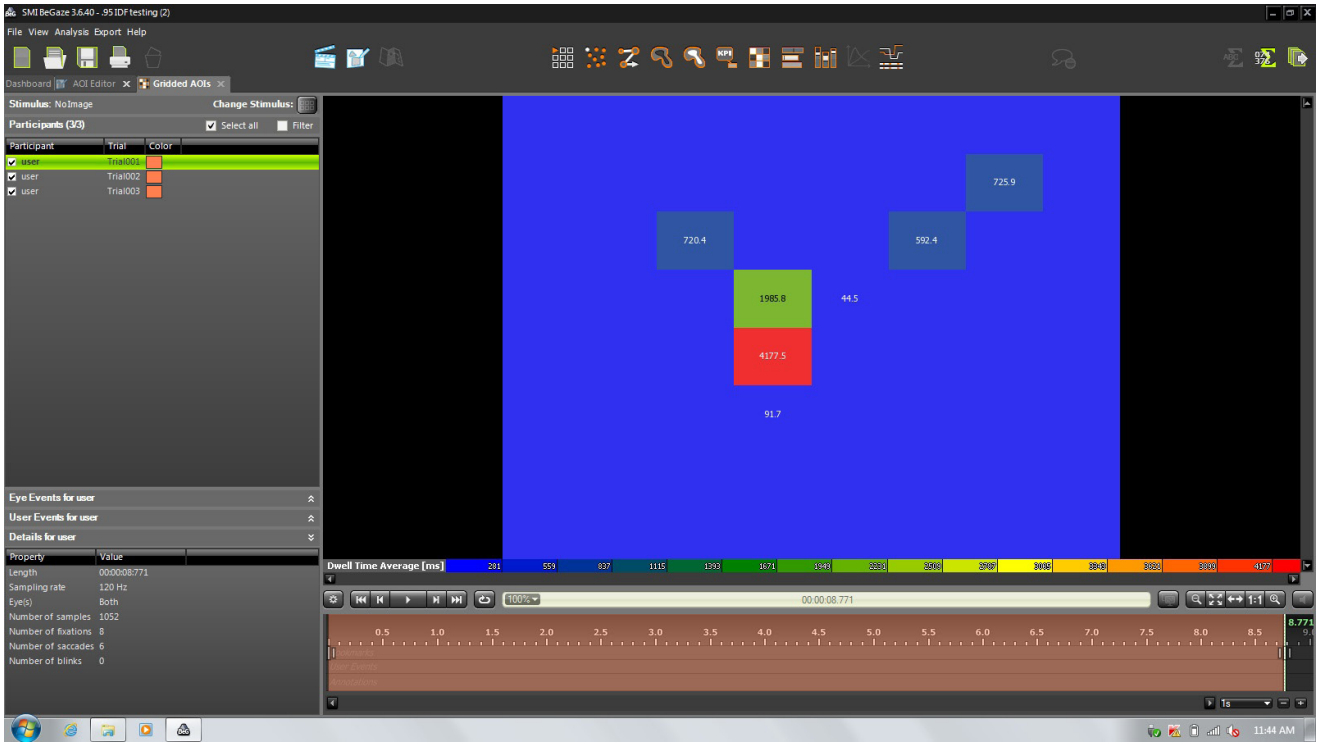
.gazedata file	
Column	***Definition***
Subject	Subject number (from E-Prime Startup info)
Session	Session number (from E-Prime Startup info)
ID	Counter of the current numbered gaze data point acquired
SMITime	
RTTime	The timestamp in milliseconds based on E-Prime's clock
CursorX	The X coordinate of the response
CursorY	The Y coordinate of the response
TimeStamp	Timestamp of the current gaze data sample, in microseconds
TimestampMillisec	Timestamp of the current gaze data sample, in milliseconds
TrialID	E-Prime trial counter
Prime	Name of the prime stimulus
AOI1	Description of stimulus that appears in the AOI1
AOI2	Description of stimulus that appears in the AOI2
AOI	AOI identifier that is currently being fixated upon
AOIStimulus	Description of the contents of the AOI that is currently being fixated upon
CRESP	Correct response for the trial
RESP	Response made by the participant
ACC	Accuracy of the participant's response (1 = correct, 0 = incorrect, 2 = omission)
RT	Response reaction time of the participant to the target display
Current Object	Object from the trial procedure that is displayed on screen for the current eye gaze data sample
LeftEyeDiam	Pupil diameter for left eye, in mm
LeftEyePositionX	Horizontal eye position relative to camera for the left eye, in mm
LeftEyePositionY	Vertical eye position relative to camera for the left eye, in mm
LeftEyePositionZ	Distance to camera for the left eye, in mm
LeftEyeGazeX	Horizontal gaze position on screen for the left eye, in pixels
LeftEyeGazeY	Vertical gaze position on screen for the left eye, in pixels
RightEyeDiam	Pupil diameter for right eye, in mm
RightEyePositionX	Horizontal eye position relative to camera for the right eye, in mm
RightEyePositionY	Vertical eye position relative to camera for the right eye, in mm
RightEyePositionZ	Distance to camera for the right eye, in mm
RightEyeGazeX	Horizontal gaze position on screen for the right eye, in pixels
RightEyeGazeY	Vertical gaze position on screen for the right eye, in pixels

Appendix D: **AOI Analysis in BeGaze**

The screen grab below illustrates a simple AOI analysis that be constructed in BeGaze from the markers experiment described in Tutorial 4.

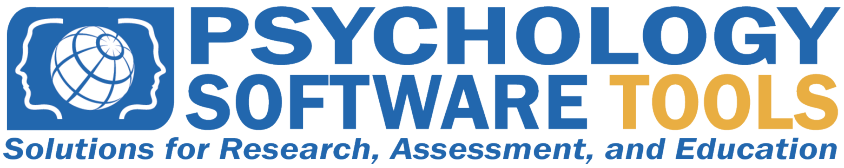


The screen grab below illustrates a gridded AOI analysis that be constructed in BeGaze from the markers experiment described in Tutorial 4.



Appendix E: **Contact Information**

For additional information or support



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 support and technical issues
Please e-mail us at **info@pstnet.com**
or visit: **http://www.pstnet.com/support/login.asp**