# User Manual

**For Research and Education Only**

*Psychology Software Tools, Inc.*
*311 23rd Street Extension, Suite 200*
*Sharpsburg, PA 15215-2821*
*Phone 412.449.0078*
*Fax: 412-449-0079*
*E-mail: info@pstnet.com*

**E-Prime Extensions for fMRI 3.0 User Manual**
**PST-101765**
**Rev 1**

**Copyright**
Copyright 2018 Psychology Software Tools, Inc. All Rights Reserved.

The information in this document is subject to change without notice. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced, or distributed in any form or by any means, or stored in a database or retrieval system, without prior written permission of Psychology Software Tools, Inc.

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

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

Software Notice: The enclosed software is provided for use by a single user who has purchased the software. The software MAY NOT be reproduced or distributed to others. Unauthorized reproduction and/or sales of the enclosed software may result in criminal and civil prosecution (17 USC 506).

**Trademark**
Psychology Software Tools, Inc. (PST), the Psychology Software Tools, Inc. logo, E-Prime® Extensions for EEfMRI, the E-Prime® Extensions for fMRI logo, E-Prime® and the E-Prime® logo are trademarks or registered trademarks of Psychology Software Tools, Inc. Windows® and Excel® are registered trademarks of Microsoft Corporation in the United States and other countries.

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

**E-Prime Extensions for fMRI Software is for research and educational purposes only.**

# Table of Contents


1. Getting Started ..... 5
   1.1. Product Overview ..... 5
   1.2. System Requirements ..... 5
   1.3. Product Service and Support ..... 6
   1.4. Resources ..... 6
2. Installing E-Prime Extensions for fMRI ..... 7
   2.1. Before Installation ..... 7
   2.2. During Installation ..... 8
   2.3. After Installation ..... 11
3. How to Create an EEfMRI Experiment ..... 12
   3.1. Overview ..... 12
   3.2. Experiment Design ..... 12
   3.3. Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment ..... 16
   3.4. Tutorial 2: Creating a Menu ..... 34
   3.5. Tutorial 3: Timing ..... 48
   3.6. Tutorial 4: Logging Block, Stimulus, and Response Data in the .pdat File ..... 52
   3.7. Tutorial 5: Creating a Practice Run ..... 63
4. Troubleshooting ..... 73
   4.1. Cannot locate E-Prime Samples and Tutorials ..... 73
   4.2. Missing Parameter Errors ..... 74
   4.3. Running an Experiment Without EEfMRI Enabled ..... 75

Appendix A: Samples and Tutorials ..... 76
Appendix B: MapperOne Experiment ..... 77
Appendix C: The .pdat File Form ..... 81
Appendix D: Add Celeritas to the Experiment ..... 82
5. Contact Information ..... 85

# 1. Getting Started

## 1.1. Product Overview

E-Prime Extensions for fMRI (EEfMRI) is a collection of customized software routines and other productivity tools designed to assist E-Prime users in creating full featured, flexible, and robust fMRI experiments. Use of EEfMRI provides the experimenter with more Timing and Stimulus control by synchronizing the start of an experimental task with the scanner trigger pulse as well as simplified logging of user-specified experiment events for later analysis. EEfMRI also features an easy to implement menu system that allows the experimenter to group multiple tasks into a single experiment and interactively choose which tasks to present at runtime. The menu system allows the user to select the experiment task from a list and includes the ability to interrupt a running task in a controlled manner to restart tasks without terminating the entire experiment.

This manual shows how to install the EEfMRI software, explains how to structure an EEfMRI-enabled E-Prime experiment, and provides useful tips for situations you may encounter while upgrading or creating EEfMRI-enabled experiment paradigms.

We caution you not to work with the tools provided until you have read the documentation and worked through the tutorials. We encourage you to learn the system by following the tutorials in the order they are presented. This minimizes your learning time and allow you to take advantage of the available tools to produce functional paradigms ready for use with the fMRI system.

## 1.2. System Requirements

E-Prime Extensions for fMRI is compatible with ***E-Prime 3.0 only***. E-Prime 2.0 is not supported with the current software. The E-Prime 3.0 file extension is .es3. This manual uses the .es3 extension. Prior to the EEfMRI installation, you need to determine the version of E-Prime on your machine.

**NOTE**: *If permitted by your research institution, we recommend updating to the latest version of E-Prime for new features and bug fixes.*

***Software Compatibility Requirements***

E-Prime 3.0 and E-Prime Extensions for fMRI are compatible with machines running Microsoft® Windows® 7 SP1, 8, 10.

The remainder of this document shows images of E-Prime 3.0 running on a Windows 10 system. Your screen images may vary slightly depending on operating system and E-Prime version.

***Computer Requirements for the machine running E-Prime***

Additionally, the computer must comply with the minimum E-Prime 3.0 requirements:

E-Prime 3.0 Requirements

## 1.3. Product Service and Support

If you have difficulty running an experiment, take note of any error messages and view the suggestions in the *Chapter 4: Troubleshooting* section. If you still have issues, please contact PST's Product Service and Support.

***Technical Support***

Psychology Software Tools, Inc. provides technical support for E-Prime via the PST Product Service and Support website. To receive technical support, you must register online at:

https://support.pstnet.com/

Registration requires a valid E-Prime serial number, an E-Prime Extensions for fMRI serial number, and an e-mail address. A Knowledge Base is available on our Product Service and Support website with release notes and frequently asked questions. The site includes E-Prime sample paradigms available for download, How-To videos, and the E-Prime 3.0 Experiment Library.

Our Product Service and Support website offers an additional support forum you can use to post general questions about E-Prime and E-Prime Extensions for fMRI:

Psychology Software Tools Community Forum

Additionally, Psychology Software Tools has a YouTube channel with various Live Stream Webinars and How-To videos:

PSTNET on YouTube

**NOTE**: *This document contains links to our online Knowledge Base which has articles for E-Prime Extensions for fMRI. If you have trouble opening the link, please search the article number on our Product Service and Support website.*

## 1.4. Resources

***Psychology Software Tools Resources***

- E-Prime Extensions for fMRI product page:

EEfMRI Product Page

- E-Prime Extensions for fMRI download link:

EEfMRI Download Link

**NOTE**: *You must be a registered user and logged into the PST support site to access the downloads for the registered product. For details, see the following Knowledge Base article:*

Article 27523: How to Register for Support and Access Downloads

# 2. Installing E-Prime Extensions for fMRI

## 2.1. Before Installation

E-Prime Extensions for fMRI 3.0 is compatible ***with E-Prime 3.0 only***. Prior to installing, determine which version of E-Prime exists on your machine.

For installation, you need either the EEfMRI 3.0 USB Flash Drive or the installation download that we host on our website. If using the USB Flash Drive, navigate in Windows Explorer and run the E-Prime Extensions for fMRI 3.0 Setup.exe file. If using an installation download provided by our website, navigate to the Psychology Software Tools Support page and click on the download link (See *Chapter 1: Section 1.4: Resources*).

After you have determined that the correct version of E-Prime is installed on your machine, you can install E-Prime Extensions for fMRI 3.0. Before continuing, be sure that you have administrative rights to install the software on your machine. The software cannot be installed without administrative rights. If you are unaware of your rights, please contact your system administrator.

If during the installation process you encounter issues that prohibit installation, contact your system administrator or contact PST's Product Service and Support at:

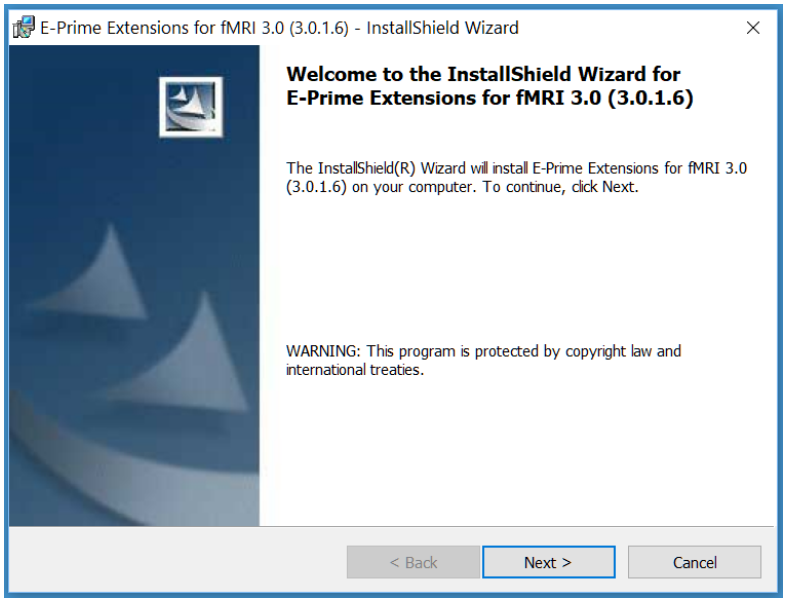
https://support.pstnet.com/

**NOTE**: *Please retain the packaging for your EEfMRI 3.0 Installation; it contains a label with your serial number which is required for installation. Additionally, register your serial number on our Product Service and Support website. Once your serial number is registered you can access our Downloads section of our website.*

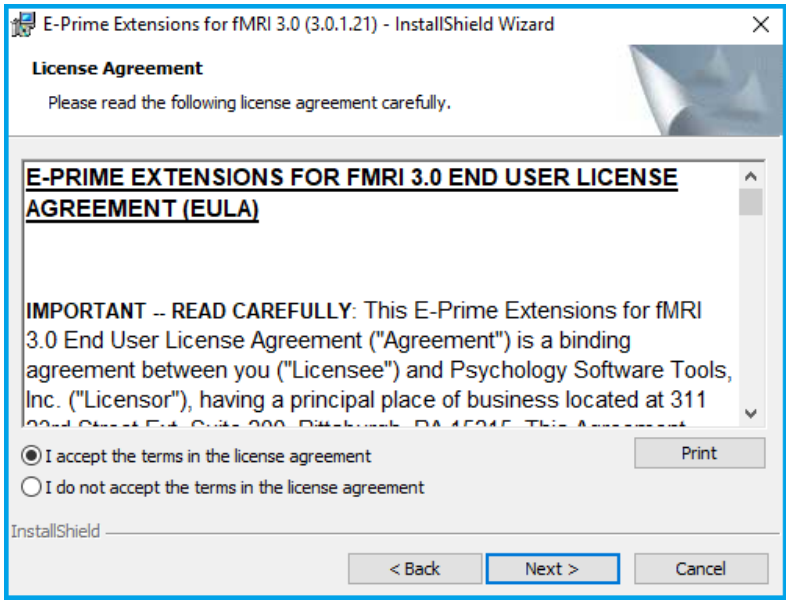
## 2.2. During Installation

**NOTE**: *The version number on the following images may not correspond to the version you are installing.*

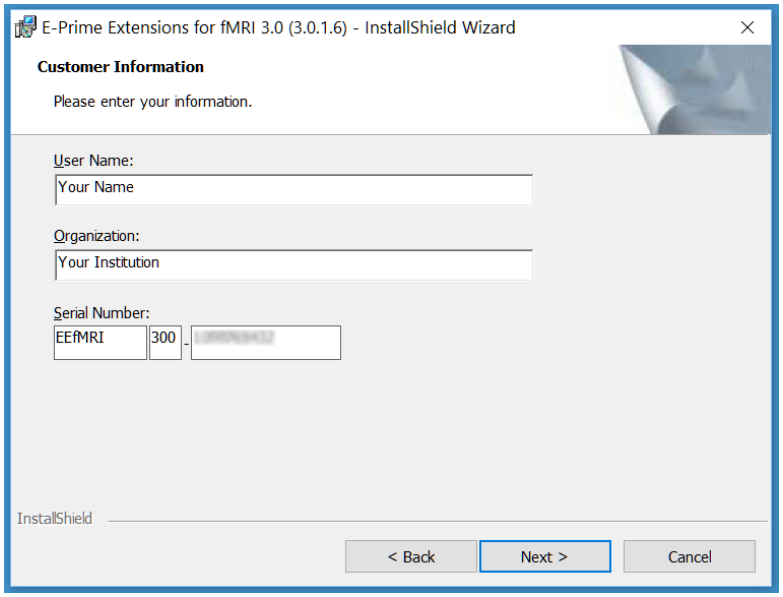
1) ***Insert*** the **E-Prime Extensions for fMRI 3.0 USB Flash Drive** into your USB port. If the Setup program does not automatically start, ***navigate*** to your **USB drive** in Windows Explorer and ***double click*** **Setup.exe** to launch the installer. ***Click*** **Next** to continue after the InstallShield Wizard begins.




2) ***Read***, and *if you agree*, ***Accept*** the **Terms and Conditions**. ***Click*** **Next** to continue.

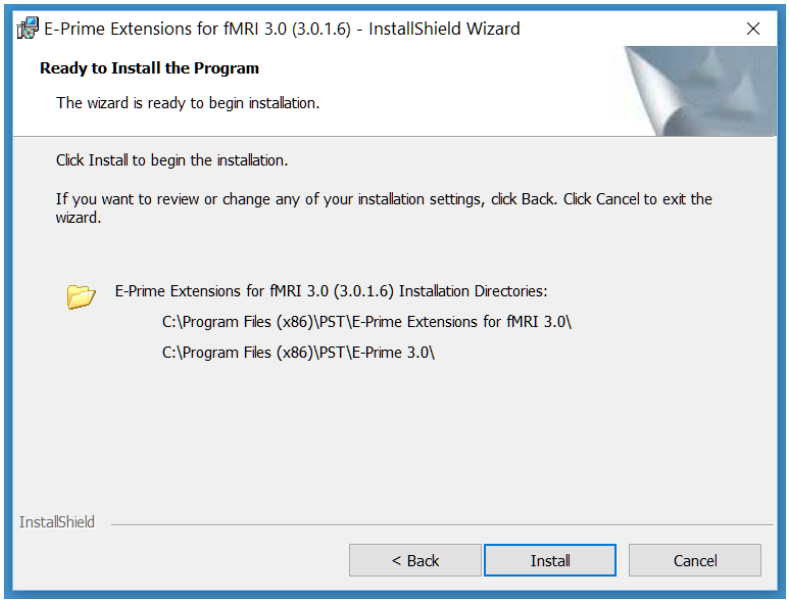


3) ***Specify*** your **User Name, Institution,** and **Serial Number**. ***Click*** **Next** to continue.

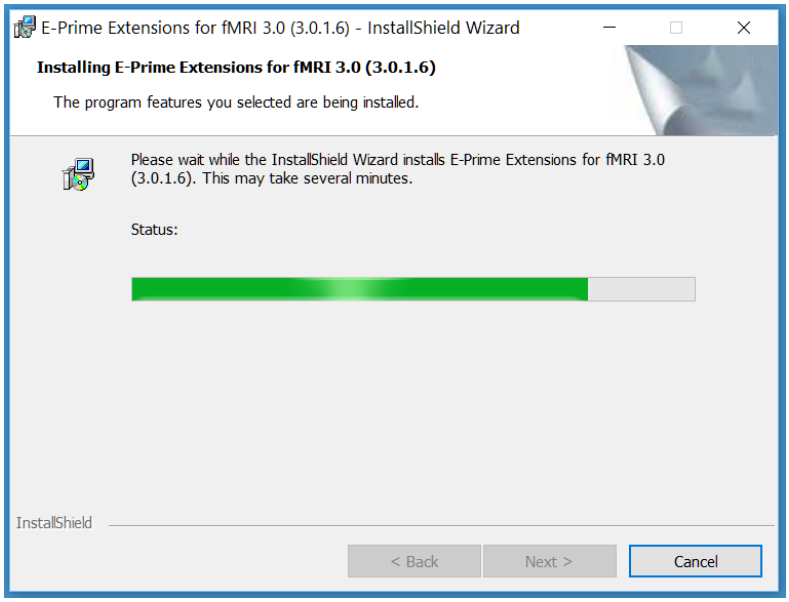



**NOTE**: *Please retain the packaging for your EEfMRI 3.0 Installation; it contains a label with your serial number which is required for installation.*

4) ***Review*** the installation **folder**. ***Click*** **Install** to continue.

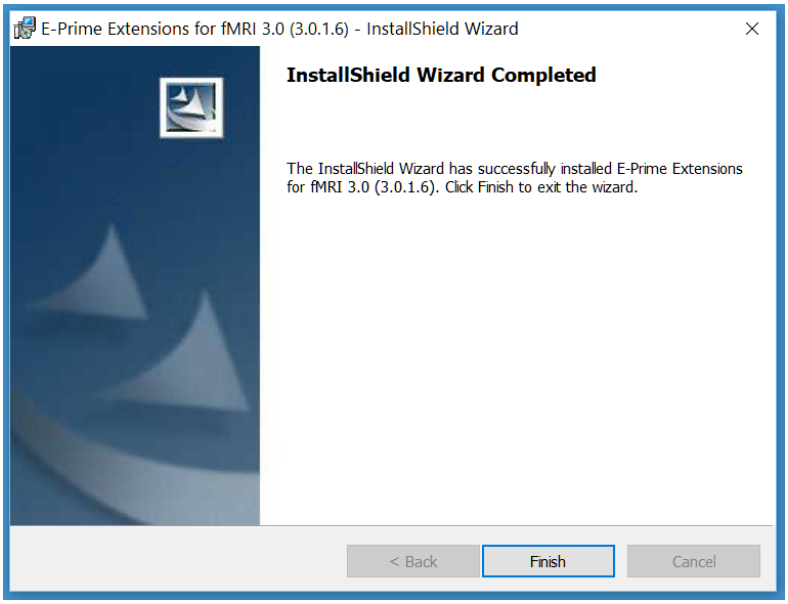


5) ***Wait*** while the software installs.




6) ***Wait*** for the confirmation message that the software installs successfully.

7) ***Click*** **Finish** to complete the installation process.



## 2.3. After Installation

### *Finding the My Experiment Folder*

E-Prime 3.0 is compatible with machines running Microsoft® Windows® 7 SP1, 8, and 10. The "**My Experiments**" 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**" and "**Tutorials**" folders which stores the experiments discussed in *Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment*.

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

**NOTE**: *The path to the personal documents folder may have been modified by your administrator.*

The table below shows the default paths to your personal documents folder:

| Folder | Path to your personal documents folder ("My Documents" or "Documents") |
|---|---|
| *Windows 7/8/10 | <drive>\Users\<user name>\Documents\My Experiments\3.0\ |

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

### *E-Prime Extensions for fMRI Samples and Tutorials*

The E-Prime Extensions for fMRI 3.0 folder is located within the "…My Experiments" folder, as shown below. The fMRI folder contains the Samples and Tutorials subfolders:

| Folder Type | Path to Samples or Tutorials |
|---|---|
| Samples | …My Experiments\3.0\fMRI\Samples |
| Tutorials | …My Experiments\3.0\fMRI\Tutorials |

For a listing of all installed Samples and Tutorials please refer to *Appendix A: Samples and Tutorials*.

**NOTE**: *This User Manual contains Tutorials. Both the starting point for the Tutorial and a finished copy of the Tutorial are provided for you* (see *Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment*).

### *Opening E-Studio*

After E-Prime Extensions for fMRI 3.0 installs, launch E-Studio. When E-Studio launches, it detects that the …My Experiments folder needs to be updated to include the E-Prime Extensions for fMRI 3.0 Sample and Tutorials. By specifying 'Yes,' you allow E-Prime to copy the new Samples and Tutorials into your …My Experiments\3.0 folder.

**NOTE**: *Copying the Samples and Tutorials upon launching E-Studio overwrites your existing Samples and Tutorials. If you wish to overwrite your files at a later point in time refer to Troubleshooting, 4.1 Cannot locate E-Prime Samples and Tutorials.*

# 3. How to Create an EEfMRI Experiment

## 3.1. Overview

This chapter illustrates the creation of an EEfMRI experiment using one of the sample paradigms provided with EEfMRI, the Press paradigm. Additionally, it introduces and explains some programming methods that E-Prime uses.

To complete the Tutorials, you need a computer with E-Prime and the EEfMRI software already installed. If you have not installed E-Prime 3.0, please complete the installation before continuing (see *Article 22670: Installation, Registration and Validation*). If you have not installed E-Prime Extensions for fMRI 3.0, please refer to the installation section in this manual (see *Chapter 2: Installing E-Prime Extensions for fMRI*).

While this document includes a basic introduction to programming concepts specific to using E-Prime with EEfMRI, the Tutorials assume you are familiar with using E-Prime to build behavioral experiments. If you are new to using E-Prime, we recommend that you work through all the Tutorials provided on our Product Service and Support website prior to beginning this tutorial.

Article 23282: Experiment Design (Beginner)
Article 22741: Experiment Design (Advanced)

## 3.2. Experiment Design

***Overview***

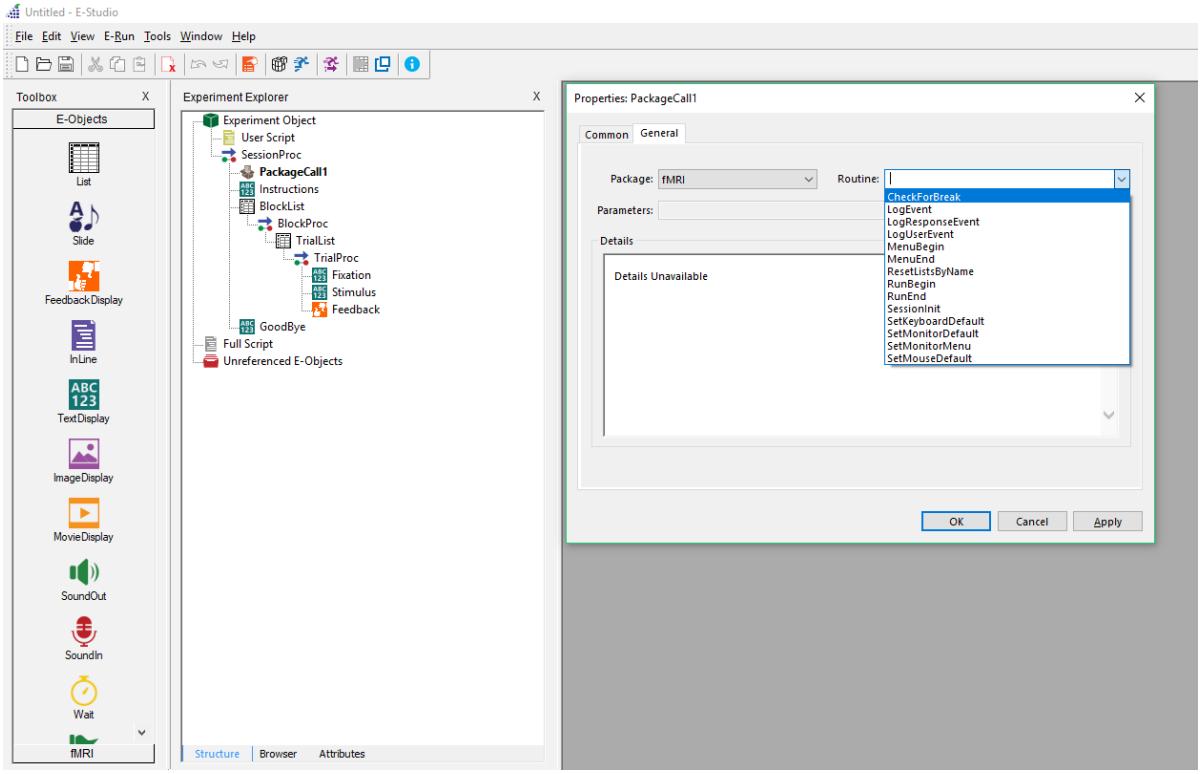
To add support for EEfMRI to an existing E-Prime experiment, you need to utilize the PackageFile feature of E-Prime. An E-Prime **PackageFile** is a pre-written, cohesive set of E-Basic script routines that are grouped together and distributed in a single file. The PackageFile can be authored and maintained externally and then shared amongst different E-Prime experiments. The Routines contained in the PackageFile serve to expand the default functionality offered by E-Prime for building experiments that interact with third party hardware and software products.

The E-Prime Extensions for fMRI 3.0 PackageFile is included in the E-Prime Extensions for fMRI installation. The Routines in the PackageFile allow you to automatically perform various functions (e.g., display an interactive menu of tasks or log data that is useful for analysis of an fMRI experiment). These Routines are invoked via graphical representations in the Experiment Explorer window in E-Studio. Some of the Routines require you to set or modify parameters to enable them to work within the context of your experiment. Once a PackageFile is added to an experiment, there are several methods that can be used to access the Routines. A brief overview of each method is provided below.

***PackageCall Method***

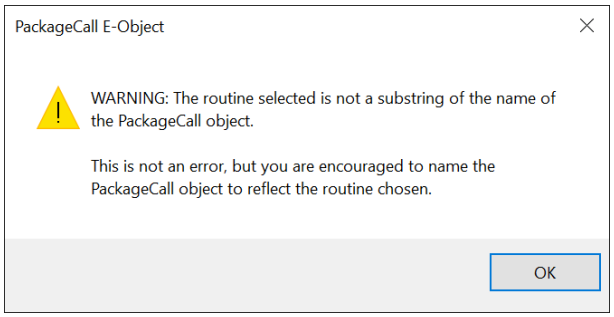
Routines contained within a PackageFile are typically called by dragging a **PackageCall** Object from the E-Studio Toolbox. The PackageCall is then dropped into a Procedure Object at the location within the experiment where the call is to be invoked. Once a PackageCall is added to the experiment, you can double click on the Object to configure its properties. Minimally, you must specify which Package and Routine need to be called and what parameters need to be sent during the call.
The PackageCall Properties includes a **Details** section which provides information regarding the purpose of the Routine. A description of any required or optional parameters that need to be specified

to properly use the Routine are also listed. When viewing the Details section, it is helpful to note that a **Remarks** section exists at the bottom of the window. The Remarks section offers further insight on how the PackageCall is used in the experiment.




**NOTE**: *In some cases, you may have to scroll down to view Remarks/additional parameters.*

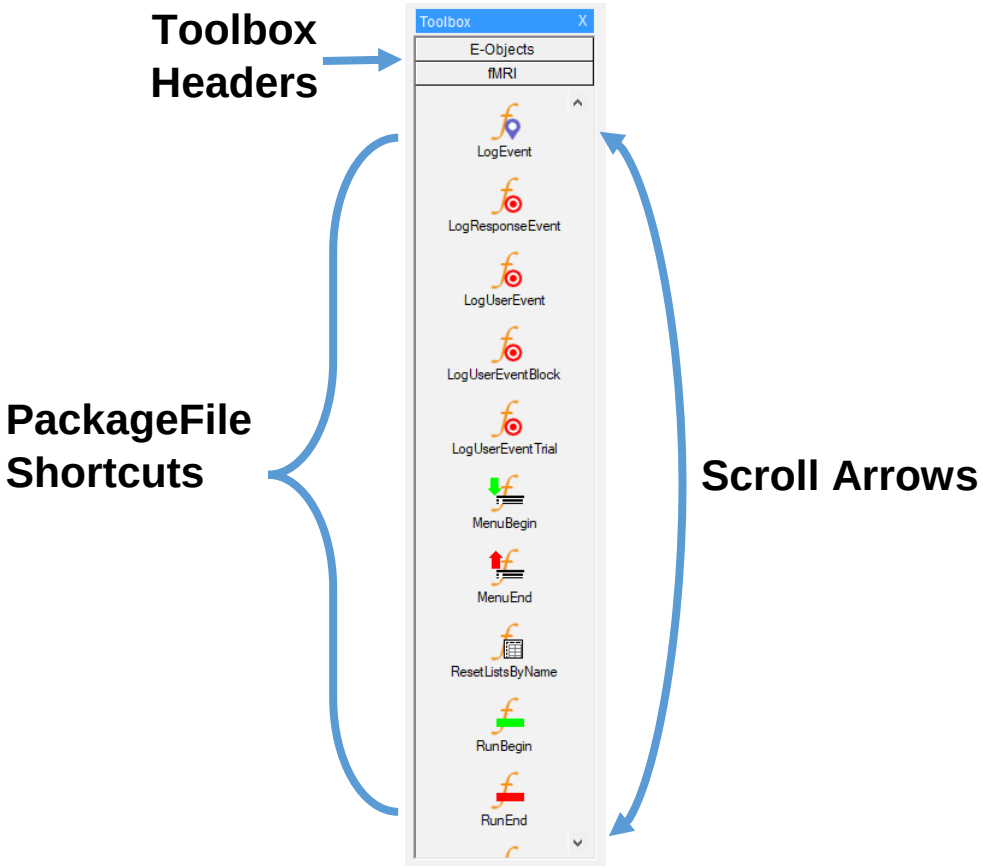
Once the Package and Routine properties of the PackageCall are set and the Properties dialog is closed, the icon associated with the PackageCall Object changes from the default icon to an author-specified icon. If there is no author-specified icon associated, then the default icon for the entire PackageFile is used. When dragging and manually configuring a PackageCall, it is strongly recommended that you rename the Object to reflect the specific PackageFile and Routines which are being referenced within the Object. If you do not follow this recommendation, E-Prime issues the following warning:



With Routines in the fMRI PackageFile, it is a common convention to create a name for each object by concatenating the name of the Package File ("fMRI") and the Routine being called. For example, the "fMRI_SessionInit" routine could be renamed "fMRISessionInit".

***PackageFile Toolbox Shortcuts***

A PackageFile author can optionally choose to provide a customized Toolbox of PackageFile Shortcuts to make it faster and easier for the experiment author to access and configure the Routines supported by the PackageFile. If a custom Toolbox is included with the PackageFile (and EEfMRI does include one), then a new panel is added to the Toolbox in E-Studio after the PackageFile is added to the experiment. Each panel of the Toolbox includes a clickable header that indicates which PackageFile is associated with the panel. You can click on the header to show the contents of the Toolbox and then drag a Shortcut from the Toolbox and drop it in a Procedure where the call is to be invoked. Scroll Arrows are present on the Toolbox for scrolling through the PackageFile Shortcuts.




Each PackageFile Shortcut contained in the Toolbox is simply a PackageCall Object (as outlined above) which has been predefined by the PackageFile author to save the experimenter time in renaming each PackageCall and configuring its required Properties. It is important to note that PackageFile Shortcuts that have different names in the Toolbox may be configured to reference the same PackageFile routine (e.g., properties may only differ by the list of default parameters that are specified). Providing multiple PackageFile Shortcuts for the same Routine can be useful when a

single common PackageFile routine is used in different scenarios with the experiment. PackageFile Shortcuts are the primary programming method used for this tutorial.

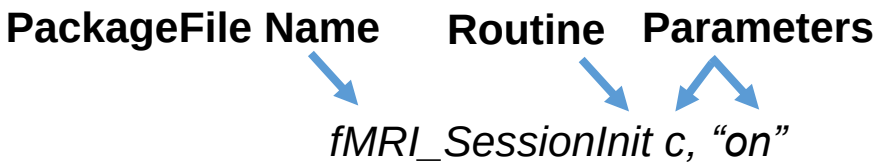
***InLine Script Method***

An **InLine** is an object used to insert user-defined script into an experiment at a specific point. An InLine Object is placed at the desired location in the Experiment Structure. The InLine Script 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, we do not typically recommend this calling method.

Calling a PackageFile routine from script is generally the same as calling any other E-Basic Subroutine or Function that could be defined in User Script. However, the definition of the Routine is contained with the PackageFile rather than within the User Script section of your experiment.

Due to a common naming convention used by PackageFile authors, a Package Routine can be invoked directly from an InLine Object. This is done by concatenating the name of the Package with an underscore ("_") and then the name of the Routine you want to call. Lastly, any parameters are added that are required for the call.

For example, the fMRI PackageFile includes a call to initialize an experiment. The Routine is named "SessionInit" and takes the Experiment Context ("c") and the requested state of EEfMRI as the required parameters. To call this Routine from an InLine Script Object, specify the following in the InLine Object:




Please view the article mentioned below to see further examples of PackageCall parameters and functions invoked by calling the Routine from an InLine.

Article 26208: Using PackageCalls via InLine Script

For specific information on PackageCalls in E-Prime Extensions for fMRI 3.0, please refer to:

E-Prime Command Reference

## 3.3. Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment

***Summary:***

Incorporating EEfMRI support into an existing E-Prime experiment primarily involves adding the fMRI PackageFile to the experiment and placing the EEfMRI PackageCalls at the appropriate locations within the Experiment Structure.

During this tutorial, you will add EEfMRI support to the Press.es3 experiment that is installed with EEfMRI. The Press.es3 experiment consists of multiple blocks where the participant is either at rest watching a fixation cross or is using specified fingers on each hand to press buttons and make responses. Response feedback is provided at the end of each trial. If you are not familiar with the Press.es3 experiment, we recommend loading, reviewing, and running the Sample experiment before beginning the Tutorial to develop an understanding of the overall design of the experiment and how it operates prior to further modifications.

**NOTE:** *All the Tutorials build upon each other and must be completed in order. The Sample experiment found in …My Experiments\3.0\fMRI\Samples\fMRIPress.es3 represents the complete set of changes in Tutorials 1 through 5.*

**NOTE**: *While many errors in your EEfMRI-enabled experiment can be detected while testing locally, some errors can only be detected or generated when using the scanner.*

***Goal****:*

This Tutorial illustrates how to add the EEfMRI PackageCalls into the Press.es3 experiment included with E-Prime Extensions for fMRI 3.0. After this tutorial, a basic EEfMRI-enabled paradigm is created.

***Overview of Tasks:***

- Load Press.es3 and resave it as fMRIPress.es3.
- Add the fMRI PackageFile to the experiment.
- Add the fMRISessionInit PackageCall to initialize the EEfMRI package at the Session level.
- Add the fMRIRunBegin PackageCall to designate the beginning of the functional scan.
- Add the fMRIRunEnd PackageCall to designate the end of the functional scan.
- Generate and run the experiment to verify functionality.

***Recommended readings:***

Before beginning the E-Prime Extensions for fMRI Tutorial, we recommend that you have read and worked through the other Tutorials provided with E-Prime. Please refer to the following for other available Tutorials:

Article 23282: Experiment Design (Beginner)
Article 22741: Experiment Design (Advanced)

It is important to understand critical timing in E-Prime to run experiments under the best possible conditions for accurate timing data. Refer to the following articles on timing:

Article 27520: E-Prime Timing Test Results

Article 22852: Timing of E-Objects
Article 22857: Stimulus Preparation Solutions

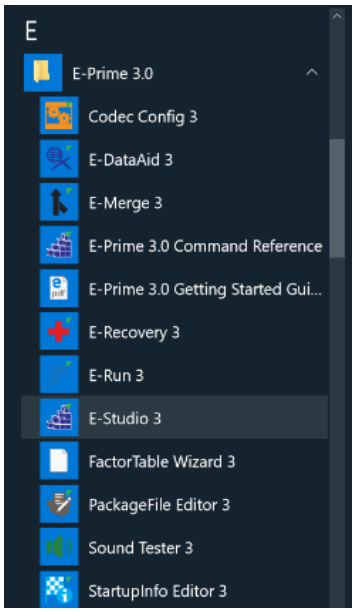
***Estimated Tutorial Time:***
20-30 minutes

# Task 1: Open the Press.es3 Experiment in E-Studio

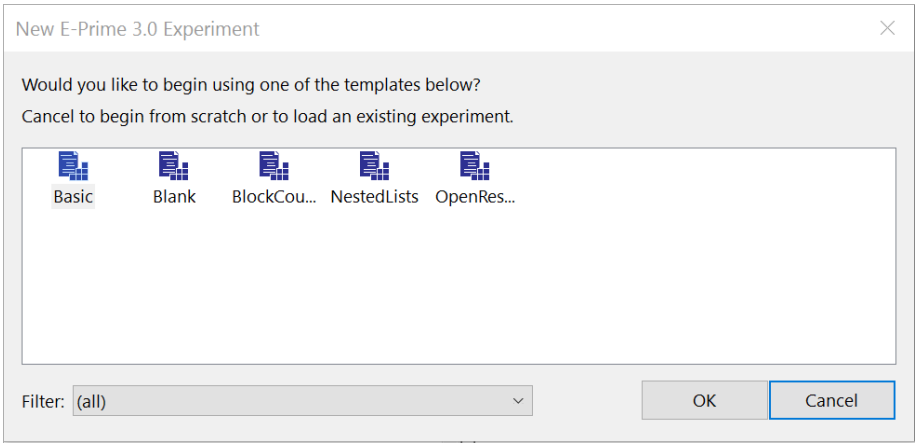
*Locate the E-Studio icon in the Start > All Programs > E-Prime 3.0 menu and launch the application by selecting it. Load the Press.es3 experiment.*

The E-Studio application is installed as part of the typical E-Prime installation. This application is used to create, modify, and test experiments within E-Prime. Open the E-Studio application, navigate to …My Experiments\3.0\fMRI\Tutorials and load the Press.es3 experiment.

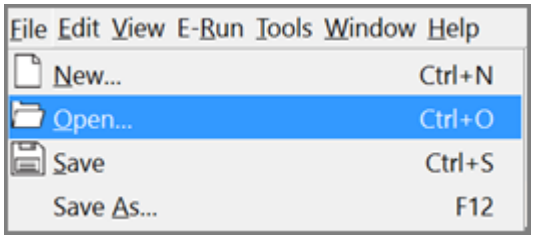
1) ***Click*** on the Windows **Start** menu, ***search*** for **E-Prime 3.0**. From the menu, ***click*** on **E-Studio** to launch the application.



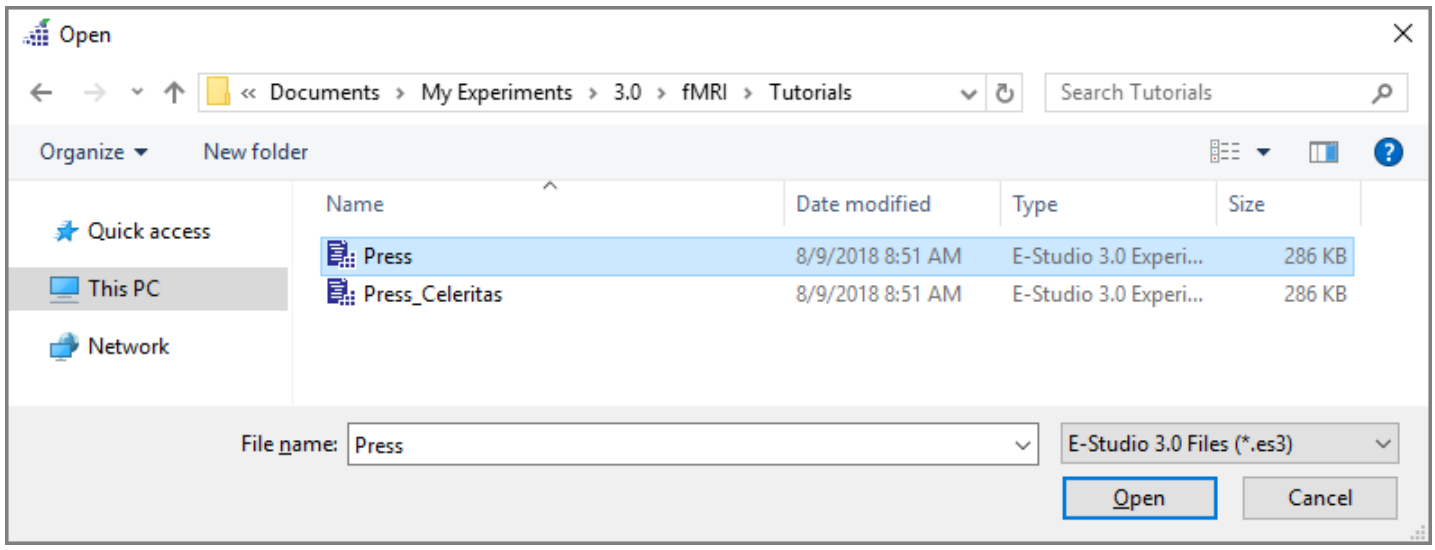

2) ***Click*** the **Cancel** button on the New E-Prime 3.0 Experiment window.




3) ***Select*** **Open** from the **File** menu**.** ***Navigate*** to the **…My Experiments\3.0\fMRI\Tutorials** folder to load the experiment.




4) ***Select*** the **"Press.es3"** file and then ***click*** the **Open** button to load the experiment into E-Studio.



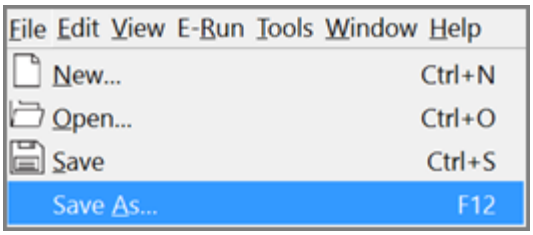

**NOTE**: *If Samples and Tutorials for E-Prime Extensions for fMRI 3.0 do not exist, see Troubleshooting, 4.1: Cannot locate E-Prime Samples and Tutorials.*

# Task 2: Save the experiment under a new name

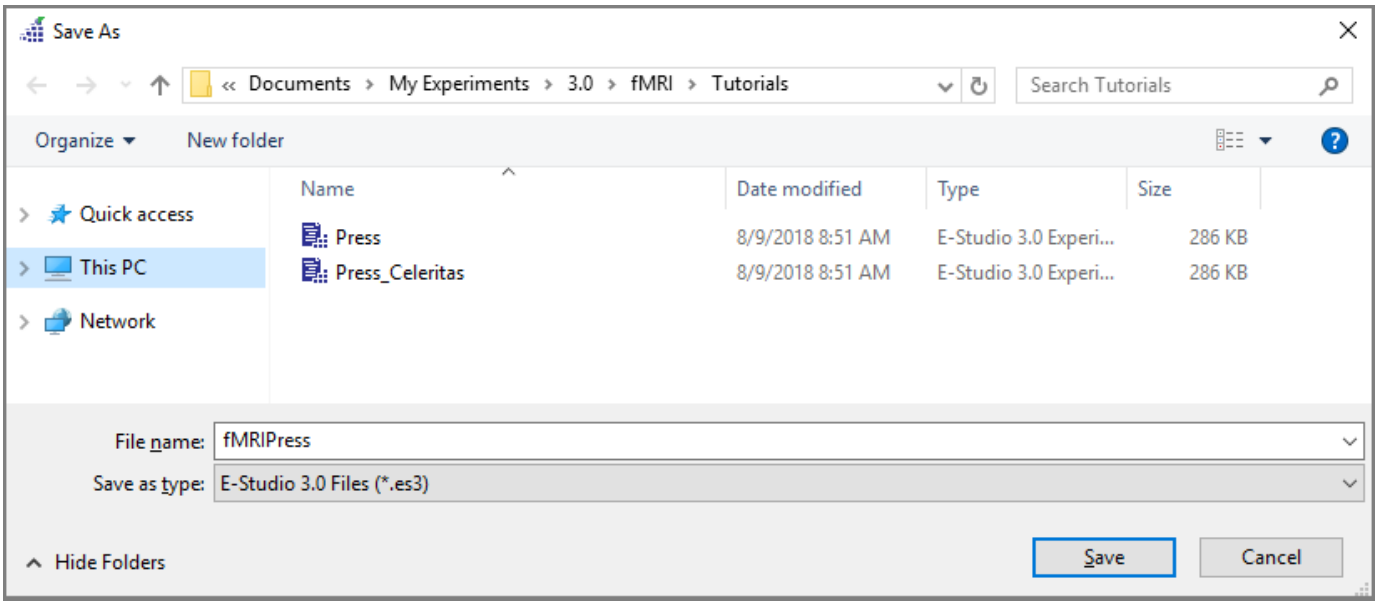
*Save the Press.es3 experiment in the same folder under the new name "fMRIPress.es3".*

Rename the experiment but be sure to save it in the same folder (…My Experiments\3.0\fMRI\ Tutorials) so that any resources within the experiment remain valid and can be reused.

1) ***Select* Save As...** from the **File** menu**.**




2) ***Type*** "**fMRIPress.es3**" as the new name in the **File name** field. ***Click*** the **Save** button.

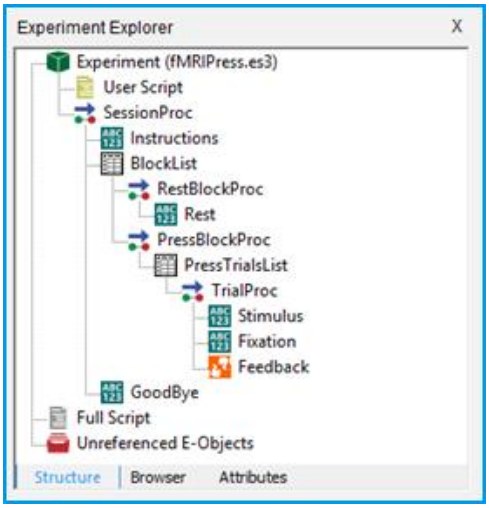


# Task 3: Add the fMRI PackageFile to the Experiment

*Open the Properties dialog for the Experiment Object and use the Packages tab to add the fMRI PackageFile to the experiment.*

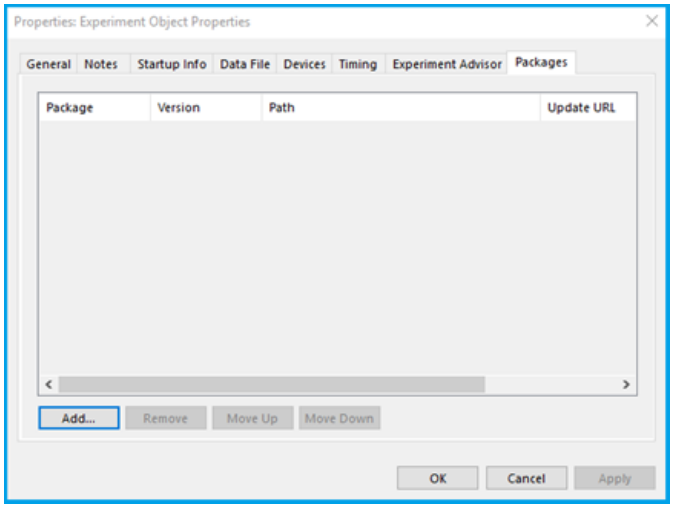
**PackageFiles** in E-Prime are cohesive sets of E-Basic routines that are grouped together. To gain access to the Routines within a PackageFile you must first add the PackageFile to the experiment. PackageFiles can be added to an experiment using the Packages tab of the Experiment Object Properties dialog. The Routines that are used to communicate with the EEfMRI hardware/software at runtime are contained within the fMRI PackageFile. The PackageFile also includes a customized Toolbox which contains shortcuts that you can drag into Procedure Objects within your experiment to invoke Routines contained in the PackageFile.

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

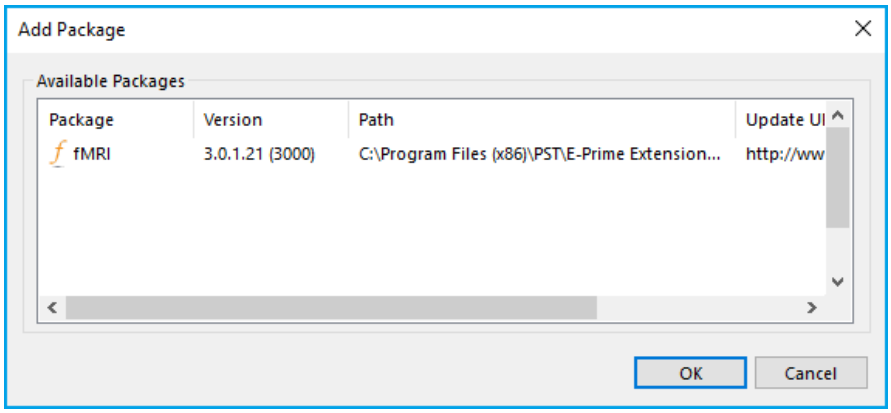



2) ***Click*** on the **Packages** tab of the **Experiment Object Properties** dialog.

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

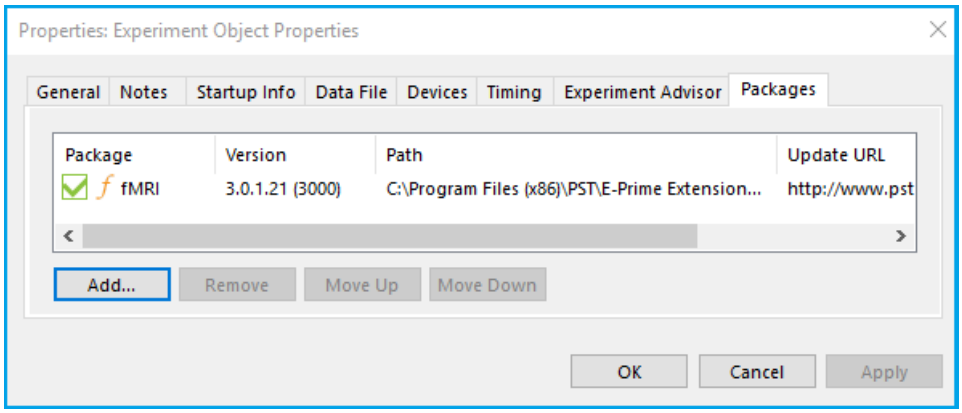


4) ***Select*** the **fMRI PackageFile** in the **Add Package** dialog.




5) ***Click*** the **OK** button to accept the dialog.

6) ***Verify*** the **fMRI PackageFile** has been added and is checked.




7) ***Click*** the **OK** button to accept and dismiss the dialog.

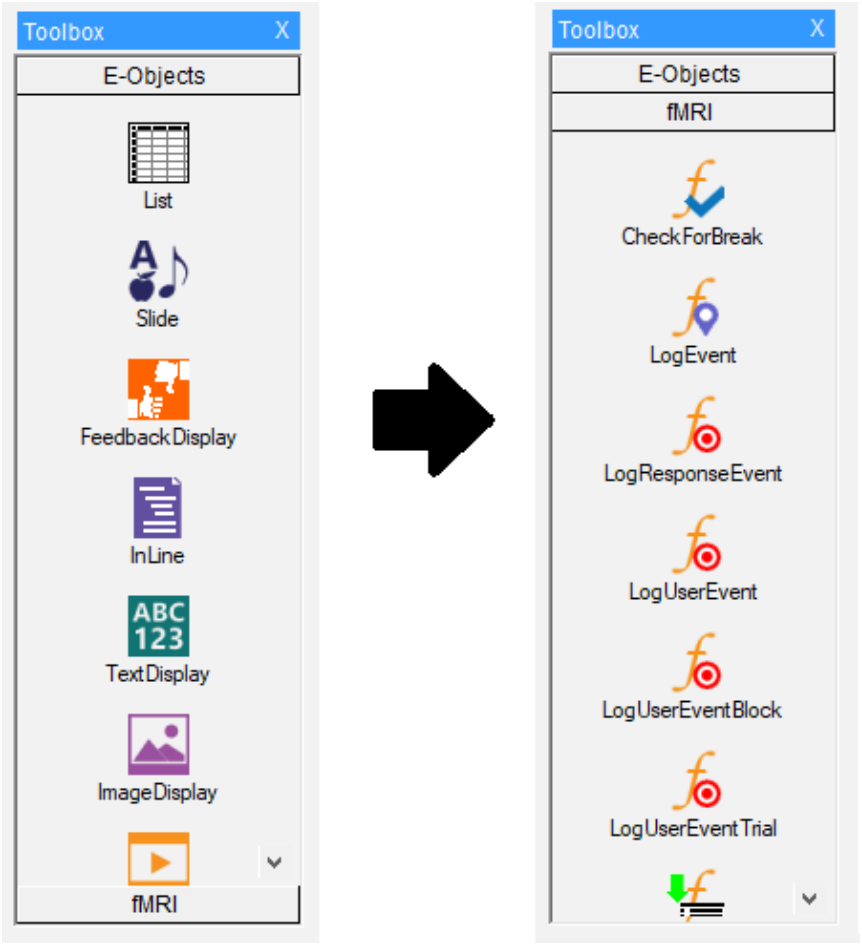
**NOTE:** *The PackageFile version number that is displayed by E-Studio reflects the version of the fMRI PackageFile that is currently installed on your machine and may not match the picture.*

# Task 4: Open the fMRI Toolbox to use the fMRI PackageCall Shortcuts

*Click the fMRI heading in the Toolbox to show the fMRI PackageCall Shortcuts.*

E-Prime Extensions for fMRI 3.0 utilizes pre-configured PackageCalls for fMRI communication. After the fMRI PackageFile is added to the experiment, these PackageCalls can be accessed from the Toolbox (i.e., the fMRI heading exists in the experiment Toolbox). The pre-configured **PackageCall Shortcuts** can be dragged and dropped to the appropriate Procedure without any further editing.

1) ***Click*** the **fMRI heading** in the **Toolbox** to display the fMRI PackageCalls.




**NOTE**: *To view the E-Objects again, click the E-Objects heading.*

# Task 5: Add the fMRI PackageCall to Initialize the System

*Add a PackageCall at the beginning of the SessionProc to initialize the system for use with fMRI.*

The fMRI PackageCall **fMRISessionInit** initializes EEfMRI at the start of the experiment. It is preconfigured to call the fMRI_SessionInit routine. The fMRI PackageCall is added at the beginning of the SessionProc for this purpose.

The first parameter (**c**) is common to most PackageCall routines and simply passes in the current Experiment Context.

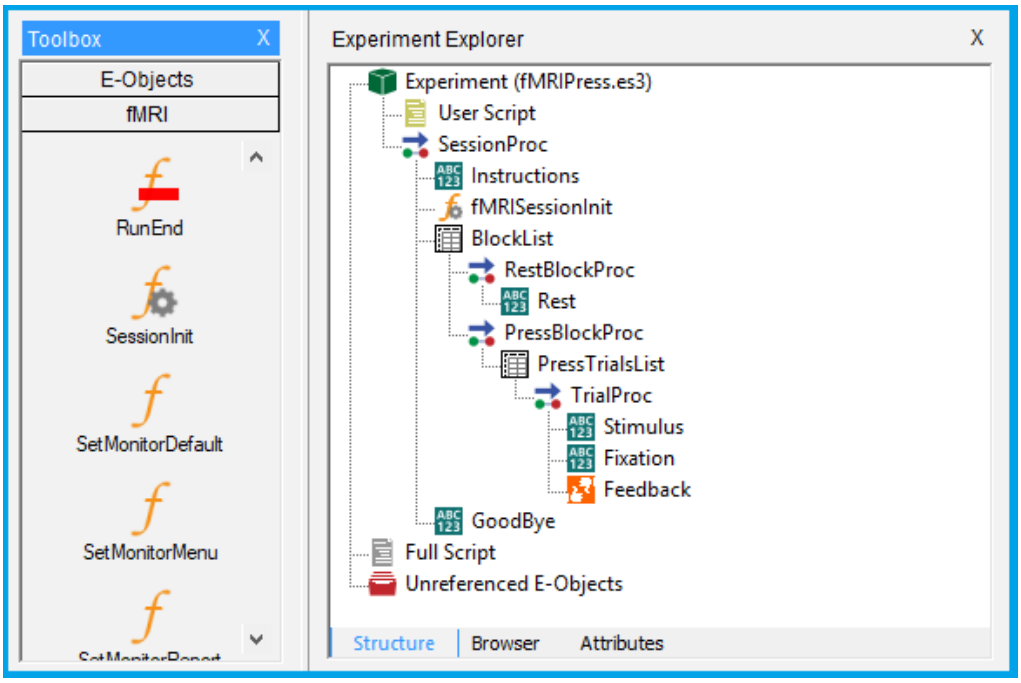
The second parameter (**strState**) is a required parameter. This parameter controls the state of the EEfMRI Package at runtime. This parameter can either be specified as “on” or “off”. When “on” is specified, EEfMRI components and routines are enabled. When “off” is specified, EEfMRI functionality is disabled and the routines are not run. For information on how to run an fMRI-enabled experiment without the hardware connected, please refer to *Running an Experiment Without EEfMRI Enabled.*

The third parameter (**vPDATFilename**) is optional. It allows users to specify the filename for the .pdat file. An empty string (i.e., "") can be specified to disable the creation of the .pdat file. The. pdat file is created in the same folder as the experiment (…My Experiments\3.0\fMRI\Tutorials) after the experiment is completed and has the same name as the experiment’s E-Prime data file (e.g., assuming you entered the default startup information for this tutorial, the file would be named “fMRIPress-1-1.pdat”). The file can be opened via Excel or any other application that can load a tab delimited text file. The information contained in the .pdat file can be used as a reference to assist the experimenter in post-processing and image analysis activities.

The fourth parameter (**vPDATUserColumns**) is an optional parameter. This parameter allows specification of the tab delimited string column names for user defined entries in the .pdat file. When this parameter is not specified, then the default column names are "User1\tUser2\tUser3\tUser4\tUser5". Specifying an empty string (i.e., "") disables the user defined column headings (i.e., no user column headings are output).

The final parameter (**vHandlePreRelease**) is an optional parameter that specifies if the HandlePreRelease routine is called prior to the execution of this Routine's script. If not specified, this defaults to True.

1) ***Drag*** the shortcut **fMRISessionInit** from the **EEfMRI Toolbox** and ***drop*** it in the **SessionProc** procedure after the **Instructions**.



## Task 6: Add the fMRIRunBegin PackageCall

*Add a PackageCall to identify the beginning of the functional scanning run and the start of the task stimulus presentation sequence.*

The **fMRIRunBegin** PackageCall marks the beginning of the functional scan and synchronizes the onset of the stimulus presentation with the scanner trigger pulse (assuming appropriate interface hardware is in use). This call writes critical timing information to the .pdat file and must be matched with a corresponding fMRIRunEnd PackageCall for proper operations.

The first parameter (**c**) is the required Experiment Context.

The second parameter (**nTriggerTime**) is required and is a millisecond timestamp that associates with the trigger. If the numerical value of -1 is specified, then this call waits until one of the designated trigger keys is received from the Keyboard device before returning. If a non-zero is specified, then the nTriggerTIme value is used directly as the trigger time and the system does not wait for a trigger.

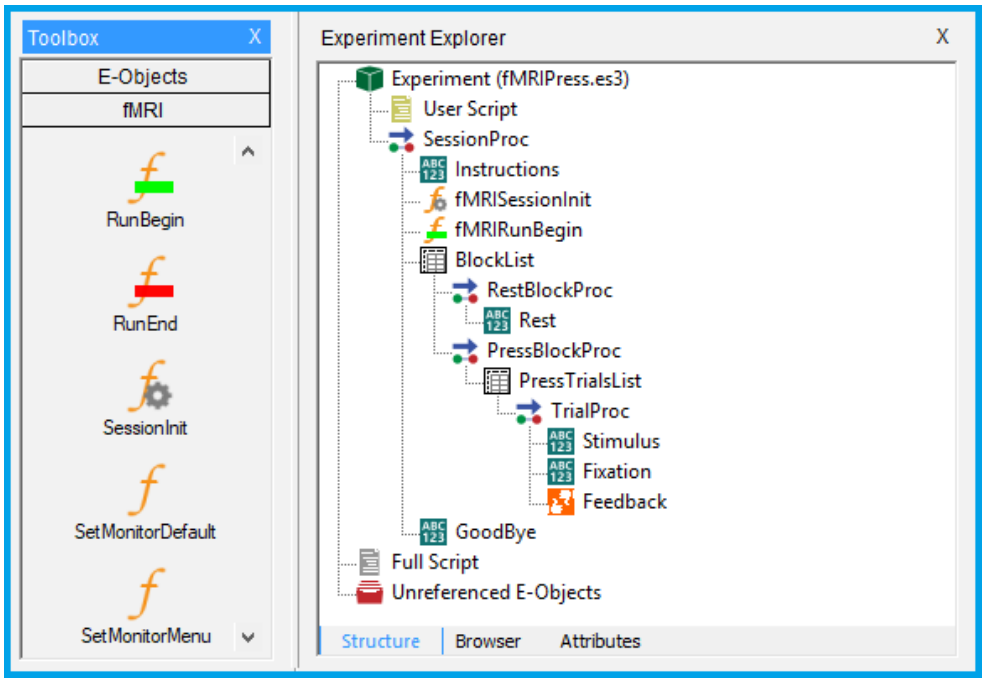
The third parameter (**strConditionId**) is required. This parameter is a string value that identifies the current experimental run condition (e.g., "RunCondition=[MenuItem]").

The fourth parameter (**vTriggerMessage**) is optional. This is a string message that is overlaid on the screen when waiting for the trigger (e.g., "Please wait"). An empty string ("") can be specified to disable the message.

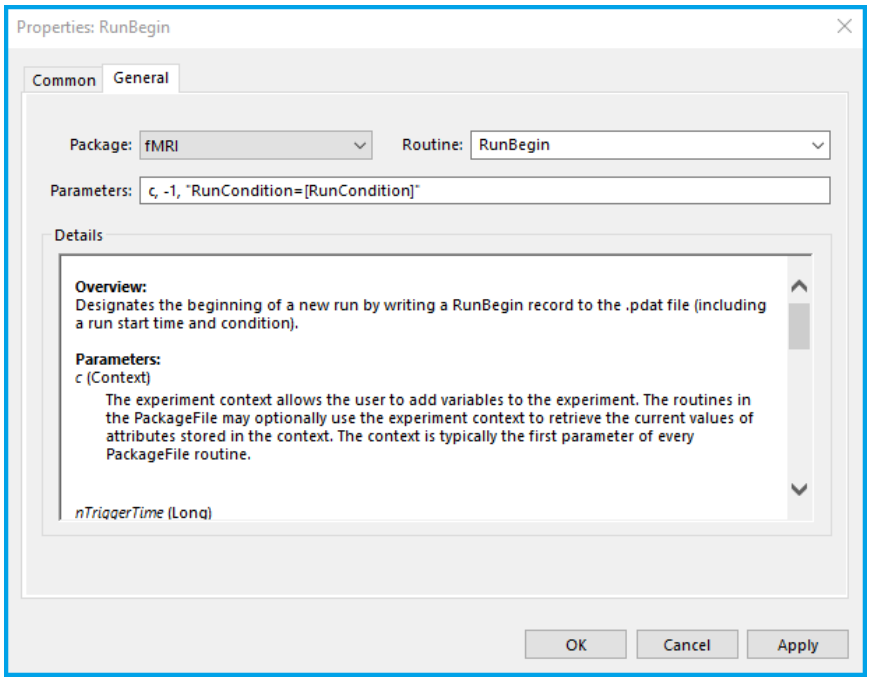
The fifth parameter (**vTriggerKeys**) refers to an optional set of keys that can be accepted as the trigger. By default, the allowable keys are "{=}", "{^}", "{ENTER}", and "{ESCAPE}".

The final parameter (**vHandlePreRelease**) is an optional parameter that specifies if the HandlePreRelease routine is called prior to the execution of this Routine's script. If not specified, this defaults to True.

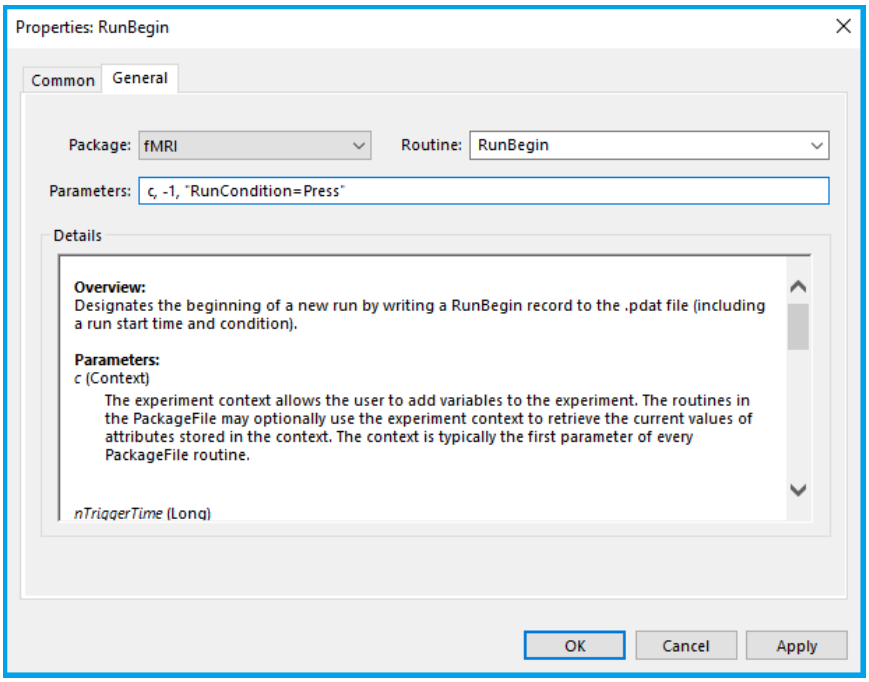
1) ***Drag*** the shortcut **fMRIRunBegin** from the **fMRI Toolbox** and ***drop*** it in the **SessionProc** procedure after the **Instructions** and the **fMRI SessionInit** shortcut but before **BlockList**.




2) ***Double click*** on the **fMRIRunBegin** Object to open its **Properties** dialog.



3) ***Edit*** the **Parameters** field.




**NOTE**: *The recommended parameters are c, -1, "RunCondition=Press". These parameters will be changed in Tutorial 5: Task 4: Modify the fMRIRunBegin PackageCall. Because there is only 1 task in the experiment now, the strConditionId can refer to the Press Trials. Tutorial 5 implements an attribute for this parameter to differentiate Press from Practice Trials.*

4) ***Click*** the **OK** button to accept and dismiss the dialog.

# Task 7: Add the fMRIRunEnd PackageCall

*Add a PackageCall to identify the end of the functional scanning run.*

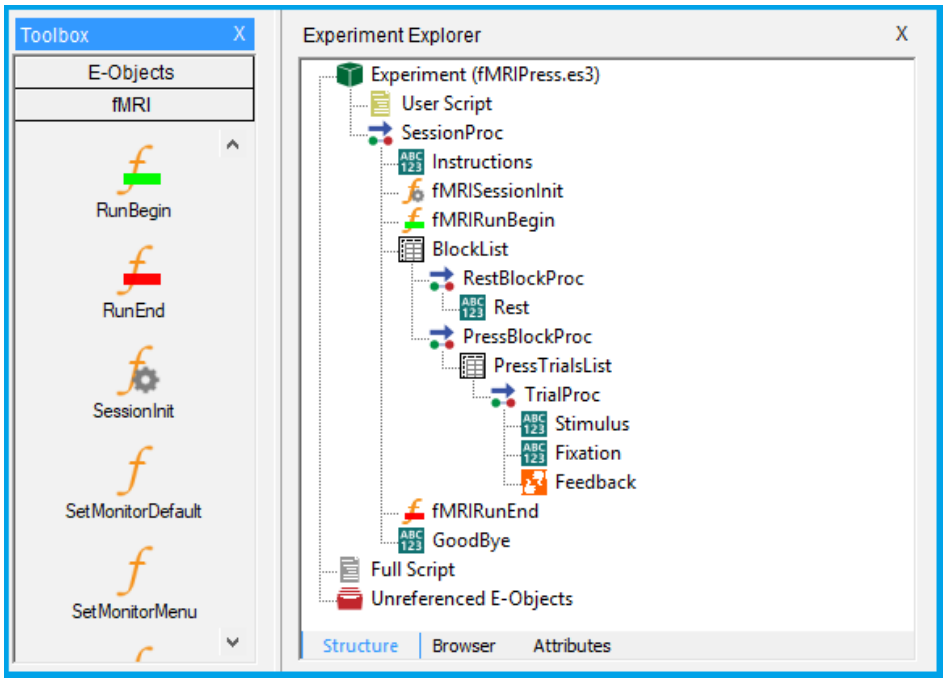
The **fMRIRunEnd** PackageCall is used to mark the end of the functional scanning run. By default, the Routine displays a Run Report screen when the run has completed (or has been terminated prematurely by the experimenter). The Run Report summary screen contains useful information about the status and observed duration of the run. This PackageCall is typically placed immediately after the Object that completes the stimulus presentation sequence.

The first parameter (**c**) is the required Experiment Context.

The second parameter (**nReportDuration**) is required. This is a numerical value that specifies how long the report should display in milliseconds (e.g., 2000). A value of 0 disables the report and does not show it. A value of -1 leaves the report displayed until the "{ENTER}" key is pressed.

The final parameter (**vHandlePreRelease**) is an optional parameter that specifies if the HandlePreRelease routine is called prior to the execution of this Routine's script. If not specified, this defaults to True.

1) ***Drag*** the shortcut **fMRIRunEnd** from the **EEfMRI Toolbox** and ***drop*** it in the **SessionProc** procedure after the **BlockList** and before the **Goodbye Object**.



# Task 8: Run the Experiment

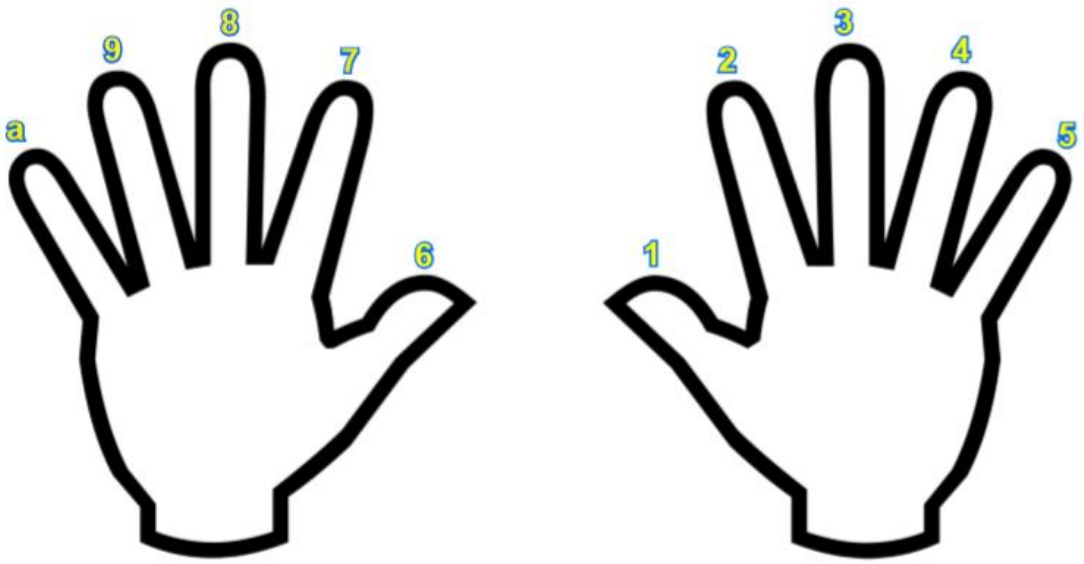
*Run the experiment verifying the appropriate prompts are presented and no runtime errors are generated.*

**Preparing to Run the Experiment**

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

Before you begin testing the experimental task, it is important that you understand how the task works and how the experiment is designed to accept responses. Once participants start the experiment, they are prompted to use a specific finger to press the button lying beneath it. For example, when "Left Middle" appears on the screen, the participant should press the button under his or her left middle finger.

While the EEfMRI Tutorial experiments accept responses from the computer keyboard, they are designed for use with other response systems (see *Appendix D: Add Celeritas to the Experiment*). These other systems are MR-compatible hardware systems that include custom molded button units that can be worn by the participant, as well as the interface components necessary to report the scanner trigger pulse to E-Prime for stimulus synchronization. More specifically, the experiment assumes that the participant has two Button Response Units (BRUs), one for each hand. Each BRU has five buttons, which enables a unique response to be sent for each finger. By default, the BRU connected on the right maps to the character keys 1 – 5. The first four buttons on the BRU connected on the left (for the thumb through ring finger) map to the character keys 6 – 9; the button under the left pinky maps to the character a. These mappings are illustrated in the figure below.



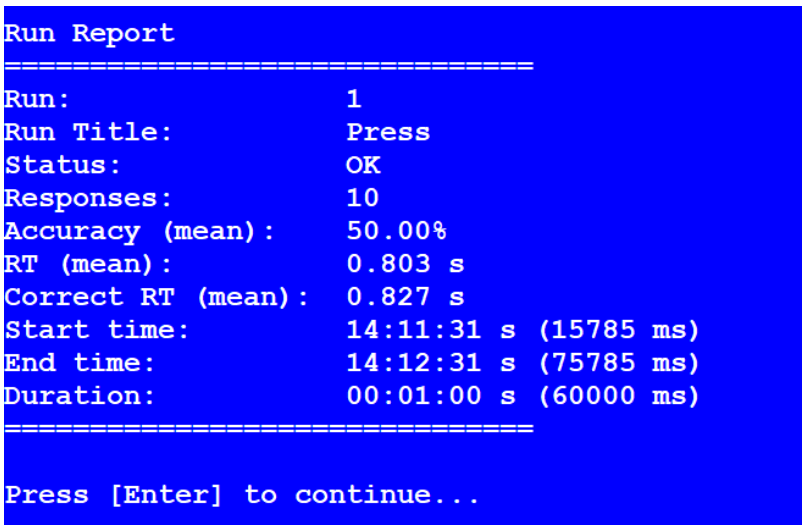

These default key mappings allow you to run your experiment at the scanner without modifying the response mappings. However, these mappings make experimental testing with a computer keyboard

cumbersome if the tester wants to make correct responses. The right hand needs to be positioned over the keys 1 – 5, which is readily accomplished. The left hand is difficult to position because the correct response for the left thumb is the 6 key; the correct response for the left index finger is the 7 key, and so on.

**NOTE**: *If you would like to add a different response device to the experiment (e.g., Celeritas), please refer to Appendix D: Add Celeritas to the Experiment.*

**The Run Report**

After running the experiment, the Run Report displays to the experimenter. The image below shows the **Run Report** which summarizes information about the experimental run that was completed. The report contains some basic timing information as well as several indicators of the participant's performance. It is often helpful to have this information available immediately at the end of a run to quickly verify that the run executed as expected and that the participant performed the task adequately (e.g., to assist the experimenter in determining if the run must be repeated).




The Start Time, End Time, and Duration fields provide the computer time at which the scanner trigger pulse was detected/simulated, the computer time the task ended, and the duration of the task.

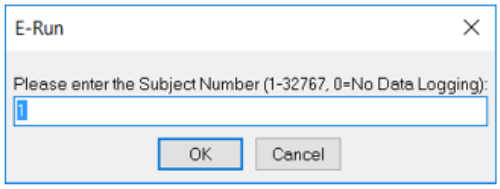
The Accuracy field is the participant's percent accuracy. The RT field is the participant's mean response time across all trials during a specific task in an experiment. The Correct RT field is the mean response time for the correct answers only. In the example above, the fields related to dependent measures are present. In the case the dependent measures were not collected, they are displayed as zero.

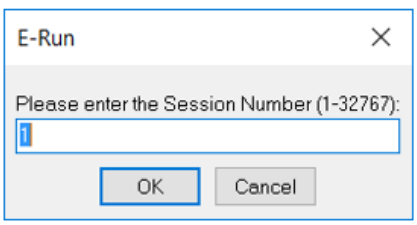
**NOTE**: *This Tutorial did not yet add the EEfMRI PackageCalls required to collect the dependent measures. After these additional PackageCalls have been added you may see these fields as zeros (0) if you do not make any responses during the task.*

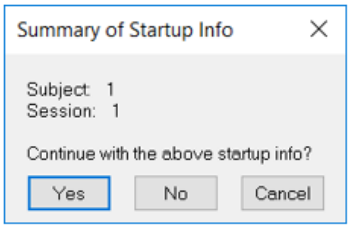
**NOTE**: *To change the amount of time the Run Report displays, refer to Task 7: Add the fMRIRunEnd PackageCall.*

1) ***Click*** the **Run icon** on the toolbar to generate and run the experiment locally.

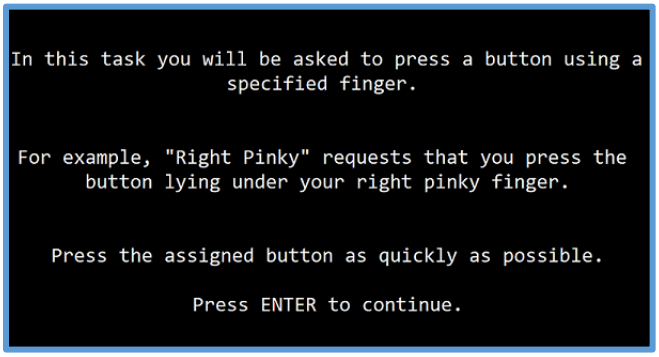
2) ***Press*** **Enter** to ***accept*** the **default values** for each of the initialization prompts presented.



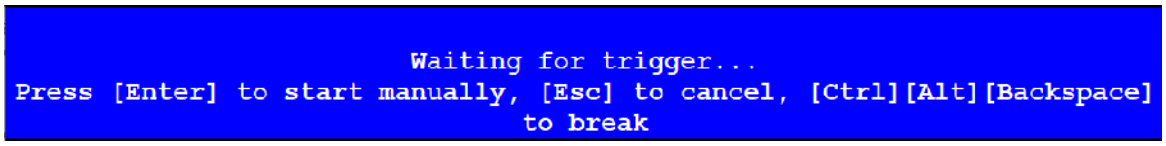




3) ***Read*** the experiment instructions then ***press*** **Enter** to continue.

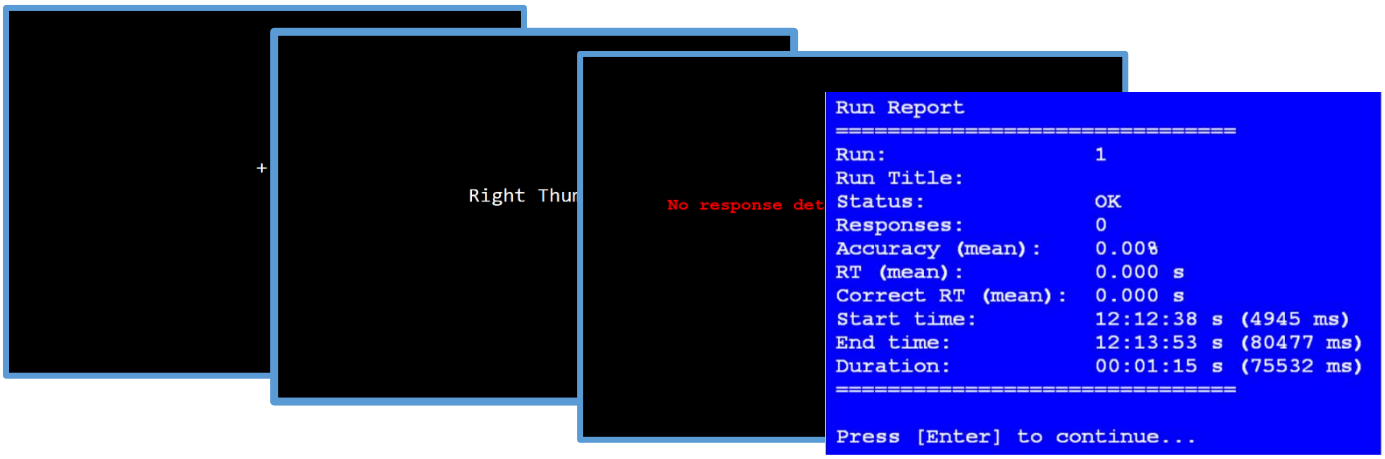



4) ***Press*** **Enter** to manually ***simulate*** the scanner trigger pulse and begin the task.




**NOTE**: *The "Wait for trigger" options appear after you press Enter on the Instructions screen.*

5) ***Verify*** the **experiment** is ***functioning*** correctly and the **Run Report *appears*** at the end of all the experimental trials.




**NOTE**: *EEfMRI 3.0 provides support for running with multiple monitors. The Menu, Run Report, and/or any runtime error message boxes are configured by default to be displayed on the first initialized display device. See the following article for information on specifying a secondary monitor for the non-stimulus experiment components:*

Article 28046: Multiple Monitor Support in EEfMRI 3.0

## 3.4. Tutorial 2: Creating a Menu

***Summary:***

This Tutorial guides you through the steps necessary to add a menu to the paradigm that was EEfMRI-enabled in the previous tutorial. It also addresses a practical concern that may arise during fMRI data collection. During an experiment it may be necessary to stop and restart the scanner (e.g., due to requests for assistance from the participant, participant movement, problems with the scanner or scanning protocol, etc.). When these events occur, it is advantageous for the experimenter to quickly interrupt the experiment, reset the stimulus presentation sequence, and restart the task from a point in the experiment (e.g., typically the menu).

This Tutorial illustrates how to use fMRI PackageCalls to ensure that any stimuli presented prior to the interruption of the scan are properly "reset" to ensure they are shown again after you return to the menu and restart the experiment.

**NOTE:** *All the Tutorials build upon each other and must be completed in order. Tutorial 2 begins where Tutorial 1 ends. This Tutorial assumes that you have completed the previous Tutorial. You must load the experiment you edited in the last Tutorial into E-Studio before continuing.*

**NOTE:** *The Sample experiment found in the folder …My Experiments\3.0\fMRI\Samples\fMRIPress.es3 represents the complete set of changes documented in Tutorials 1 through 5.*

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

***Goal:***

When you have completed this tutorial, an experiment that allows you to choose a task to run from the menu list is created. This Tutorial also illustrates how to reset the stimulus presentation sequence and return to the menu to start data collection again.

***Overview of Tasks:***

- Add the fMRIMenuBegin PackageCall to implement the menu.
- Add the fMRIMenuEnd PackageCall to designate the end of the menu function.
- Add a RunList with the "MenuItem" attribute that populates the menu text.
- Create a PressRunProc to allow for multiple runs.
- Move E-Objects to change the structure for multiple runs.
- Add the fMRIResetListsByName PackageCall to the SessionProc to reset the trial stimuli after the experiment has been interrupted.
- Verify the overall experiment structure and run the experiment.

***Recommended readings:***
It is required that you have read and worked through and completed Tutorial 1 before beginning this tutorial.

Because this tutorial requires adding and editing an additional List Object, we recommend reviewing the following article on List Objects:

Article 22700: List Object

***Estimated Tutorial Time:***
10-15 minutes

# Task 1: Add the fMRIMenuBegin PackageCall

*Add a PackageCall to the SessionProc to direct EEfMRI to create a menu.*

The **fMRIMenuBegin** and **fMRIMenuEnd** PackageCalls are used to create an interactive menu within the experiment. This menu can be used by the experimenter to dynamically choose different tasks to present to the participant at runtime. The menu is created by processing a designated List Object defined within the experiment to extract the text for the items that appear on the menu.

The fMRIMenuBegin PackageCall is typically placed after the fMRISessionInit PackageCall, but prior to the fMRIRunBegin because it is not part of the functional run.

The first parameter (**c**) is the required Experiment Context.

The second parameter (**strMenuState**) is a required parameter. This parameter controls the display of the menu at runtime. This parameter can be specified as the string value “on” or “off”. When “on” is specified, the menu displays to the user and the Selection and Reset/Exit related properties specified on the List Object are ignored. When the menu is disabled, the List Object executes normally.

The third parameter (**theMenuList**) is a required parameter. This parameter requires the name of the List Object that is used to create the menu. The menu List Object must contain an attribute that specifies the text of each menu item that is displayed to the user. By default, the attribute is named "MenuItem", you may change this with the vMenuItemAttribute parameter (see below).

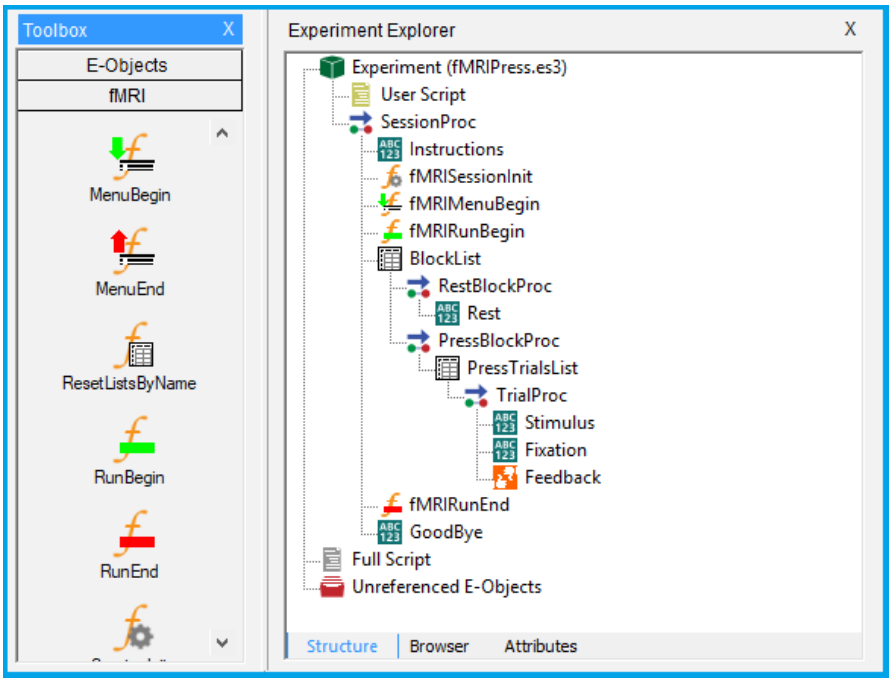
The fourth parameter (**vMenuItemAttribute**) is an optional parameter that is only specified when using an attribute name other than the default “MenuItem” on the List Object to define the menu item text.

The final parameter (**vHandlePreRelease**) is an optional parameter that specifies if the HandlePreRelease routine is called prior to the execution of this Routine's script. If not specified, this defaults to True.

**NOTE**: *The List Object identified by theMenuList parameter is not created until Task 3: Add the List Object that Populates the Menu.*

**NOTE**: *If it is not already opened, open the fMRIPress.es3 experiment you created in Tutorial 1.*

1) ***Drag*** the shortcut **fMRIMenuBegin** from the **fMRI Toolbox** and ***drop*** it in the **SessionProc** procedure after the **fMRISessionInit** shortcut



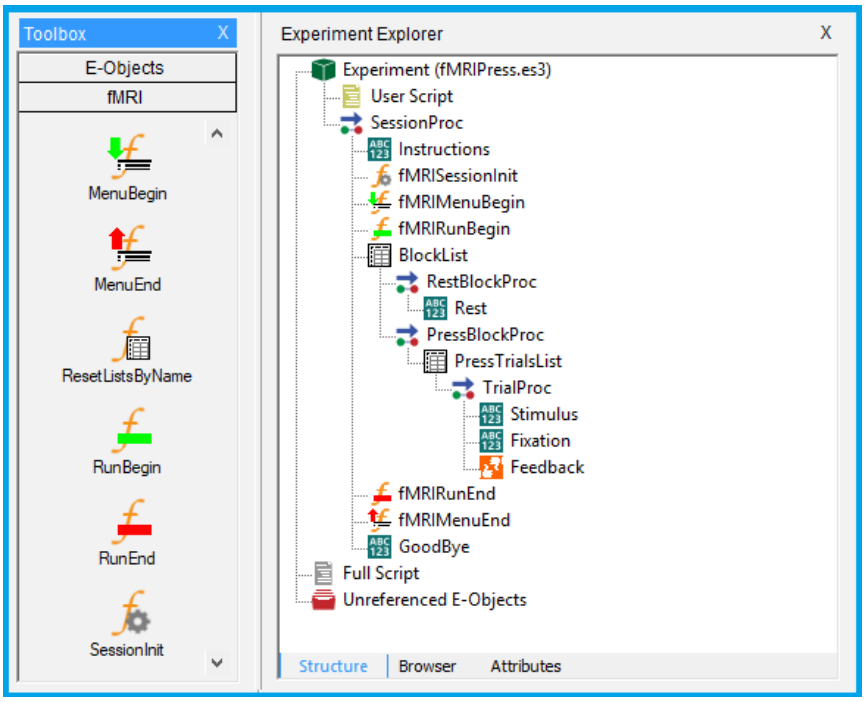

**NOTE**: *The recommended parameters are c, "on", RunList. The RunList is created and described in* *Task 3: Add the List Object that Populates the Menu*.

# Task 2: Add the fMRIMenuEnd

*Add a PackageCall to the SessionProc to complete the menu.*

The **fMRIMenuEnd** PackageCall completes the implementation of the fMRIMenuBegin PackageCall. The call is typically placed at the end of the experiment prior to any final screen that you want the participant to see, such as the “Goodbye” screen at the experiment end.

1) ***Drag*** the shortcut **fMRIMenuEnd** from the **EEfMRI Toolbox** and ***drop*** it before the **Goodbye** Object.



# Task 3: Add the List Object that Populates the Menu

*Add a List Object to the SessionProc.*

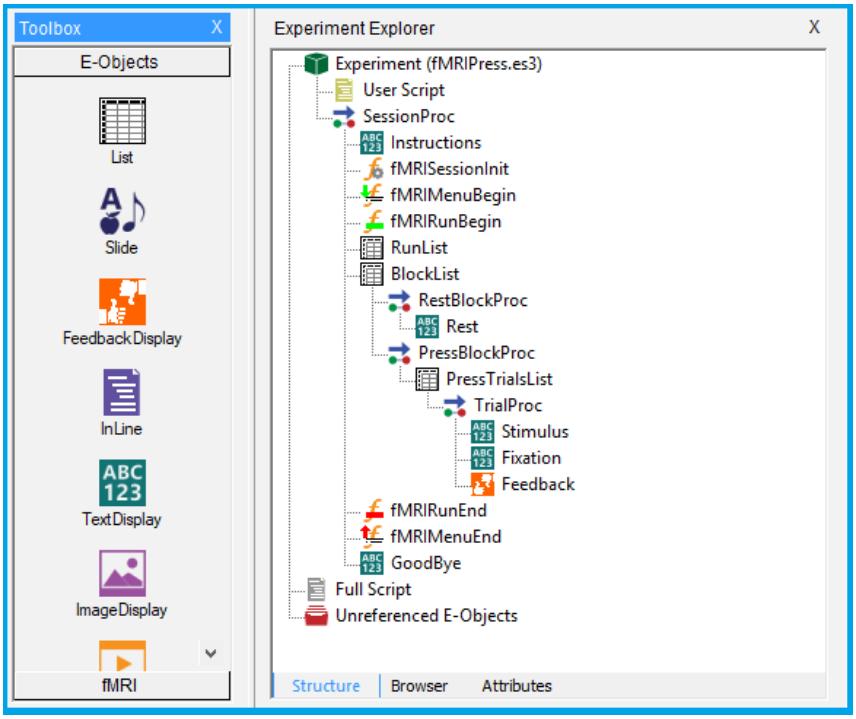
A **List** Object is used to define the entries that are seen by the experimenter when the menu is displayed. The Object also associates a Procedure with each menu entry. When the user chooses the menu item, EEfMRI runs the Procedure associated with the selected menu item. The experiment that we are using in this tutorial only has one type of task defined, but it is possible to customize the menu to display several different types of tasks for the experimenter to select. For example, you could include a practice version of each experimental task (see *Tutorial 5: Creating a Practice Run*).

**NOTE***: The name of the List Object created in this step must match the value of "theMenuList" parameter specified in the fMRIMenuBegin PackageCall (see Task 1: Add the fMRIMenuBegin PackageCall).*

1) ***Click*** the **E-Objects heading** in the **Toolbox** to display the E-Prime E-Objects.

2) ***Drag*** a new **List Object** from the **E-Object's Toolbox** and ***drop*** it after **fMRIMenuBegin**.

3) ***Rename*** the **List Object** as "**RunList**".

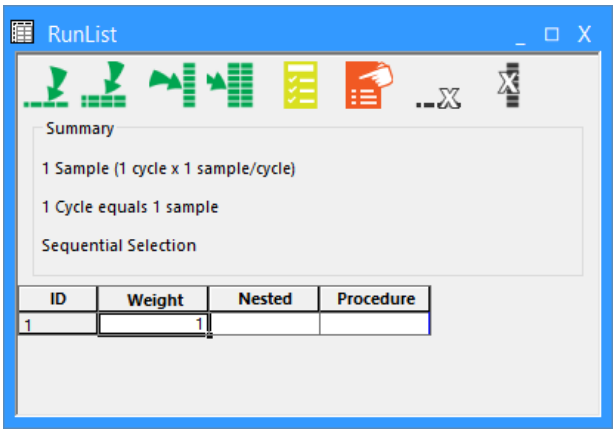


# Task 4: Enable the RunList Object to Populate the Menu

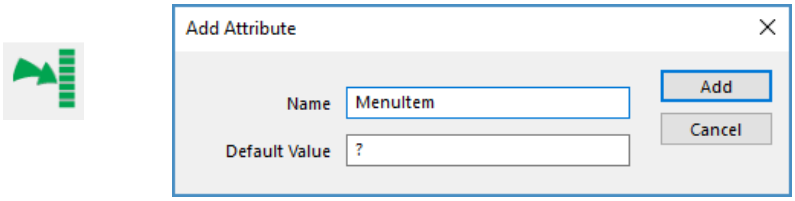
*Add the "MenuItem" attribute to the RunList and create the PressRunProc.*

When the fMRIMenuBegin PackageCall creates the menu of tasks at runtime, it retrieves the value of the selected **MenuItem** attribute and displays it as the text description of the task in the menu. Each MenuItem entry must be associated with a Procedure Object. When the experimenter selects an entry from the menu, fMRI runs the Procedure that is associated with the selected MenuItem.

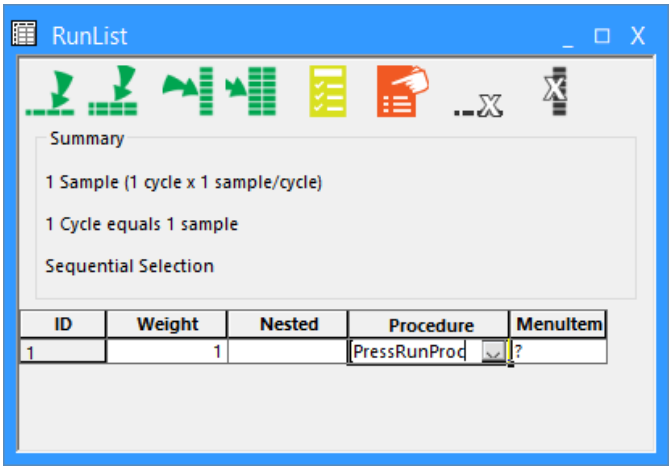
1) ***Double click*** the **RunList** Object to open it in the workspace.



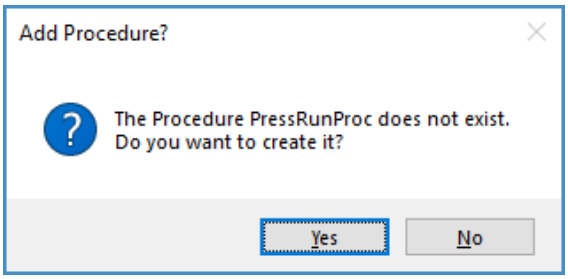

2) ***Open*** the **Add Attribute** dialog, ***name*** the **Attribute** "**MenuItem**" and ***click*** **Add.**



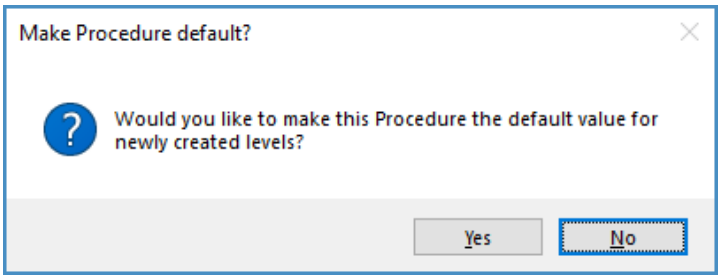

3) ***Edit*** the text cell in the **Procedure** column to read, "**PressRunProc**", and ***press*** **Enter** to accept the change.



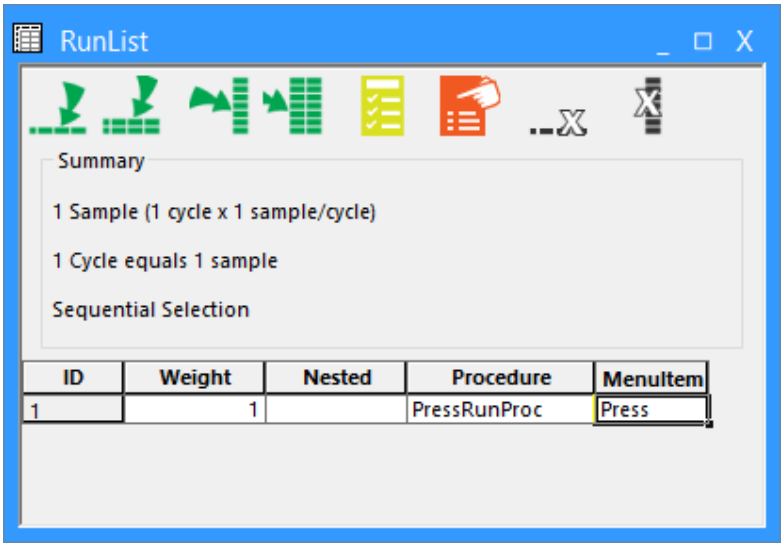
4) ***Click* Yes** on the **Add Procedure** dialog to create the new **PressRunProc Procedure** Object.




5) ***Click* No** when prompted to make the procedure default.




6) ***Edit*** the text cell in the **MenuItem** column to read "**Press**". ***Press* Enter** to accept the change.



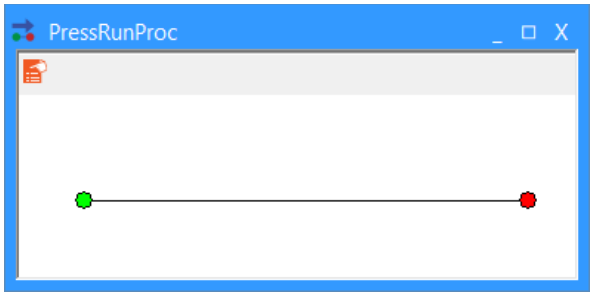
# Task 5: Move some E-Objects to the New Procedure

*Move the Instructions, fMRIRunBegin, fMRIRunEnd, and the BlockList Objects from the SessionProc to the new PressRunProc.*

Now that the new Procedure is created, you need to move the Instructions, fMRIRunBegin, fMRIRunEnd, and the BlockList Objects from their current location on the SessionProc procedure to the procedure that you created in the last step.

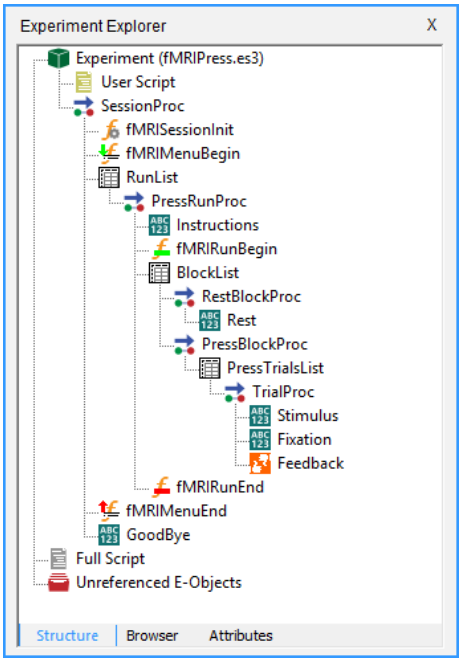
**NOTE**: *The E-Objects need moved, not copied. If you drag and drop without holding the **Shift key**, then the E-Objects are copied rather than moved.*

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




2) ***Drag*** and ***drop*** (while holding the Shift key) the **Instructions** Object from the **SessionProc**, onto the **PressRunProc**. ***Repeat*** this for the **fMRIRunBegin, BlockList,** and **fMRIRunEnd** Objects.

3) ***Verify*** your **Experiment Structure** matches the image below.



## Task 6: Add the fMRIResetListsByName PackageCall

*Add a PackageCall to the SessionProc that directs fMRI to reset and reuse the Stimulus items in a List.*

In experiments where a menu system is in use, it is likely you want to reset the stimuli in most List Objects before the next run begins. For the purposes of this manual, a run is defined by the start and subsequent stop of the scanner. Unless you take care to assure that stimulus Lists are properly reset, you may observe unexpected results. For example, if an experiment is interrupted then any stimuli that were presented prior to the interruption may not be reused again until the stimulus List is reset.

The proper use and placement of the **fMRIResetListsByName** PackageCall ensures that Stimulus Lists are reset consistently and behave as expected. Generally, we recommend that any List Object that controls the sequence of task blocks, trials, or stimuli should be reset if the experiment is interrupted. *However, the experimenter should decide how best to handle this situation on a task by task basis.*

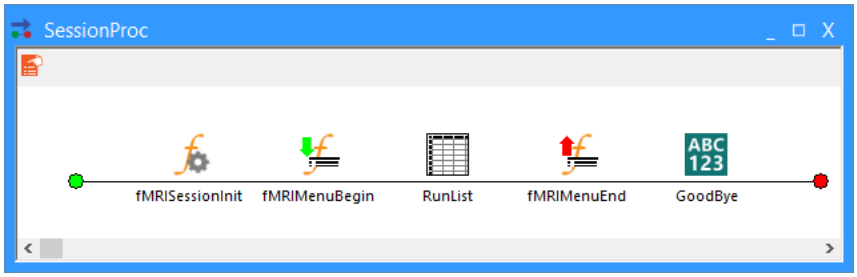
When running an EEfMRI-enabled experiment with MenuBegin and MenuEnd PackageCalls, the **Conditional Exit** key sequence returns the user to the Menu. This escape sequence is invoked by pressing Ctrl-Alt-Backspace simultaneously. Pressing this key sequence immediately ends the current trial and returns the user to the menu. Use of the **Conditional Exit** key sequence is recommended for use with EEfMRI-enabled experiments.

The **User Break State** key sequence can also be used to interrupt an EEfMRI-enabled experiment. This escape sequence is invoked by pressing Ctrl-Shift simultaneously. This method sets a user break state variable that is accessed via the GetUserBreakState() method. When GetUserBreakState() is used with advanced E-Basic scripting techniques, the currently executing List Objects can be terminated at the end of the current trial or block, and the accumulated data that had been collected to the current point of the experiment can be saved. Because the use of the GetUserBreakState() method requires additional E-Prime scripting, the fMRI PackageFile includes the CheckForBreak PackageCall to assist with this task.

**NOTE:** *Pressing Ctrl-Alt-Shift terminates an experiment, but it does not close out devices or create a data file. This termination method prompt is designed for use during initial experiment development and testing.*

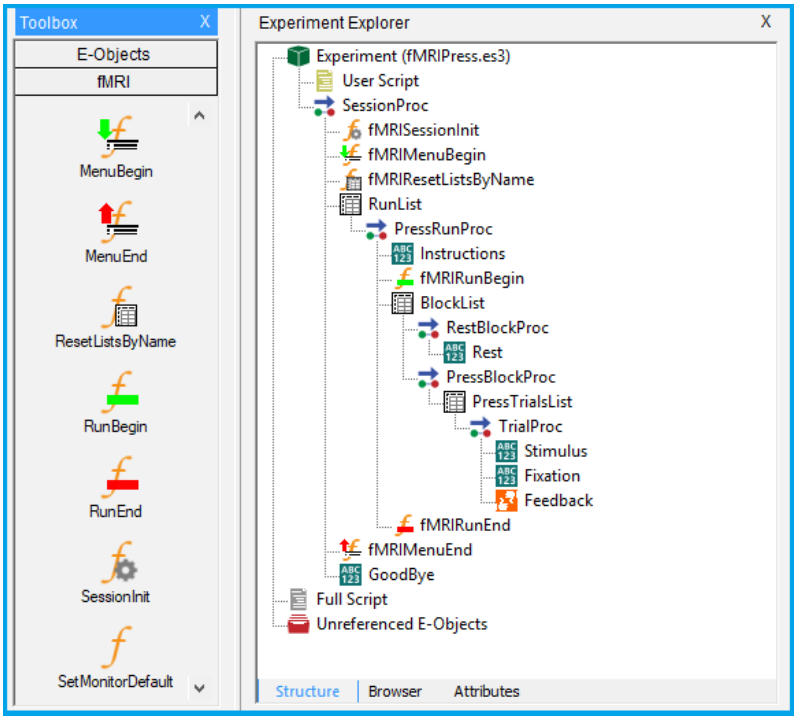
fMRIResetListsByName has two optional parameters that have already been discussed: the **Experiment Context** and **vHandlePreRelease**. However, note that fMRIResetListsByName has a second parameter, **strListNames**, that is required. It is a comma delimited string containing the names of List Objects that should be reset.

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



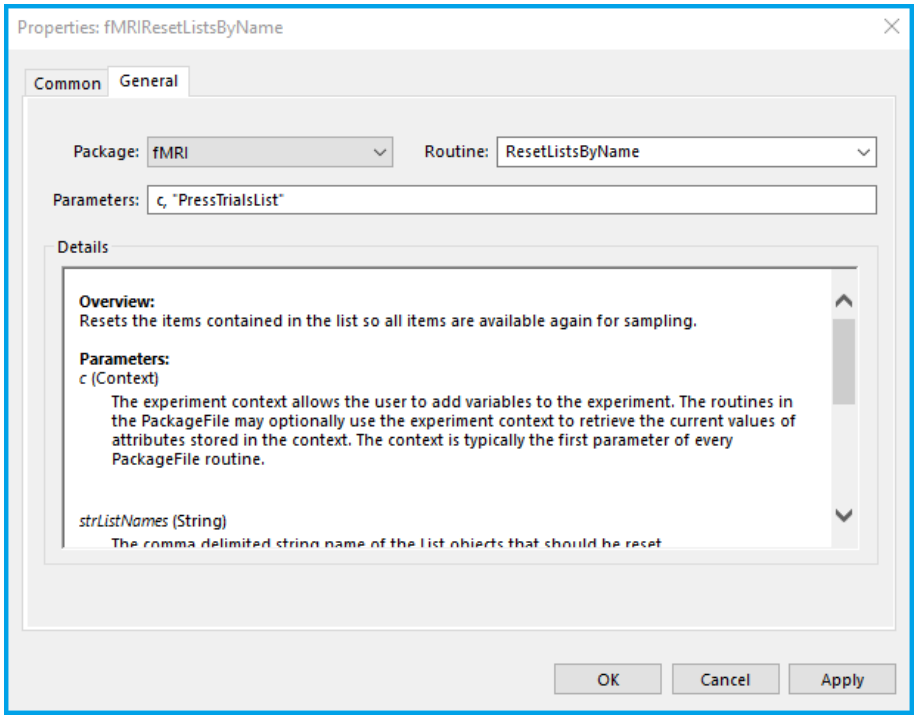

2) ***Click*** the **fMRI heading** in the **Toolbox** to display the **fMRI PackageCall Shortcuts**.

3) ***Drag*** the shortcut **fMRIResetListsByName** from the **EEfMRI Toolbox** and ***drop*** it in the **SessionProc** procedure after the **fMRIMenuBegin**.




4) ***Double click*** on the **fMRIResetListsByName** Object to open its **Properties** dialog.

5) ***Edit*** the **Parameters** field.




**NOTE**: *The recommended parameters are c, "PressTrialsList".*

6) ***Click*** the **OK** button to accept and dismiss the dialog.

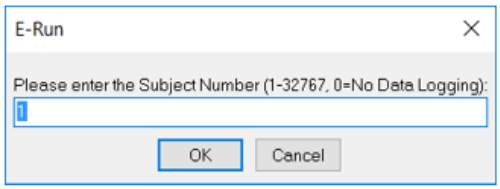
# Task 7: Run the Experiment

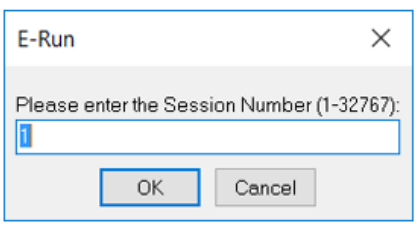
*Run the experiment to verify that the menu is functioning correctly and that no runtime errors are generated.*

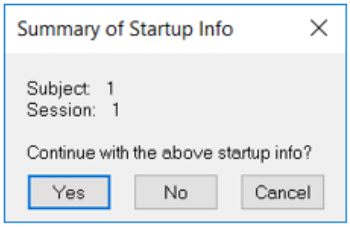
A functional menu is now implemented in the experiment. Run the experiment to verify that the experiment runs, the menu is shown, and the experiment completes successfully.

1) ***Click*** the **Run icon** on the toolbar to generate and run the experiment locally.

2) ***Press*** **Enter** to ***accept*** the **default values** for each of the initialization prompts presented.



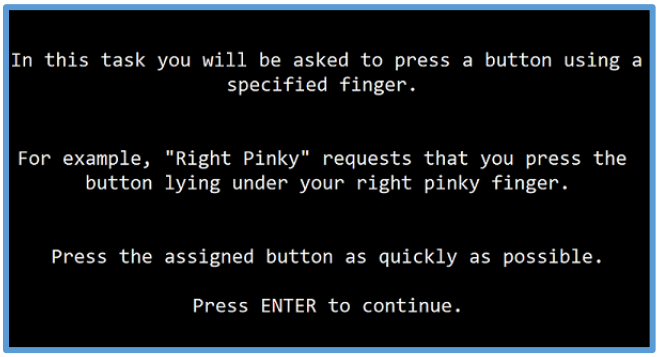




3) ***Verify*** the **Menu** appears.

4) ***Press*** **Enter** to ***select*** the highlighted menu option (i.e., “**Press**”).

5) ***Read*** the experiment instructions then ***press*** **Enter** to continue.




**NOTE**: *Enter is pressed to skip the wait for scanner trigger.*

6) ***Verify*** the **experiment** is ***functioning*** correctly and ***completes*** with **no errors** being generated.
***Verify*** the **Run Report *appears*** at the end of all the experimental trials.

## 3.5. Tutorial 3: Timing

***Summary:***

In an fMRI experiment, timing is critical. For many paradigms, such as when fixed parameters are in use, both the time between stimuli (the inter-stimulus interval or ISI), and the time between trials (the inter-trial interval or ITI) needs to vary as little as possible for all trials. Using the PreRelease option allows an executing object to free some available time for the next object in the execution sequence to prepare/ready itself for execution.

This tutorial takes you through the steps necessary to modify a behavioral experiment's timing using the PreRelease option and Cumulative Timing Mode to collect meaningful data during a functional MRI study.

**NOTE:** *All the Tutorials build upon each other and must be completed in order. Tutorial 3 begins where Tutorial 2 ends. This tutorial assumes that you have completed the previous Tutorials. You must load the experiment you edited in those Tutorials into E-Studio before continuing with Tutorial 3.*

**NOTE:** *The Sample experiment found in the folder …My Experiments\3.0\fMRI\Samples\fMRIPress.es3 represents the complete set of changes documented in Tutorials 1 through 5.*

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

***Goal:***

This tutorial illustrates how to modify Object properties to alter the timing to collect fMRI data.

***Overview of Tasks:***

- Modify the Rest, Stimulus, and Fixation Object timing.

***Recommended readings:***

It is required that you have read through and completed at least Tutorial 1 and 2 before beginning this tutorial.

Because this tutorial requires editing the Duration/Input tab of a given Object, we recommend reading the following article.

Article 22717: Duration/Input

***Estimated Tutorial Time:***

10-15 minutes

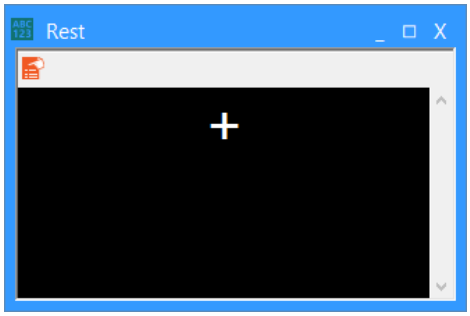
# Task 1: Modify Object Timing

*Open Object properties and modify the Pre-Release and Timing Mode.*

**Cumulative Timing Mode** corrects for delays introduced into the paradigm by delays in stimulus onset by altering the presentation time of the Stimulus. Used with PreRelease, this minimizes the delay of subsequent events due to accumulating error. The **PreRelease** option allows the computer to free resources to prepare for the next trial. If used correctly, both techniques serve as powerful tools to correct for timing errors. The observed effect of cumulative error in an fMRI experiment is typically that the scanner stops running before the experiment is finished presenting stimuli.

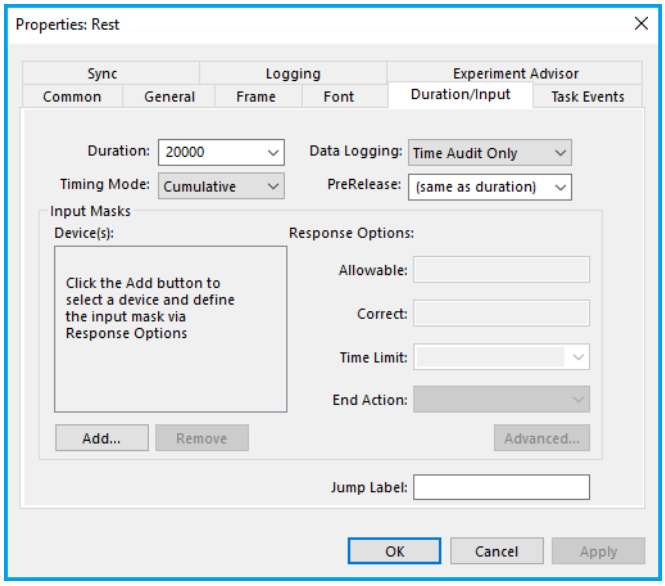
**NOTE**: *If it is not already opened, open the fMRIPress.es3 experiment you created and edited in the previous Tutorials.*

1) **Open** the **Rest** Object in the workspace.




2) ***Open*** the **Rest Object's Properties**.

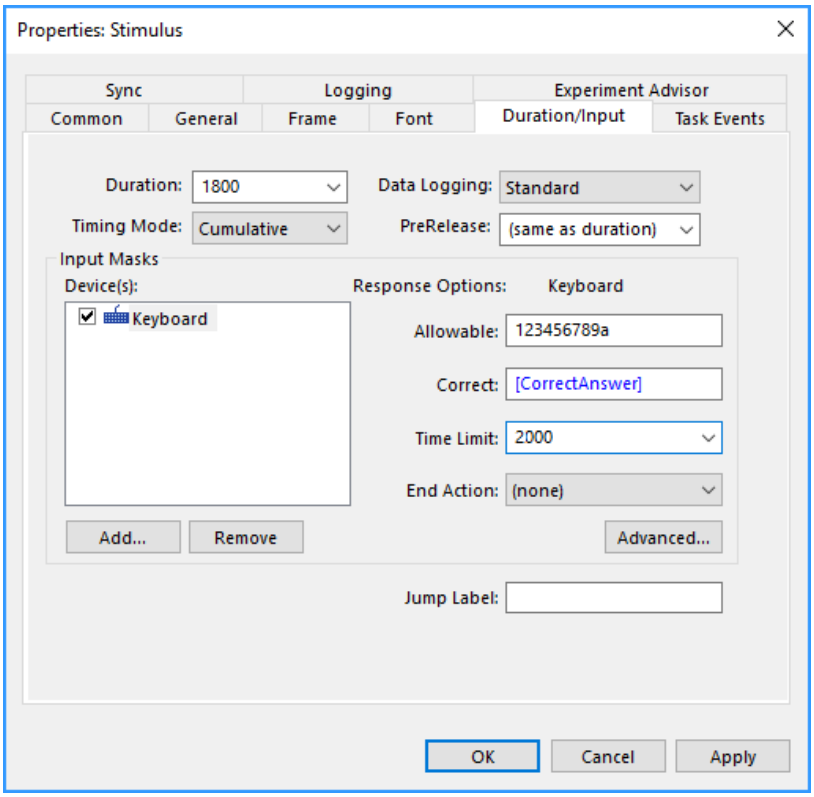
3) On the **Duration/Input tab**, ***verify*** **Cumulative** is selected from the **Timing Mode** dropdown.




4) **Verify** that **PreRelease** is **(same as duration)**.

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

6) ***Repeat* steps 1-5** for the **Stimulus** and **Fixation** Objects. However, when editing the **Stimulus** Object, ***select* 2000** from the **Time Limit** dropdown list.




**NOTE:** *With using 2000 ms for the Stimulus's Time Limit, the Stimulus Object presents for 1800 ms before being replaced by the Fixation Object for 200 ms between stimuli. However, the response period exists over the duration of both objects (i.e., you can respond during Stimulus and Fixation). For more information on Input Masks please view Article 22723: Input Devices.*

**NOTE**: *Cumulative Timing Mode minimizes delays in stimulus presentation by absorbing errors in the case there is an error in the stimulus onset time. For more information on Cumulative Timing Mode, see Article 22856: Operating System Solutions and Article 18285: Event vs. Cumulative Mode Timing.*

## Task 2 (Optional): Run the Experiment

*Run the experiment.*

**NOTE:** *This Task is optional. There are no observable differences between experiment runs that use Event Mode (as in Tutorials 1- 2) and those that use Cumulative Mode (Tutorials 3-5). To assess the impact of the timing mode change, compare the timestamps in the .pdat file for each timing mode. The .pdat file will be examined in the next Tutorial. You may run the experiment now to ensure that no errors were introduced in this tutorial, or you may proceed directly to the next tutorial.*

1) ***Click*** the **Run icon** on the toolbar to generate and run the experiment locally.

2) ***Press*** **Enter** to ***accept*** the **default values** for each of the initialization prompts presented.

3) ***Read*** the experiment instructions then ***press*** **Enter** to continue.

**NOTE**: *Enter is pressed to skip the wait for scanner trigger.*

4) ***Verify*** the **experiment** is ***functioning*** correctly and ***completes*** with **no errors** being generated.

## 3.6. Tutorial 4: Logging Block, Stimulus, and Response Data in the .pdat File

***Summary:***

An EEfMRI experiment can contain several runs (time between each scanner start and end) consisting of different tasks comprised of varied trial types. It is important to log the type of block (set of trials) and onset time of each block for later analysis.

To analyze EEfMRI data, it is essential to know the Stimulus order, what time the Stimulus occurred and the participant's response to the Stimulus. This data is often referred to as the stimulus record or the behavioral data. E-Prime collects the data and EEfMRI can write it to a file that is optimal for data analysis, the .pdat file.

**NOTE:** *All the Tutorials build upon each other and must be completed in order. Tutorial 4 begins where Tutorial 3 ends. This Tutorial assumes that you have completed the three previous Tutorials. You must load the experiment you edited in those Tutorials into E-Studio before continuing with Tutorial 4.*

**NOTE:** *The Sample experiment found in the folder …My Experiments\3.0\fMRI\Samples\fMRIPress.es3 represents the complete set of changes documented in Tutorials 1 through 5.*

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

***Goal:***

This Tutorial illustrates a method used to log the block condition and block onset time in an experiment. The structure of this experiment is necessary for the EEfMRI Package to log data. Additionally, it illustrates how to log the Stimulus, participant's responses, and stimulus onset times. It discuses logging the response for the Stimulus and behavioral response data collected during an EEfMRI experiment in the .pdat file.

***Overview of Tasks:***

- Add the fMRILogUserEventBlock PackageCall to log Block level data.
- Copy the fMRILogUserEventBlock PackageCall to other appropriate Procedures.
- Run the experiment to verify fMRILogUserEventBlock logged data in the .pdat file.
- Add the fMRILogEvent PackageCall to log the Stimulus information.
- Add the fMRILogResponseEvent PackageCall to collect the participant's response information.
- Run the experiment to verify fMRILogEvent and fMRILogResponseEvent logged data in the .pdat file.

***Recommended readings:***

It is required that you have read through and completed at least Tutorials 1, 2, and 3 before beginning this tutorial.

***Estimated Tutorial Time:***

10-15 minutes

# Task 1: Add and Copy the fMRILogUserEventBlock PackageCall

*Add the fMRILogUserEventBlock PackageCall to the Block Procedures.*

It is necessary to know what condition is active at any given point throughout the duration of the experiment. To achieve this, the fMRILogUserEventBlock PackageCall needs to be placed at the beginning of each block procedure. The **fMRILogUserEventBlock** PackageCall directs EEfMRI to log timing information and a user defined condition Id into the .pdat file for later user during analysis.

The first parameter (**c**) of fMRILogUserEventBlock is the Experiment Context and is required.

The second parameter (**nOnsetTime**) is required and is the onset time that is associated with the event. The onset time passed in is used directly unless -1 or -2 is specified ( -1 = Get current time stamp by calling Clock.Read, -2 = Get expected time stamp by calling GetNextTargetOnsetTime()).

The third parameter (**TargetOnsetTime**) is required. This is the target onset time that is associated with the event. The target onset time passed in is used directly unless -1 or -2 is specified ( -1 = Get current time stamp by calling Clock.Read, -2 = Get expected time stamp by calling GetNextTargetOnsetTime()).

The fourth parameter (**strRecordId**) is a required string parameter. This is an Id that describes the type of event record described. The EEfMRI system reserves some default Id strings, but the user may define their own as needed to assist in post processing activities.

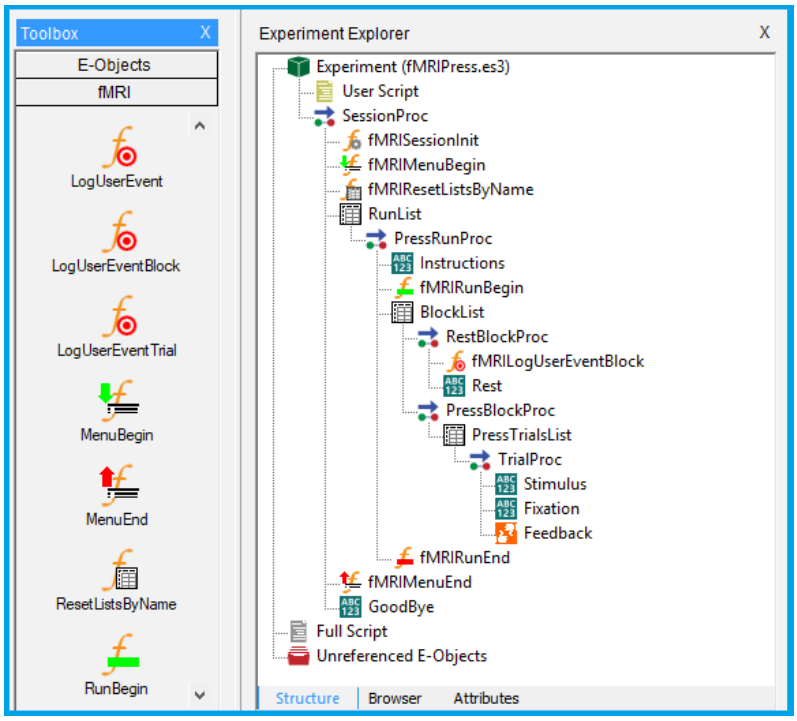
The fifth parameter (**nDuration**) is required and is the duration that is associated with the event. The duration passed in is used directly unless -1 or -2 is specified (-1 = Reserved, -2 = Flag the value to be calculated via post processing).

The sixth parameter (**strConditionId**) is optional. It is a string that describes an experimental condition to associate with the event.

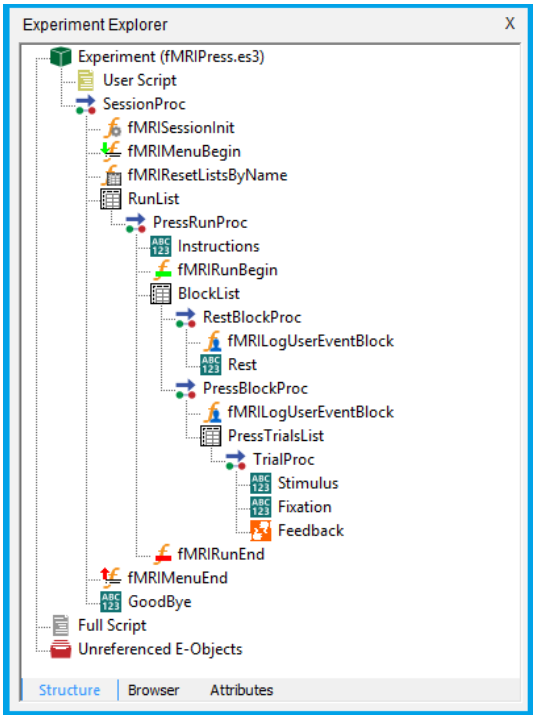
The seventh parameter (**vUserDefinedData**) is an optional tab delimited string that provides data for user defined columns in the .pdat file (e.g.,”\t” is used as the tab delimiter).

**NOTE**: *If it is not already opened, open the fMRIPress.es3 experiment you created and edited in the previous Tutorials.*

1) ***Drag*** the shortcut **fMRILogUserEventBlock** from the **fMRI Toolbox** and ***drop*** it before the **Rest** Object**.**




2) ***Drag*** the shortcut **fMRILogUserEventBlock** from **RestBlockProc** and ***drop*** it in the beginning of **PressBlockProc** before **PressTrialsList**.



# Task 2: Run the Experiment to Verify the Block Level Logging

*Run the experiment to verify that fMRILogUserEventBlock logged timing information for the Block level.*

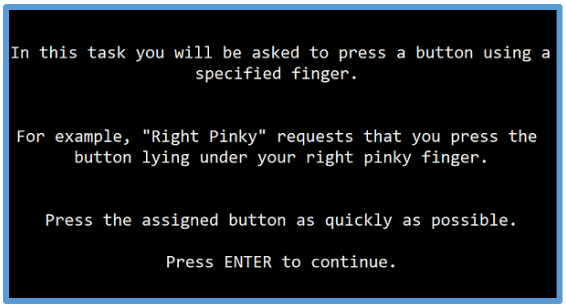
Both the RestBlockProc and PressBlockProc are now configured to have the block conditions logged with the addition of the PackageCall fMRILogUserEventBlock. After running the experiment, the .pdat file will have timing information from the Block level logged. We recommend running the experiment to verify the Block level information has logged appropriately.

1) ***Click*** the **Run icon** on the toolbar to generate and run the experiment locally.

2) ***Press*** **Enter** to ***accept*** the **default values** for each of the initialization prompts presented.

3) ***Verify*** the **Menu** appears.

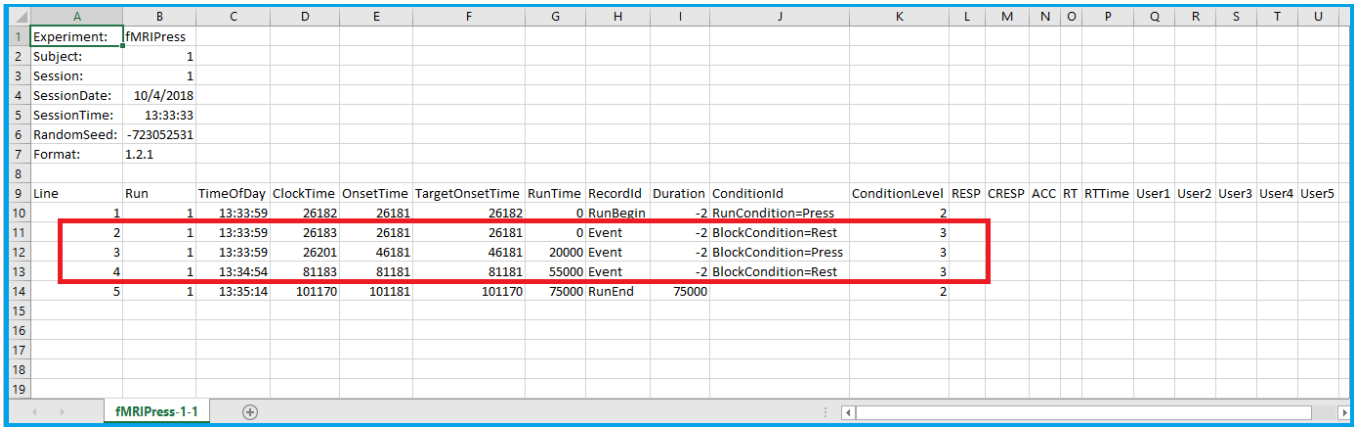
4) ***Read*** the experiment instructions then ***press*** **Enter** to continue.




**NOTE**: *Enter is pressed to skip the wait for scanner trigger.*

5) ***Verify*** the **experiment** is ***functioning*** correctly and ***completes*** with **no errors** being generated. ***Verify*** the **Run Report *appears*** at the end of all the experimental trials.

6) ***Open*** the .**pdat** file. ***Verify*** the **Block** level data was logged by **fMRILogUserEventBlock.**



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Experiment: | fMRIPress | | | | | | | | | | | | | | | | | | | |
| 2 | Subject: | 1 | | | | | | | | | | | | | | | | | | | |
| 3 | Session: | 1 | | | | | | | | | | | | | | | | | | | |
| 4 | SessionDate: | 10/4/2018 | | | | | | | | | | | | | | | | | | | |
| 5 | SessionTime: | 13:33:33 | | | | | | | | | | | | | | | | | | | |
| 6 | RandomSeed: | -723052531 | | | | | | | | | | | | | | | | | | | |
| 7 | Format: | 1.2.1 | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | |
| 9 | Line | Run | TimeOfDay | ClockTime | OnsetTime | TargetOnsetTime | RunTime | RecordId | Duration | ConditionId | ConditionLevel | RESP | CRESP | ACC | RT | RTTime | User1 | User2 | User3 | User4 | User5 |
| 10 | 1 | 1 | 13:33:59 | 26182 | 26181 | 26182 | 0 | RunBegin | -2 | RunCondition=Press | 2 | | | | | | | | | | |
| 11 | 2 | 1 | 13:33:59 | 26183 | 26181 | 26181 | 0 | Event | -2 | BlockCondition=Rest | 3 | | | | | | | | | | |
| 12 | 3 | 1 | 13:33:59 | 26201 | 46181 | 46181 | 20000 | Event | -2 | BlockCondition=Press | 3 | | | | | | | | | | |
| 13 | 4 | 1 | 13:34:54 | 81183 | 81181 | 81181 | 55000 | Event | -2 | BlockCondition=Rest | 3 | | | | | | | | | | |
| 14 | 5 | 1 | 13:35:14 | 101170 | 101181 | 101170 | 75000 | RunEnd | 75000 | | 2 | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | | | | | | | | | |

fMRIPress-1-1

**NOTE**: *In the above image the data in red is the Block level data.*

# Task 3: Add the fMRILogEvent PackageCall

*Add a PackageCall to the TrialProc that directs EEfMRI to log the specified object in the .pdat file.*

The **fMRILogEvent** PackageCall logs the onset time and condition of each stimulus through the duration of an experiment. Furthermore, it directs EEfMRI to record this information in the .pdat file for use during analysis. We recommend that this PackageCall is placed after the Object (e.g., Stimulus in this Task) with the event that has been run.

The first parameter (**c**) is the Experiment Context and is required.

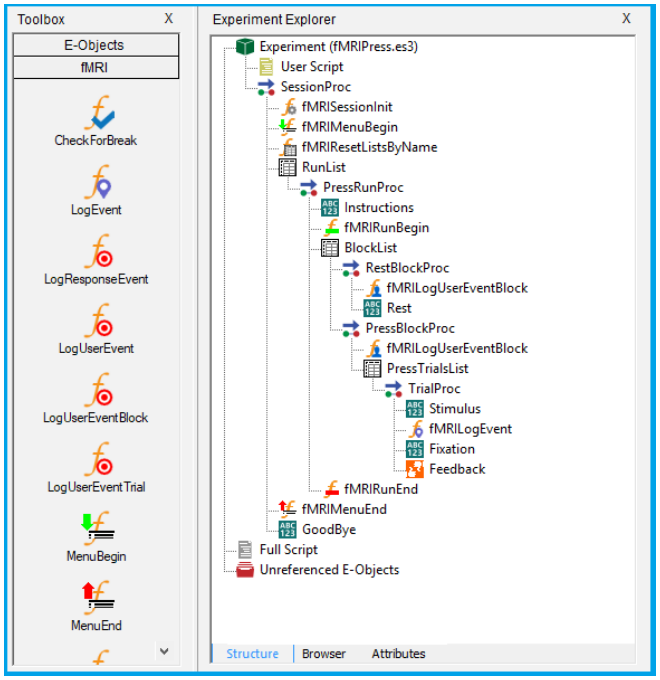
The second parameter (**theObject**) is required. It is the Object that corresponds to the event (e.g., Stimulus in this Task). The time of the event is set to value saved in object's OnsetTime property.

The third parameter (**nDuration**) is required and is the duration associated with the event. The duration passed in is used directly unless -1 or -2 is specified (-1 = Set the duration based on value of the theObject.Duration property, determines if handling of PreRelease from previous object is needed, -2 = Flag the value to be calculated via post processing).

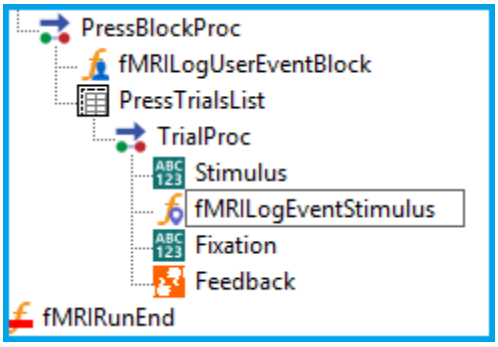
The fourth parameter (**strConditionId**) is a required string parameter that describes an experimental condition associated with the event.

The fifth parameter (**vUserDefinedData**) is an optional tab delimited string that provides data for user defined columns in the .pdat file (e.g.,"\t" is used as the tab delimiter).

1) ***Drag*** the shortcut **fMRILogEvent** from the **fMRI Toolbox** and ***drop*** it after the **Stimulus** Object**.**

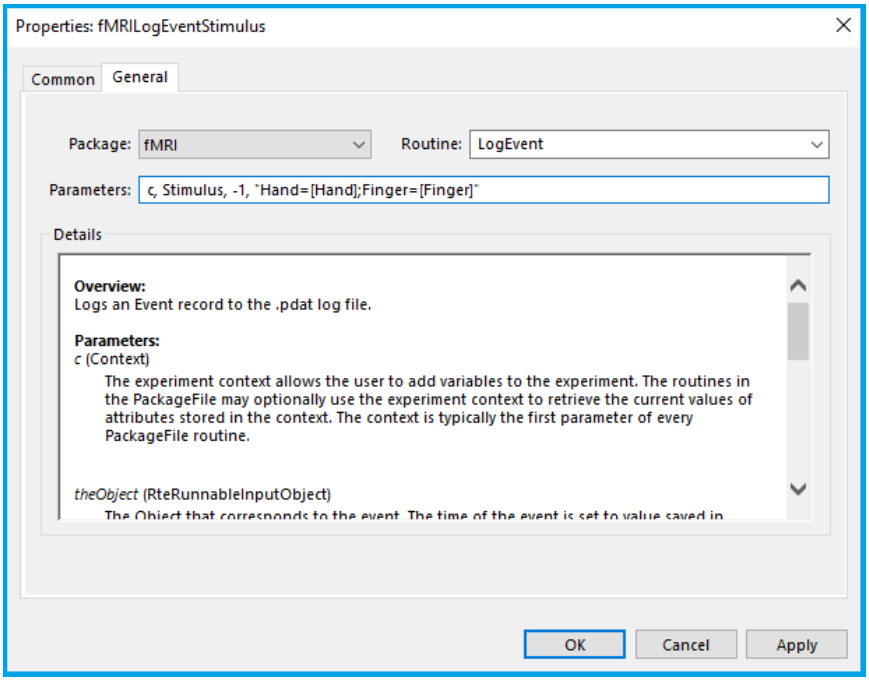


2) ***Rename*** (F2) **fMRILogEvent** as " **fMRILogEventStimulus**".




3) ***Double click*** on the **fMRILogEventStimulus** Object to open its **Properties** dialog.

4) ***Edit*** the **Parameters** field.




**NOTE**: *The recommended parameters are c, Stimulus, -1, "Hand=[Hand];Finger=[Finger]"*

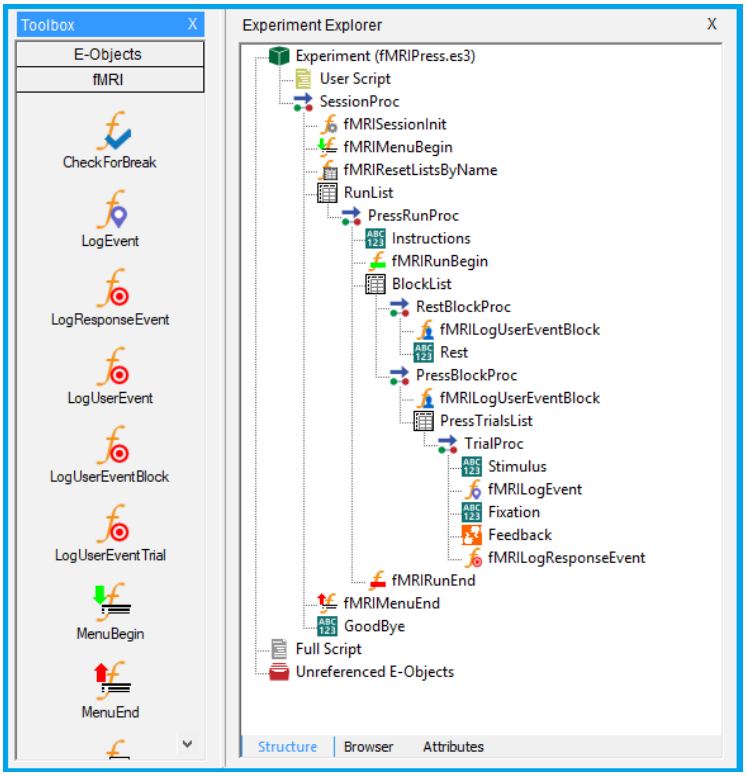
5) ***Click*** the **OK** button to accept and dismiss the dialog.

# Task 4: Add fMRILogResponseEvent PackageCall

*Add the fMRILogResponseEvent PackageCall to the TrialProc.*

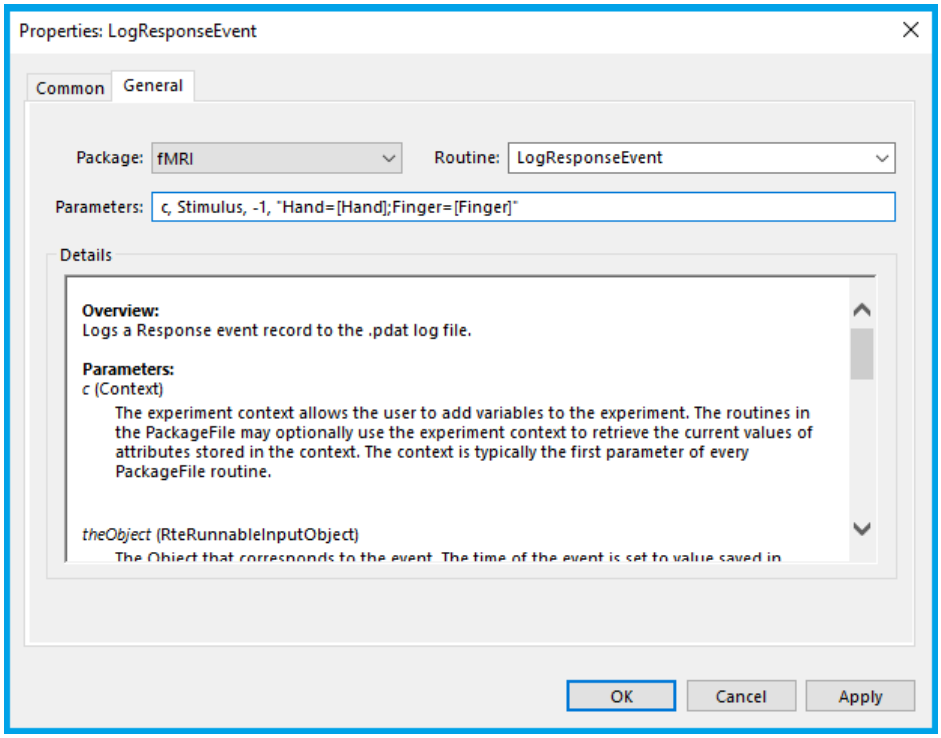
**fMRILogResponseEvent** enables EEfMRI to output the participant's response and the time it occurred into the .pdat file. It is important to place the call after the response period has been collected. Like the fMRILogEvent PackageCall, the fMRILogResponseEvent PackageCall is also associated with the Object displaying the Trials/Stimulus; in this task the Stimulus Object. However, the fMRILogResponseEvent is specifically designed to collect responses, label responses, and record the reaction times and timestamps. The same parameters as the fMRILogEvent are used. The placement of the PackageCall is crucial to collecting a response.

1) ***Drag*** the shortcut **fMRILogResponseEvent** from the **fMRI Toolbox** and ***drop*** it after the **Feedback** Object**.**




2) ***Double click*** on the **fMRILogResponseEvent** Object to open its **Properties** dialog.

3) ***Edit*** the **Parameters** field.




**NOTE**: *The recommended parameters are c, Stimulus, -1, "Hand=[Hand];Finger=[Finger]"*

4) ***Click*** the **OK** button to accept and dismiss the dialog.

# Task 5: Run the Experiment to Verify the Trial Level Logging

*Run the experiment to verify that information is logged for the Trial level.*

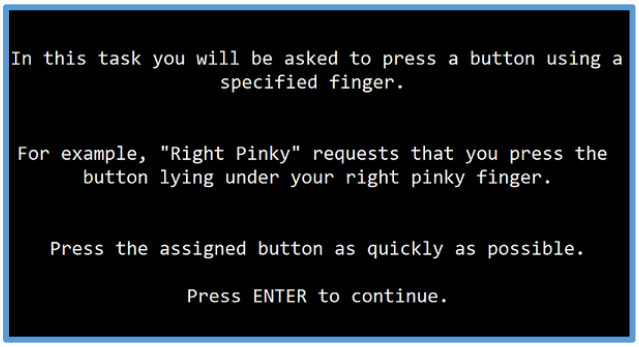
Stimulus and response data are now being logged in the .pdat file with the additions of the **fMRILogEventStimulus** and **fMRILogResponseEvent** PackageCalls. After running the experiment, the .pdat file will have timing information from the Stimulus logged. We recommend running the experiment to verify the Stimulus response information has logged appropriately.

1) ***Click*** the **Run icon** on the toolbar to generate and run the experiment locally.

2) ***Press*** **Enter** to ***accept*** the **default values** for each of the initialization prompts presented.

3) ***Verify*** the **Menu** appears.

4) ***Read*** the experiment instructions then ***press*** **Enter** to continue.




**NOTE**: *Enter is pressed to skip the wait for scanner trigger.*

5) ***Verify*** the **experiment** is ***functioning*** correctly and ***completes*** with **no errors** being generated. ***Verify*** the **Run Report *appears*** at the end of all the experimental trials.

**NOTE**: *In the image above notice that the Run Report is displaying data for responses, mean accuracy, mean RT, and mean correct RT. This is due to the logging by fMRILogEventStimulus and fMRILogResponseEvent.*

6) ***Open*** the .**pdat** file. ***Verify*** the **Trial** level data was logged by **fMRILogEventStimulus** and **fMRILogResponseEvent.**

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Experiment: | fMRIPress | | | | | | | | | | | | | | | | | | | |
| 2 | Subject: | 1 | | | | | | | | | | | | | | | | | | | |
| 3 | Session: | 1 | | | | | | | | | | | | | | | | | | | |
| 4 | SessionDate: | 10/4/2018 | | | | | | | | | | | | | | | | | | | |
| 5 | SessionTime: | 14:17:37 | | | | | | | | | | | | | | | | | | | |
| 6 | RandomSeed: | -15789975 | | | | | | | | | | | | | | | | | | | |
| 7 | Format: | 1.2.1 | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | |
| 9 | Line | Run | TimeOfDay | ClockTime | OnsetTime | TargetOnsetTime | RunTime | RecordId | Duration | ConditionId | ConditionLevel | RESP | CRESP | ACC | RT | RTTime | User1 | User2 | User3 | User4 | User5 |
| 10 | 1 | 1 | 14:17:54 | 17268 | 17267 | 17268 | 0 | RunBegin | -2 | RunCondition=Press | 2 | | | | | | | | | | |
| 11 | 2 | 1 | 14:17:54 | 17269 | 17267 | 17267 | 0 | Event | -2 | BlockCondition=Rest | 3 | | | | | | | | | | |
| 12 | 3 | 1 | 14:17:54 | 17290 | 37267 | 37267 | 20000 | Event | -2 | BlockCondition=Press | 3 | | | | | | | | | | |
| 13 | 4 | 1 | 14:18:14 | 37271 | 37271 | 37267 | 20004 | Event | 1800 | Hand=Right;Finger=Index | 4 | | | | | | | | | | |
| 14 | 5 | 1 | 14:18:18 | 40767 | 37271 | 37267 | 20004 | Response | 1140 | Hand=Right;Finger=Index | 4 | 2 | 2 | 1 | 1140 | 38411 | | | | | |
| 15 | 6 | 1 | 14:18:18 | 40788 | 40787 | 40767 | 23520 | Event | 1800 | Hand=Right;Finger=Ring | 4 | | | | | | | | | | |
| 16 | 7 | 1 | 14:18:21 | 44267 | 40787 | 40767 | 23520 | Response | 1376 | Hand=Right;Finger=Ring | 4 | 4 | 4 | 1 | 1376 | 42163 | | | | | |
| 17 | 8 | 1 | 14:18:21 | 44287 | 44287 | 44267 | 27020 | Event | 1800 | Hand=Left;Finger=Ring | 4 | | | | | | | | | | |
| 18 | 9 | 1 | 14:18:25 | 47767 | 44287 | 44267 | 27020 | Response | 1628 | Hand=Left;Finger=Ring | 4 | 4 | 9 | 0 | 1628 | 45915 | | | | | |
| 19 | 10 | 1 | 14:18:25 | 47787 | 47787 | 47767 | 30520 | Event | 1800 | Hand=Right;Finger=Middle | 4 | | | | | | | | | | |
| 20 | 11 | 1 | 14:18:28 | 51267 | 47787 | 47767 | 30520 | Response | 976 | Hand=Right;Finger=Middle | 4 | 3 | 3 | 1 | 976 | 48763 | | | | | |
| 21 | 12 | 1 | 14:18:28 | 51287 | 51287 | 51267 | 34020 | Event | 1800 | Hand=Left;Finger=Pinky | 4 | | | | | | | | | | |
| 22 | 13 | 1 | 14:18:32 | 54767 | 51287 | 51267 | 34020 | Response | 924 | Hand=Left;Finger=Pinky | 4 | 5 a | | 0 | 924 | 52211 | | | | | |
| 23 | 14 | 1 | 14:18:32 | 54787 | 54786 | 54767 | 37519 | Event | 1800 | Hand=Left;Finger=Index | 4 | | | | | | | | | | |
| 24 | 15 | 1 | 14:18:35 | 58267 | 54786 | 54767 | 37519 | Response | 1081 | Hand=Left;Finger=Index | 4 | 2 | 7 | 0 | 1081 | 55867 | | | | | |
| 25 | 16 | 1 | 14:18:35 | 58287 | 58286 | 58267 | 41019 | Event | 1800 | Hand=Right;Finger=Pinky | 4 | | | | | | | | | | |
| 26 | 17 | 1 | 14:18:39 | 61767 | 58286 | 58267 | 41019 | Response | 877 | Hand=Right;Finger=Pinky | 4 | 5 | 5 | 1 | 877 | 59163 | | | | | |
| 27 | 18 | 1 | 14:18:39 | 61786 | 61786 | 61767 | 44519 | Event | 1800 | Hand=Left;Finger=Middle | 4 | | | | | | | | | | |
| 28 | 19 | 1 | 14:18:42 | 65267 | 61786 | 61767 | 44519 | Response | 1194 | Hand=Left;Finger=Middle | 4 | 3 | 8 | 0 | 1194 | 62980 | | | | | |
| 29 | 20 | 1 | 14:18:42 | 65286 | 65286 | 65267 | 48019 | Event | 1800 | Hand=Left;Finger=Thumb | 4 | | | | | | | | | | |
| 30 | 21 | 1 | 14:18:46 | 68767 | 65286 | 65267 | 48019 | Response | 1757 | Hand=Left;Finger=Thumb | 4 | 1 | 6 | 0 | 1757 | 67043 | | | | | |
| 31 | 22 | 1 | 14:18:46 | 68786 | 68786 | 68767 | 51519 | Event | 1800 | Hand=Right;Finger=Thumb | 4 | | | | | | | | | | |
| 32 | 23 | 1 | 14:18:49 | 72267 | 68786 | 68767 | 51519 | Response | 465 | Hand=Right;Finger=Thumb | 4 | 1 | 1 | 1 | 465 | 69251 | | | | | |
| 33 | 24 | 1 | 14:18:49 | 72270 | 72267 | 72267 | 55000 | Event | -2 | BlockCondition=Rest | 3 | | | | | | | | | | |
| 34 | 25 | 1 | 14:19:09 | 92255 | 92267 | 92255 | 75000 | RunEnd | 75000 | | 2 | | | | | | | | | | |

fMRIPress-1-1

**NOTE**: *In the above image the data in red is the Stimulus response data.*

## 3.7. Tutorial 5: Creating a Practice Run

***Summary:***

The goal of running an experiment is to collect useable data. To increase the likelihood of collecting good data, participants are often trained before beginning the functional MRI scan to ensure that the task is understood. During the training process, it is necessary to provide the participant with feedback to facilitate his/her learning process. However, when it is time to start the data collection, feedback is often unnecessary or unwanted. It is advantageous for an experiment to have the ability to control the presence or absence of feedback when the experimenter deems necessary.

**NOTE:** *All the Tutorials build upon each other and must be completed in order. Tutorial 5 begins where Tutorial 4 ends. This Tutorial assumes that you have completed the four previous Tutorials. You must load the experiment you edited in those Tutorials into E-Studio before continuing with Tutorial 5.*

**NOTE:** *The Sample experiment found in the folder …My Experiments\3.0\fMRI\Samples\fMRIPress.es3 represents the complete set of changes documented in Tutorials 1 through 5.*

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

***Goal:***

This Tutorial illustrates how to create a practice run and include it as a new menu item.

***Overview of Tasks:***

- Edit the RunList Object to include a practice run.
- Add an InLine Object capable of skipping feedback when a practice run is not occurring.
- Add a Label to indicate where to resume the experiment if Feedback is skipped.

***Recommended readings:***

It is required that you have read through and completed at least Tutorials 1 and 2 before beginning this tutorial.

***Estimated Tutorial Time****:*

10-15 minutes

# Task 1: Edit the RunList Properties to Generate a Practice Choice in the Menu

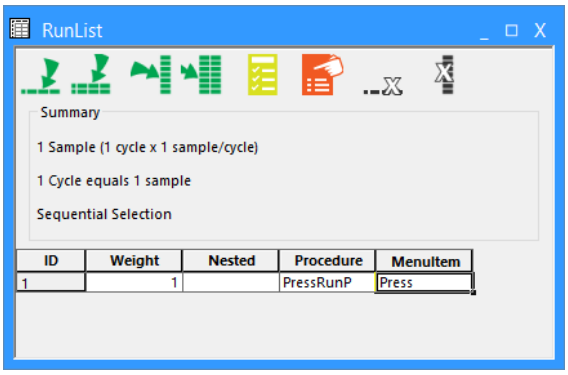
*Edit the RunList to differentiate between a practice and experiment run.*

For this tutorial, the difference between a practice and experiment run is the presence or absence of feedback to the participant on each trial. To enable an experiment task to either present or omit feedback requires several steps. The first step is to distinguish between practice and experiment runs (tasks). To accomplish this, an attribute named “Practice” is added to one of the List Objects.

In *Tutorial 2: Creating a Menu*, the MenuItem attribute was introduced. This attribute controls the text labels that are shown in the user menu. The menu can be modified so practice and experiment runs can be distinguished. To do this, the MenuItem attribute needs to be modified to read “Press-Practice.”

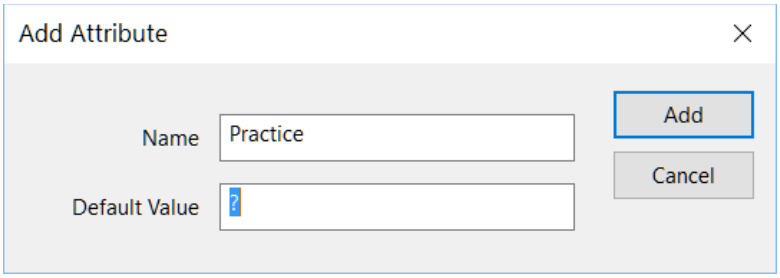
**NOTE**: *If it is not already opened, open the fMRIPress.es3 experiment you created and edited in the previous Tutorials.*

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




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

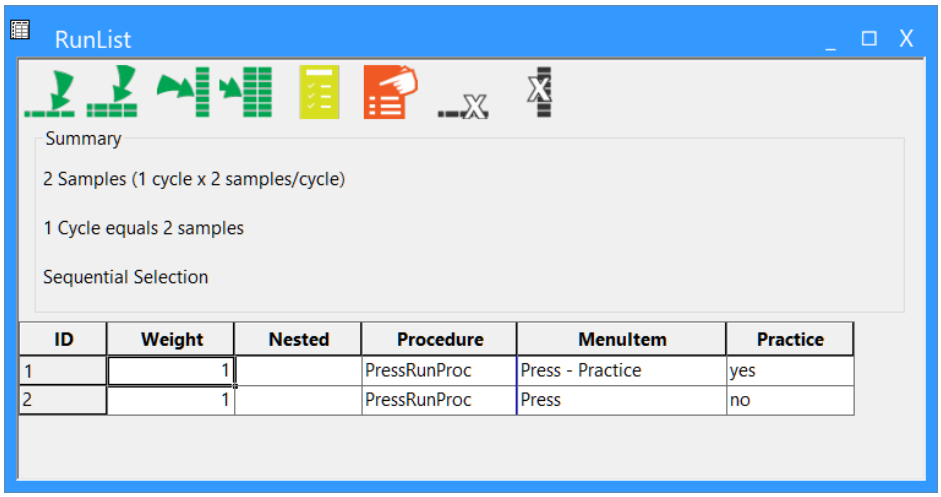
3) ***Edit*** the **Name** to read *“***Practice***”. **Click** **Add**.



4) ***Click*** the **Add Level** button.

5) In **row one**, ***Edit*** the **MenuItem** cell to read *"***Press-Practice***"*. ***Edit*** **the Practice** cell to read "**yes**".

6) In **row two**, ***Edit*** the **Procedure** cell to read *"***PressRunProc***"*. ***Edit*** **the MenuItem** cell to read "**Press**". ***Edit*** **the Practice** cell to read "**no**".




| ID | Weight | Nested | Procedure | MenuItem | Practice |
|---|---|---|---|---|---|
| 1 | 1 | | PressRunProc | Press - Practice | yes |
| 2 | 1 | | PressRunProc | Press | no |

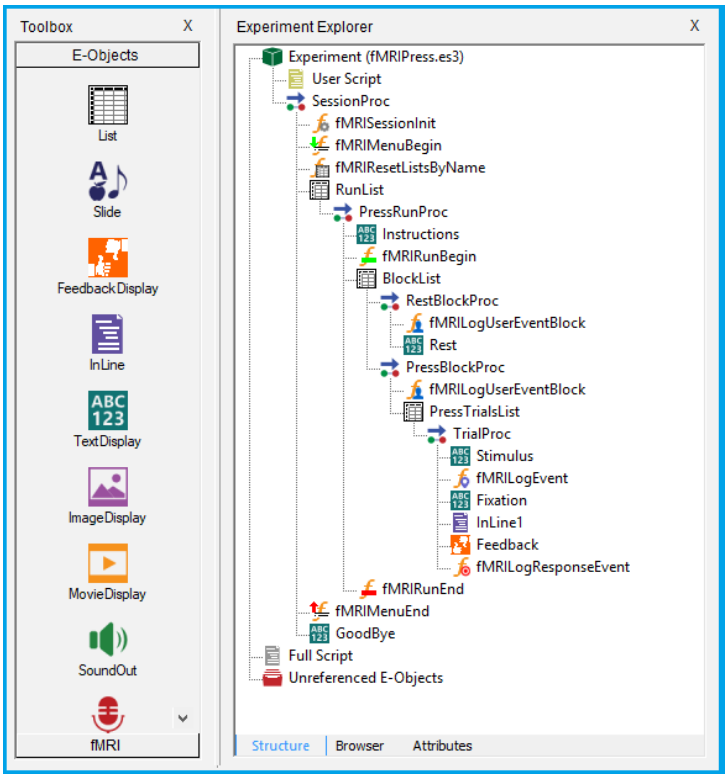
# Task 2: Add and Edit an InLine Object

*Add an InLine Object to TrialProc to skip the Feedback Object.*

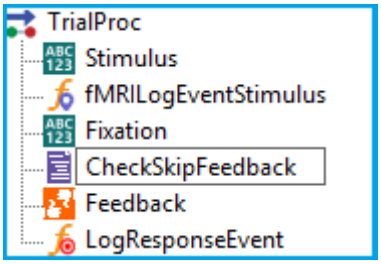
In this step we will enable the program to skip the Feedback display if the user has selected the Press (non-practice) run. This requires adding an InLine Object to the TrialProc that instructs the experiment to skip over the Feedback Object and go to a designated Label (the Label will be added in Task 3).

1) ***Click*** the **E-Objects heading** in the **Toolbox** to display the E-Prime E-Objects.

2) ***Drag*** an **InLine** Object from the **E-Objects Toolbox** and ***drop*** it after the **Fixation Object**.




3) ***Rename*** (F2) the new **InLine** object "**CheckSkipFeedback**".

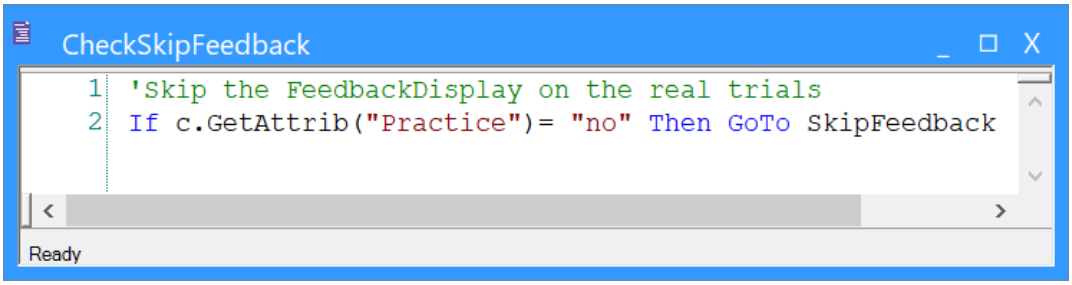


4) ***Double click*** on the **CheckSkipFeedback** InLine Object to open it in the workspace.

5) ***Type*** in **the following script**.

```
'     Skip the Feedback during the real trials
If c.GetAttrib("Practice") = "no" Then GoTo SkipFeedback
```

**NOTE**: *Your final CheckSkipFeedback InLine Object should look the same as the image below:*



# Task 3: Add and Edit a Label Object

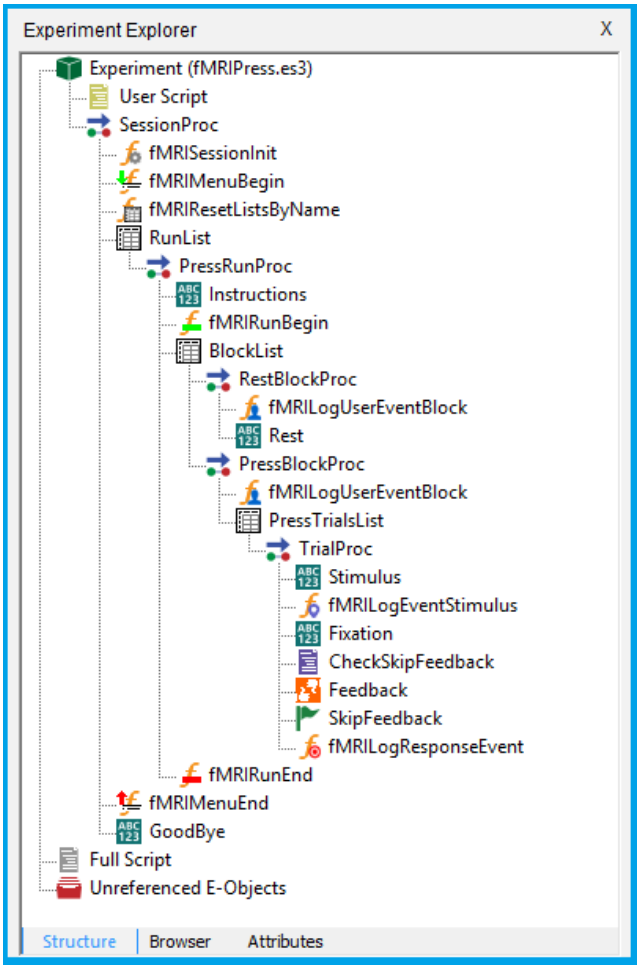
*Add a label to TrialProc to skip the Feedback Object.*

Once the InLine Object is placed in the experiment, a corresponding Label Object needs to be added. The Label signifies the location where the experiment "jumps to" during the execution of the script in the InLine Object. The Label needs to occur after the Feedback Object so that the Feedback is skipped during the real trials.

1) ***Drag*** a Label Object from the **E-Objects Toolbox** and ***drop*** it after the **Feedback** Object.

2) ***Rename*** (F2) the new **Label** Object "**SkipFeedback**".

3) ***Verify*** your **experiment structure** matches the **image** below.



# Task 4: Modify the fMRIRunBegin PackageCall

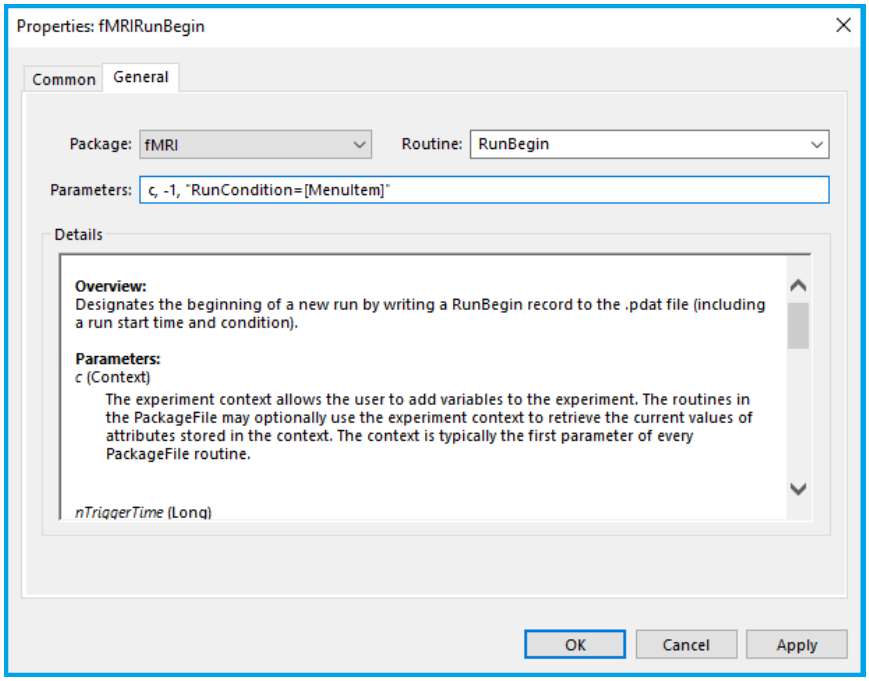
*Modify the fMRIRunBegin PackageCall to identify the experimental run condition.*

Now that the Attribute “MenuItem” contains both “Press-Practice” and “Press,” it is necessary to edit fMRIRunBegin’s third parameter, strConditionId. Editing the parameters to include the “MenuItem” Attribute allow strConditionId to record whether practice or experimental trials are occurring.

**NOTE**: For more information on fMRIRunBegin refer to *Tutorial 1: Task 6: Add the fMRIRunBegin PackageCall.*

1) ***Double click*** on the **fMRIRunBegin** Object to open its **Properties** dialog.

2) ***Edit*** the **Parameters** field.




**NOTE**: *The recommended parameters are c, -1, "RunCondition=[MenuItem]".*

3) ***Click*** the **OK** button to accept and dismiss the dialog.

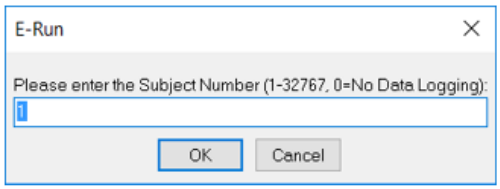
# Task 5: Run the Experiment

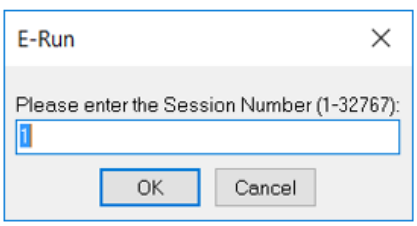
*Run the experiment to verify that the Feedback is skipped during the real trials.*

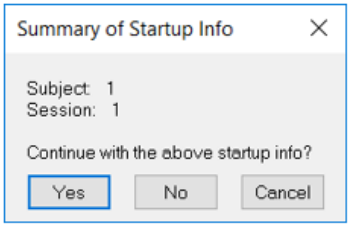
The option to select practice trials is now available in the Menu. Run the experiment and select Press-Practice from the Menu. Verify that the Feedback Object is displayed during the practice trials and shown during the real trials.

1) ***Click*** the **Run icon** on the toolbar to generate and run the experiment locally.

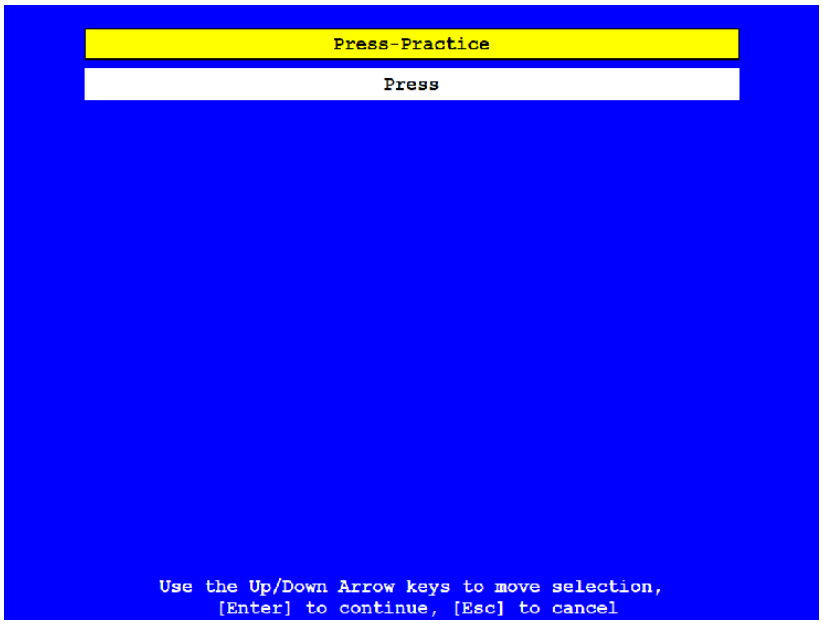
2) ***Press*** **Enter** to ***accept*** the **default values** for each of the initialization prompts presented.



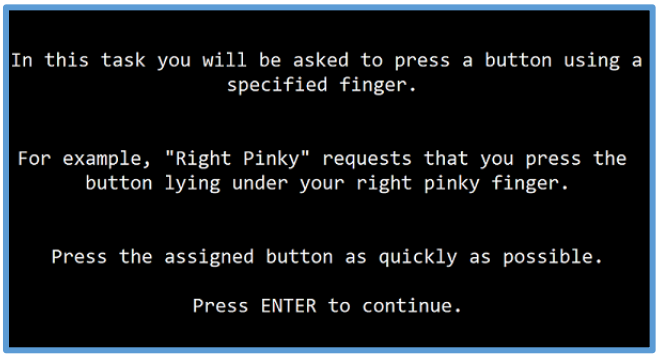




3) ***Verify*** the **Menu** appears. ***Press*** **Enter** to ***select*** the highlighted menu option (i.e., "**Press-Practice**").

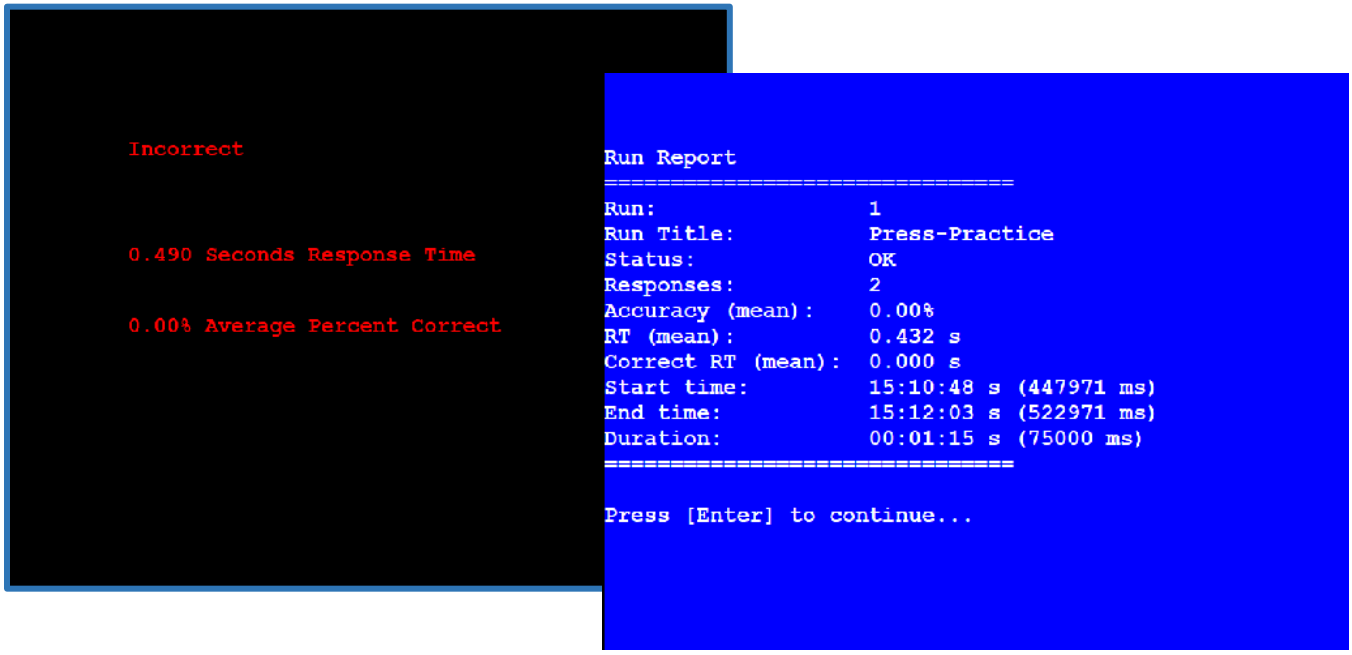



4) ***Read*** the experiment instructions then ***press*** **Enter** to continue.




**NOTE**: *Enter is pressed to skip the wait for scanner trigger.*

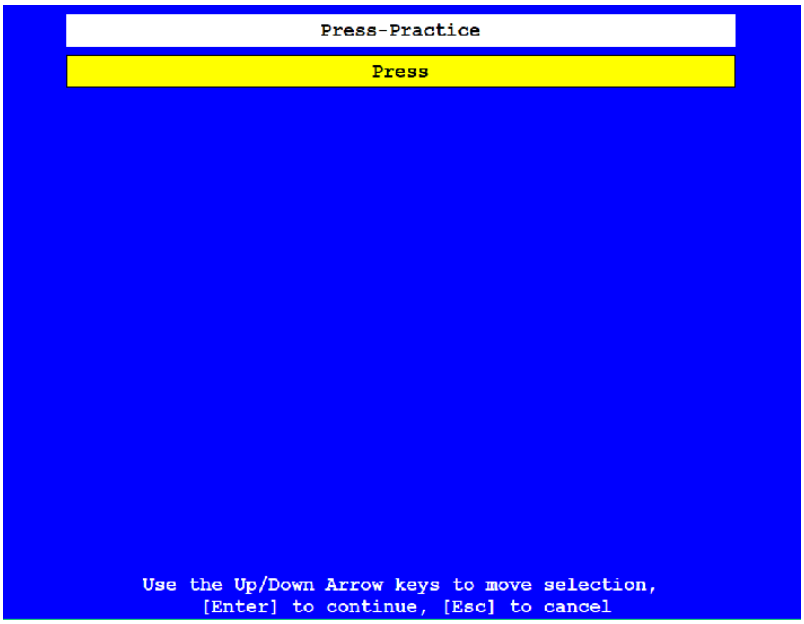
5) ***Verify*** the **experiment** is ***functioning*** correctly and ***completes*** with **no errors** being generated. ***Verify*** **Feedback** is shown during the **practice trials**. ***Verify*** the **Run Report** ***appears*** at the end of the practice trials.




**NOTE**: *In the image above notice that the Run Report is displaying the Run Title as "Press-Practice."*

6) ***Press*** **Enter** to **continue** and **dismiss** the **Run Report**.

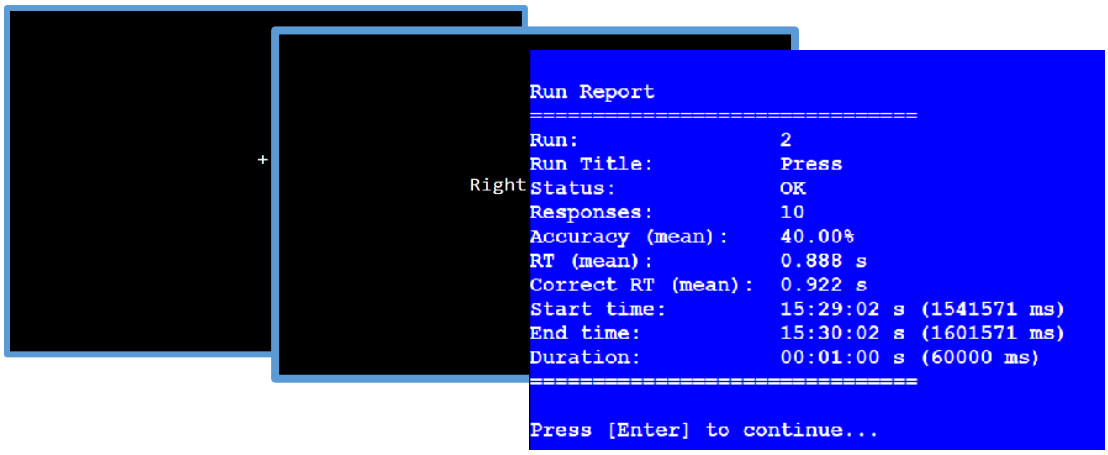
7) ***Verify*** the **Menu** appears. ***Navigate*** with the **Down Arrow** on the keyboard to ***Select*** the **menu option** "**Press.**" ***Press*** **Enter** to ***select*** the menu option.




8) ***Read*** the experiment instructions then ***press*** **Enter** to continue.

**NOTE**: *Enter is pressed to skip the wait for scanner trigger.*

9) ***Verify*** the **experiment** is ***functioning*** correctly and ***completes*** with **no errors** being generated. ***Verify*** **Feedback** is not shown during the **experimental trials**. ***Verify*** the **Run Report** ***appears*** at the end of the practice trials.




**NOTE**: *In the image above notice that the Run Report is displaying the Run Title as "Press."*

# 4. Troubleshooting

The following topics are common issues and solutions to help you when implementing and testing experiments with EEfMRI. In the case that one of the topics cannot fix your issue, Psychology Software Tools recommends saving the experiment, closing your E-Studio session, and then attempting to replicate the problem. If any issue persists, please contact Product Service and Support.

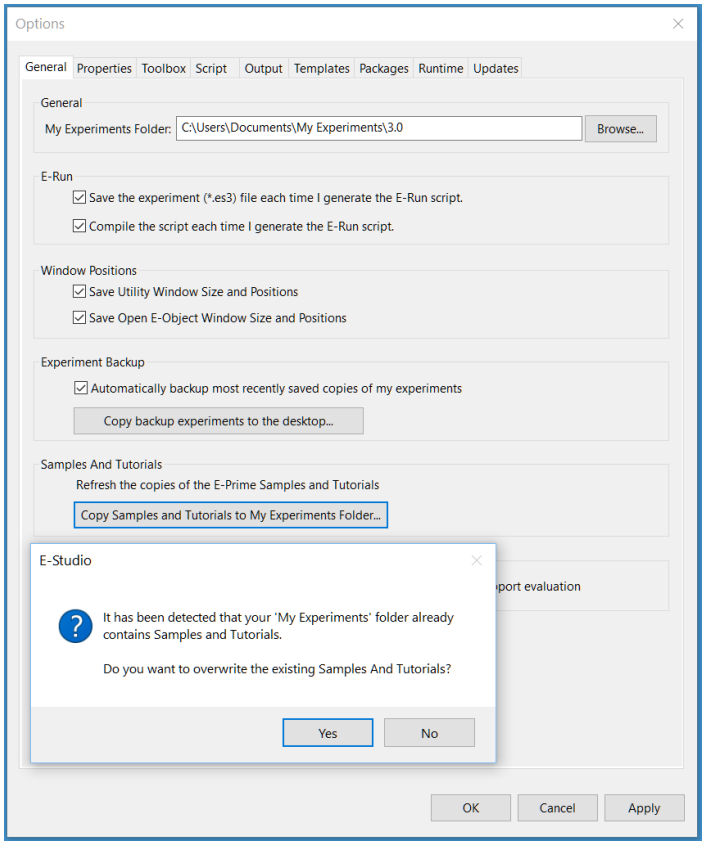
## 4.1. Cannot locate E-Prime Samples and Tutorials

When you launch E-Studio for the first time after installing E-Prime Extensions for fMRI, you are prompted to update the Samples and Tutorials experiments. If you choose to not update now, you can force another update of the Sample and Tutorials at any time from within E-Studio. Forcing a refresh of the Samples and Tutorials can be helpful if you cannot locate the Samples and Tutorials. This feature is also helpful if you make changes to a Sample or Tutorial but then wish to revert to the original Sample or Tutorial.

1) ***Open*** **E-Studio** and ***select*** **Tools** > **Options**… from the application menu bar.

2) ***Click*** on the "**Copy Samples and Tutorials to My Experiments folder…**" button at the bottom of the Options dialog.

3) ***Click*** on **Yes** if prompted to overwrite the Samples and Tutorials.

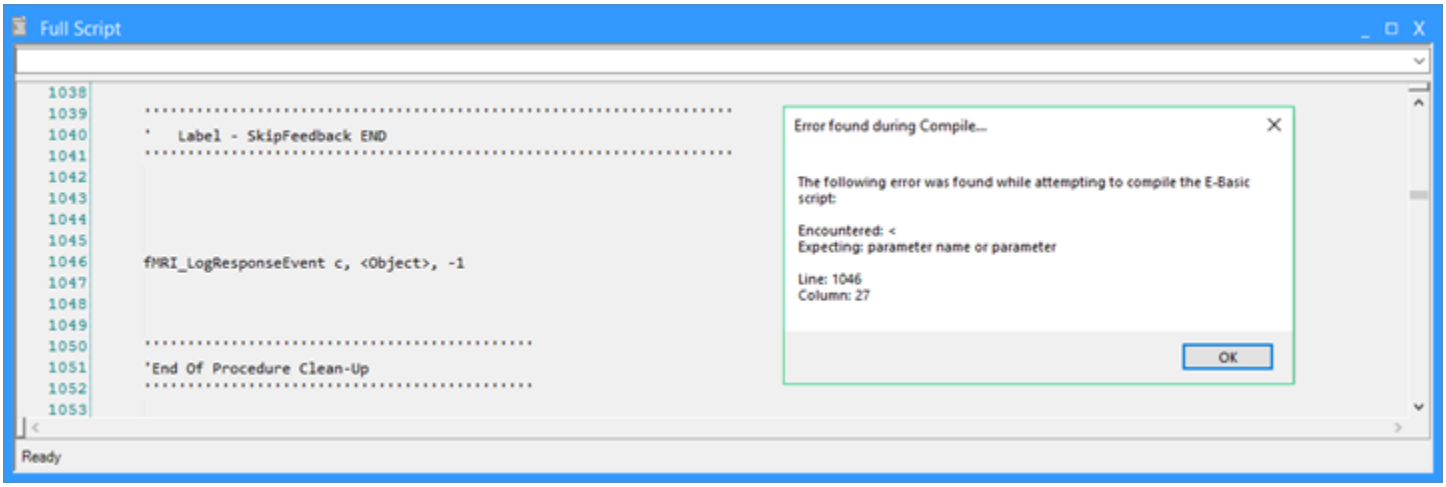


## 4.2. Missing Parameter Errors

Some PackageCalls require additional information specified in the parameters. For example, fMRILogResponseEvent requires an additional parameter: the name of the E-Object that corresponds to a response event.

When designing an experiment, be sure to check the PackageCall Properties for any additional parameters. When a PackageCall has “<>” for the parameters, then more information must be added. The angle brackets (“<>”) are placeholders for parameter values that are specific to your machine or experiment configuration.

Some PackageCalls require additional information for a parameter, but the parameter does not appear in angle brackets. In these cases, if you generate the experiment but have not specified the required parameter value(s) then an error occurs. In the example below, the error message indicates what Routine generated the error; however, this might not always be the case (e.g., a more generic error occurs). You can find out which Routine generated the error by looking in the Full Script window. The Routine with the missing parameter is highlighted. Additionally, you can scroll up in the Full Script window to determine which Procedure the error occurred on.



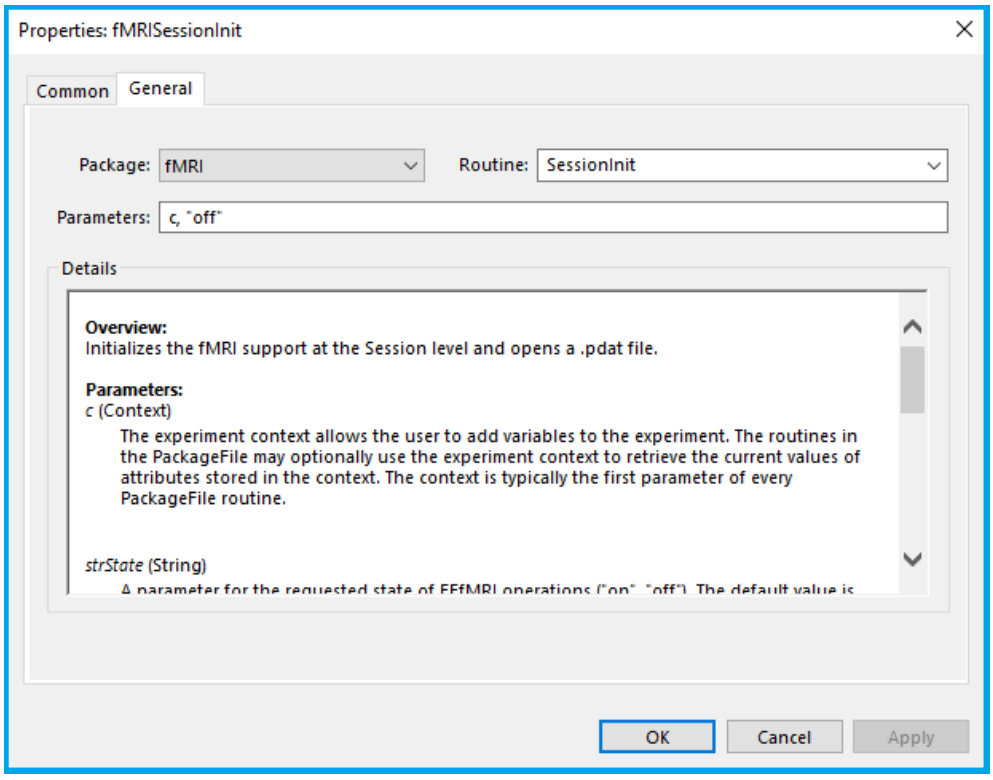

**NOTE**: *When designing your experiment, we recommend reading through the Details of the PackageCall’s properties.* ***Details*** *provides descriptions and remarks of all required and optional parameters.*

## 4.3. Running an Experiment Without EEfMRI Enabled

Occasionally, you may want to run your experiment without calling the EEfMRI PackageCalls (i.e., EEfMRI-disabled). It is often beneficial to run the experiment under those conditions whenever you are designing and testing your experiment. This can be helpful for determining if your critical stimuli are not presenting correctly or if there is a runtime error in your experiment. Whenever you want to run with EEfMRI disabled, you need to alter the fMRISessionInit PackageCall in your experiment.

1) ***Open*** the PackageCall **fMRISessionInit**.

2) ***Change*** the **strState** from "**on**" to "**off**".



# Appendix A: Samples and Tutorials

The following table contains a list of all the Samples and Tutorials included with E-Prime Extensions for fMRI. The folder the experiment is in and a short description are included.

| Sample / Tutorial | Folder | Description |
|---|---|---|
| fMRIPress | Samples | fMRIPress is a reaction time test which presents a fixation followed by a Stimulus prompting to respond with a certain finger. The right hand's fingers map to keys 1, 2, 3, 4, 5 beginning with the right-hand thumb, and the left hand's fingers map to keys 6, 7, 8, 9, a beginning with the left-hand thumb. |
| fMRIPress_Celeritas | Samples | fMRIPress_Celeritas is the same experiment as fMRIPress with the exception being it is configured for Celeritas input. |
| MapperOne | Samples | MapperOne is a collection of 5 research-oriented fMRI tasks designed to map eloquent cortical areas of the brain. The task battery is intended to reliably produce activation patterns in the areas of the brain associated with visual, motor, cognitive, and emotional function. This experiment is discussed more in *Appendix B: MapperOne Experiment*. |
| MapperOne_Celeritas | Samples | MapperOne_Celeritas is the same experiment as MapperOne with the exception being it is configured for Celeritas input. |
| Press | Tutorials | Press serves as the starting point for EEfMRI Tutorial 1. |
| Press_Celeritas | Tutorials | Press_Celeritas is the same experiment as Press with the exception being it is configured for Celeritas input. |

# Appendix B: MapperOne Experiment

***Summary:***

This section of the manual is designed to describe the **MapperOne.es3** sample paradigm included with the EEfMRI installation. The experiment is a collection of research-oriented tasks designed to map various areas of the brain. The task design is based on the published work of Drobyshevsky et al. 2006 and is intended to reliably produce activation patterns in the areas of the brain associated with visual, motor, cognitive, and emotional function. The task resolution is set at 1024 X 768, and the included image resources are the same resolution, to replicate the original published work.

***Task Overview:***

There are five tasks contained in this set divided into four runs. To save time the visual and motor tasks have been combined. All the tasks except the emotional pictures task consist of a probe block (18 seconds duration) alternated with a control block of equal length, and an inter-stimulus interval (ISI) of two seconds.

Each run consists of eight blocks and lasts for two minutes and 30 seconds (including two discarded acquisitions. Our recommended scanning parameters for 1.5T scanners are: TE = 35 ms, FOV = 20 cm, TR = 3000 ms, 37 slices, and the discard of the two initial volumes collected per sequence. These timing perimeters give you a total of 50 volumes. The tasks are set up for the instructions to verbally be given to the participant at the beginning of each block. We highly recommended that the participant practices the task prior to the actual scan.

The emotional pictures task consists of a baseline block, a control pictures block, a novel pictures block and a rest block. Each block condition repeats four times and the run lasts three minutes and ten seconds. The control and novel picture blocks each repeat four times (18 seconds each). The rest block repeats eight times (48 seconds total). The entire runs last three minutes and 12 seconds.

***Recommended readings:***

We recommend reading the research article by Drobyshevsky et al. to get a full understanding of the MapperOne tasks:

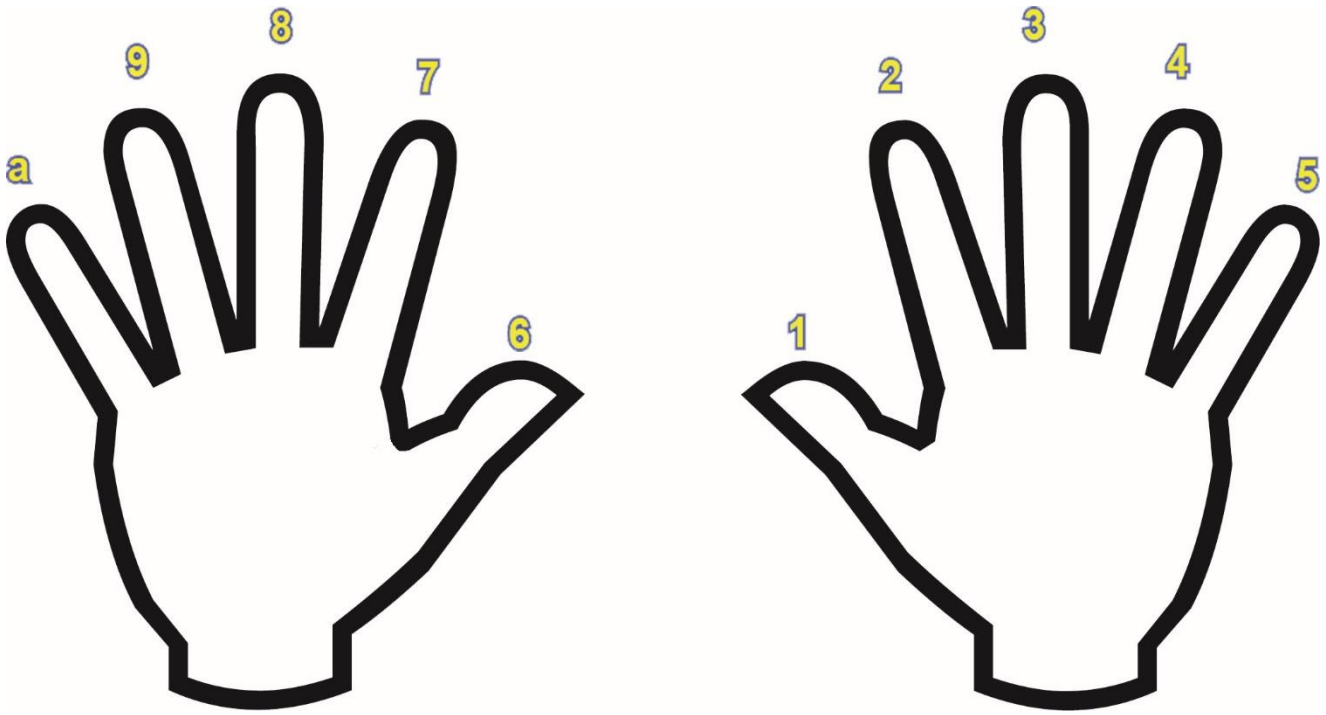
A Rapid fMRI Task Battery for Mapping of Visual, Motor, Cognitive and Emotional Function

***Seizure Warning:***

Some individuals (about 1 in 4,000) may have seizures or blackouts triggered by light flashes or patterns even if they have never had a seizure before. Risk of seizure may also increase in patients with epilepsy, those with a history of seizures, or in patients with a recognized risk for seizure (such as head trauma, metabolic disorders, alcohol and drug withdrawal, and CNS infections). The MapperOne, Visual - Motor Task presents quick flashing images.  We recommend that participants prone to seizures should not run that task.

***Visual - Motor Task:***

In this task, the participant passively observes a fixation for a period of time, a series of single digit clues, and a flashing checkerboard. Participants are instructed to press the corresponding button on their response unit (e.g., Keyboard, Celeritas, etc.). You should stress to the participant that TWO responses are required on each trial, one from each hand. This number identifies which finger should be used to make the response. From the participant's perspective, the fingers are numbered from 1 to 5, from the thumb to the pinky. For example, the correct response to the number '4' is to press the ring finger on each hand. However, the numbering for the right and left hand on the MotorTrialList object is as follows: the right-hand buttons are numbered 1, 2, 3, 4, 5 (1=thumb) and the left-hand buttons are numbered 6, 7, 8, 9, a (6=thumb).




The numbering shown above is required to enable the MotorProbe Object to uniquely identify each finger press.

During the flashing checkerboard pattern, remind the participant to remain still. The digit blocks and visual stimulation alternate eight times.

| | | |
|---|---|---|
| 6 seconds<br>Fixation | 18 seconds<br>Motor | 18 seconds<br>Checkerboard |

***Language Task (Verb Generation):***
In this task the participant passively observes a control stimulus (####) for a period of time and then observes a series of nouns. For each noun the participant is shown, he or she is to think of a corresponding verb and say the verb silently to themselves.

| **Trial** | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **Noun** | truck | apple | key | dog |
| **Silent Verb (examples)** | drive | eat | unlock | bark |

In trials one and three, the corresponding verb could have been “honk” and “lock”, respectively. Be sure to inform the participant there are no right or wrong answers. The important thing is that they silently say the verb.

| | | | | |
|---|---|---|---|---|
| 6 seconds Fixation | 18 seconds ‘###’ | 18 seconds See noun Silently say verb | 18 seconds ‘###’ | … |

***Working Memory (Verbal N-Back)***
In this task, the participant performs two alternating letter comparison tasks. During the first comparison task, Zero-Back or target letter ‘X’, the participant sees a series of letters shown in **red**. The participant’s task is to look for the target letter ‘X’. Every time the letter ‘X’ is shown, the participant should respond by pressing the ‘2’ key (index finger on their right hand). If the letter shown is not an ‘X,’ the participant should respond by pressing the ‘3’ key (middle finger on their right hand).

If the letters are shown in **yellow,** then the participant performs a slightly harder task called Two-Back or letter skips task. During the two-back task the participant needs to keep the previous two letters in their working memory, compare them to the letter presented in the current trial and make a judgment if the letter in the current trial is the same letter as the letter presented two trials ago. If the letter shown in the current trial matches the letter shown two trials back, the participant should respond by pressing the ‘2’ key (index finger on their right hand). If the letter does not match the letter shown in the previous two trials the participant should respond by pressing the ‘3’ key (middle finger on their right hand). You may want to remind the participant that this task requires a response on EACH trial. Participants are not responding only to the presence of The Target; they must respond ‘3’ for any non-target trial.

The table below shows a series of letters. The only target occurs on trial five because trial three (the trial two previous) is the same letter; The Target letter "skips" a letter.

| **Trial** | 1 | 2 | 3 | 4 | **5** | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **Letter** | G | J | J | K | **J** | G | K |
| **Correct Response** | 3 | 3 | 3 | 3 | **2** | 3 | 3 |

The Zero-Back (**red**) and Two-Back (**yellow**) blocks will alternate several times.

| **6 seconds Fixation** | **18 seconds Zero-Back** | **18 seconds Two-Back** | … |
|---|---|---|---|

***Emotional Pictures***

In this task, the participant passively observes a fixation for six seconds then views a block of control pictures lasting 18 seconds. This is followed by another rest of six seconds. Then a novel picture block is presented for 18 seconds. This pattern repeats for three minutes and 12 seconds.

| 6 seconds Baseline | 18 seconds Control Pictures | 6 seconds Rest | 18 seconds Novel Pictures | 6 seconds Rest | 18 seconds Control Pictures | 6 seconds Rest | 18 seconds Novel Pictures | 6 seconds Rest | ... |
|---|---|---|---|---|---|---|---|---|---|

We cannot legally distribute the control and emotional pictures used in the Drobyshevsky et al. publication. The original experiment used images from The International Affective Picture System (IAPS).

IAPS provides normative ratings of emotion (pleasure, arousal, dominance) for a set of color photographs that provide a set of normative emotional stimuli for experimental investigations of emotion and attention.

You can request a copy of the stimuli via this website:

http://csea.phhp.ufl.edu/media/iapsmessage.html

The .bmp files used in the Drobyshevsky et al. 2006 study are:

1121, 1390, 1660, 1670, 1720, 1810, 1811, 1812, 2030, 2070, 5260, 5390, 5500, 5875, 5890, 5920, 6150, 7000, 7002, 7004, 7009, 7010, 7030, 7034, 7050, 7090, 7130, 7150, 7200, 7233, 7235, 7270, 7351, 7500, 7595, 8161, 8162, and 8300.

# Appendix C: The .pdat File Form

The .pdat is the output file produced by the EEfMRI PackageFile. This file contains the timing information created by the EEfMRI PackageCalls. The .pdat file is created in the same folder as the experiment. It is a tab delimited data file and can be opened in several formats. The table below provides definitions of the common columns contained in the file.

| .pdat File Column Definitions | |
|---|---|
| **Columns** | **Descriptions** |
| OnsetTime | The E-Prime clock time related to the onset of a critical event. This will depend on what PackageCall is related to that event. For a LogEvent it will be the object passed into the object call, but for RunBegin it will be the timestamp of the RF Trigger Pulse/trigger key press. |
| TargetOnsetTime | The scheduled time related to the onset of a critical event. This will depend on what PackageCall is related to that event. For a LogEvent it will be the Object passed into the Object call, but for RunBegin it will be the timestamp of the RF Trigger Pulse/trigger key press. |
| RunTime | The time of the event relative to the start time of RunBegin. |
| RecordID | String that identifies what type of condition has been logged. This information comes from the fMRILogEvent, fMRILogResponseEvent and fMRILogUserEvent PackageCalls. |
| Duration | Length of time in milliseconds that the condition lasted. |
| ConditionID | User defined string that uniquely identifies the condition being logged. |
| ConditionLevel | Number that indicated what Log Level in the hierarchy the event is assigned in the experiment. |
| RESP | Returns the last (i.e., for single response input) or entire response (i.e., for multiple response input) collected by the Object. |
| CRESP | Returns the correct response associated with the input collected by the Object. This property is generally set internally according to the value in the Correct field in the Response Options for an object but may be set via script at runtime. |
| ACC | Reflects the accuracy of the response logged by the input object. ACC is based on a comparison of the RESP and CRESP properties. |
| RT | Returns the reaction time of the last input collected by the Input Object, timed relative to the start of the input. |
| RTTime | Returns the reaction time of the input relative to the start time for the experiment. |
| User1, User2, etc. | User defined values. These are created by the fMRI_LogUserEvent PackageCall |

**NOTE**: *The variables CRESP, RESP, ACC, and RT specify information about the response made, accuracy, and reaction time.*

# Appendix D: Add Celeritas to the Experiment

*Open the Devices tab to add the response device to the experiment and edit its properties.*

The EEfMRI Sample and Tutorial experiments are set up to use the keyboard as the participant's response device. These experiments can be completed as written with no modifications. Using the keyboard to make responses is typically done when developing and testing experiments in the lab outside of the scanning center. However, a different response device must be used for experiment data collection in the magnet. PST provides Celeritas, a MR-compatible hardware system for capturing button presses.

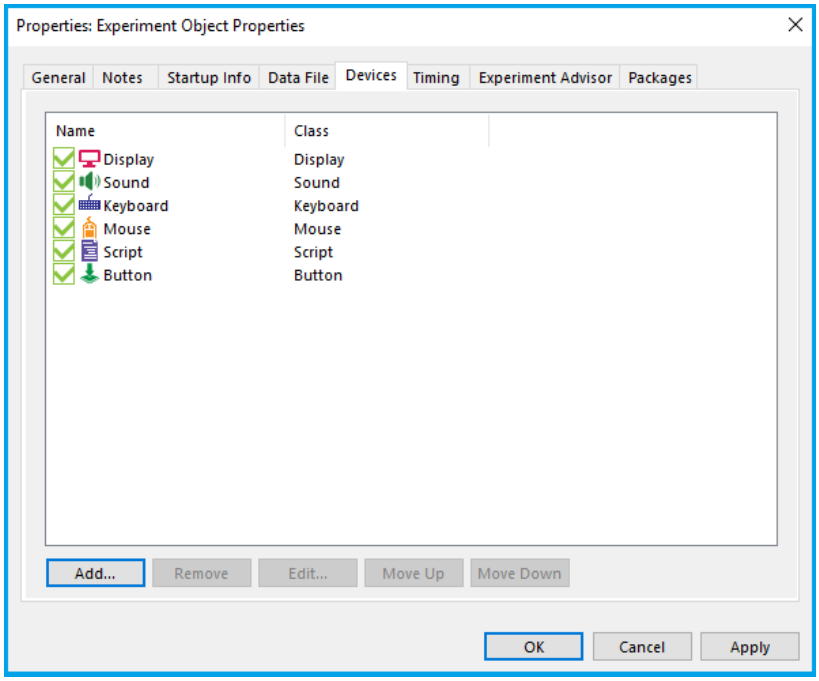
This Appendix provides a brief overview of what steps are necessary to add Celeritas to the experiment.

**NOTE**: *Celeritas-enabled versions of the Samples and Tutorials are provided with your EEfMRI system (see Appendix A: Samples and Tutorials).*

**NOTE**: *If you enable the Celeritas device in the experiment, you must have the Celeritas software installed on your computer to load the Celeritas-enabled experiment into E-Studio. You cannot share your Celeritas-enabled experiment with anyone who does not also have Celeritas installed.*

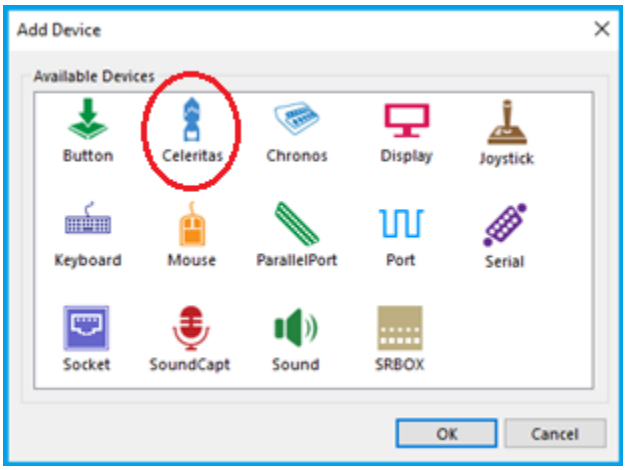
**NOTE**: *The following steps assume Celeritas is connected with 2 BRU's attached.*

1) ***Click*** on the **Devices** tab of the **Experiment Object Properties** dialog.



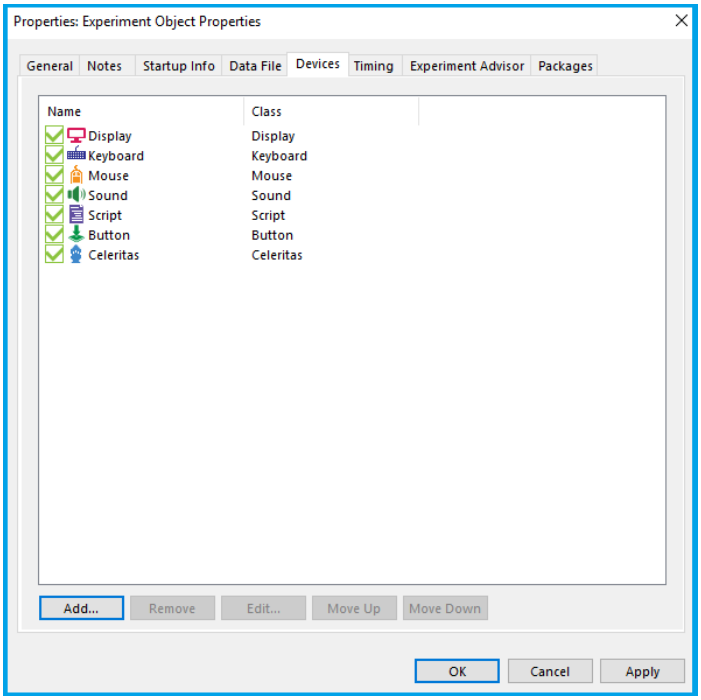

2) ***Click*** the **Add...** button.

3) ***Select*** the **Celeritas** Device in the **Add Device** dialog.



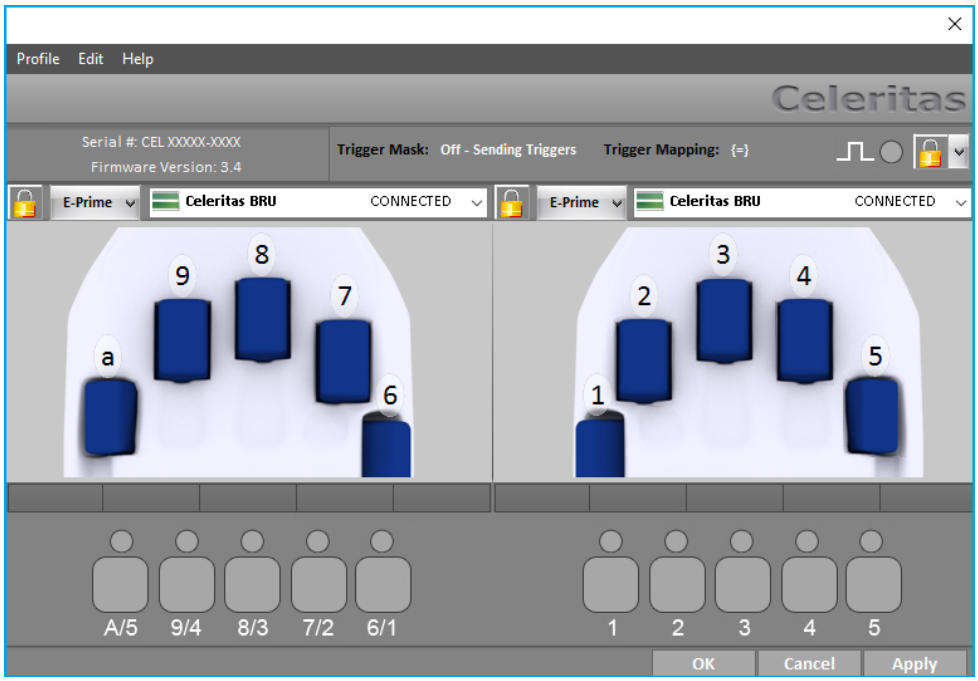

4) ***Click*** the **OK** button to dismiss the dialog.

5) ***Verify*** the **Celeritas Device** is listed last under the **Name** column and is checked.



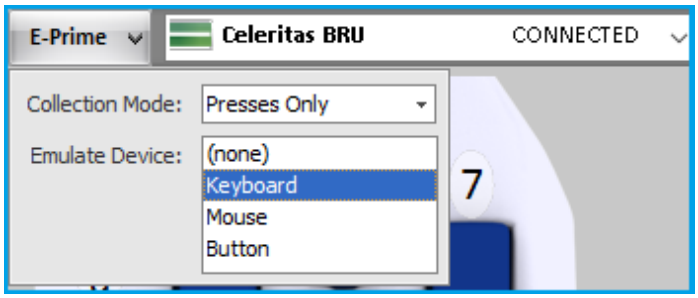

**NOTE:** *The order of devices from top to bottom is the order in which the devices become initialized in the beginning of an experimental run. Do not move another Device to the top as the Display Device must open prior to the remaining Devices. If this order is not followed, the experiment terminates.*

6) ***Click*** on **Celeritas** and ***click*** the **Edit** button to configure **Celeritas**.




7) ***Locate* one of the three lock buttons** and ***click*** it**.**

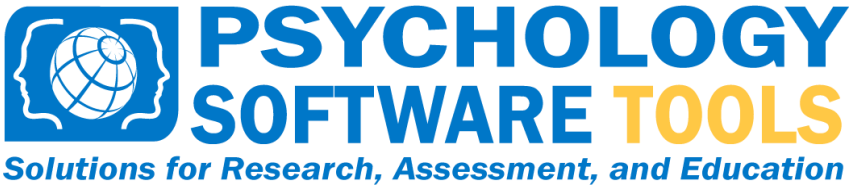
8) ***Click*** the **E-Prime dropdown** and ***select* Keyboard** to **emulate Celeritas into the Keyboard**. ***Click*** the **E-Prime dropdown** to **close** it.




**NOTE**: *If you chose not to emulate the keyboard and you wish to use Celeritas for response collection, you must add a Celeritas Input Mask to any response collection object or additional Input Masks to your response collection objects will be needed to run the experiment.*

9) ***Click*** the **OK** button accept the **Celeritas Device Properties** dialog. ***Click*** the **OK** button to dismiss the **Add Device dialog**.

# 5. 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 Service and Support:
https://support.pstnet.com