E-Prime® Extensions for fMRI™

User Manual

For Research and Education Only

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

PST-100780

E-Prime Extensions for fMRI User Manual
PST-100785
Rev 6

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

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

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

The E-Prime Extensions for fMRI (PST-100780) is for research and educational purposes only.

Table of Contents

Chapter 1: **Getting Started Guide**

1.1 Compatibility

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

For a summary of E-Prime's new features, review the *E-Prime New Features-Reference Guide* that can be accessed through the Start menu: Start > All Programs > E-Prime 2.0 > Documentation, or through the Help menu in E-Studio: Help > Documentation > New Features-Reference Guide. See the Knowledge Base article 2596 - INFO: How to Determine Version Number of E-Prime or Components of E-Prime for details.

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

1.2 E-Prime Extensions for fMRI 2.0 Overview

The EEfMRI software 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 a greater degree of timing and stimulus control by synchronizing the start of an experiment 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 run time. 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 guide will show you how to install the EEfMRI software, explaining how to structure an EEfMRI-enabled E-Prime experiment, provide descriptions of the tools for which EEfMRI is equipped, and provide 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 will minimize your learning time and allow you to take advantage of the available tools to produce functional paradigms ready for use with the fMRI system.

Recommended readings:
It is recommended that you have read and worked through the tutorials included in the *E-Prime Getting Starting Guide* before beginning this tutorial.

1.3 Software Installation

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

Install E-Prime 2.0

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

Installing E-Prime Extensions for fMRI 2.0

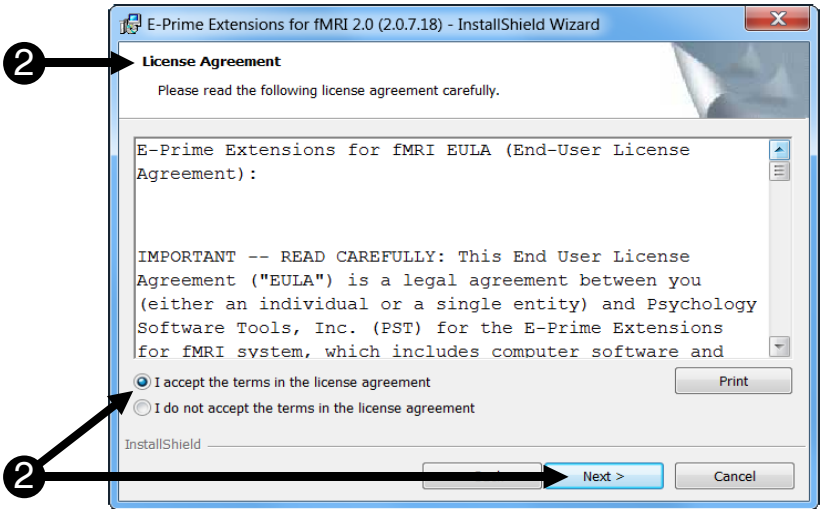
Before continuing, be sure that you have administrative rights to install this software on the computer. If you do not have administrative rights, you will be unable to install E-Prime Extensions for fMRI. If you are unsure of your administrative privileges, contact your System Administrator.

⚠ **NOTE:** *EEfMRI 2.0 is compatible with E-Prime 2.0 Professional SP1 only.*

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

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

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



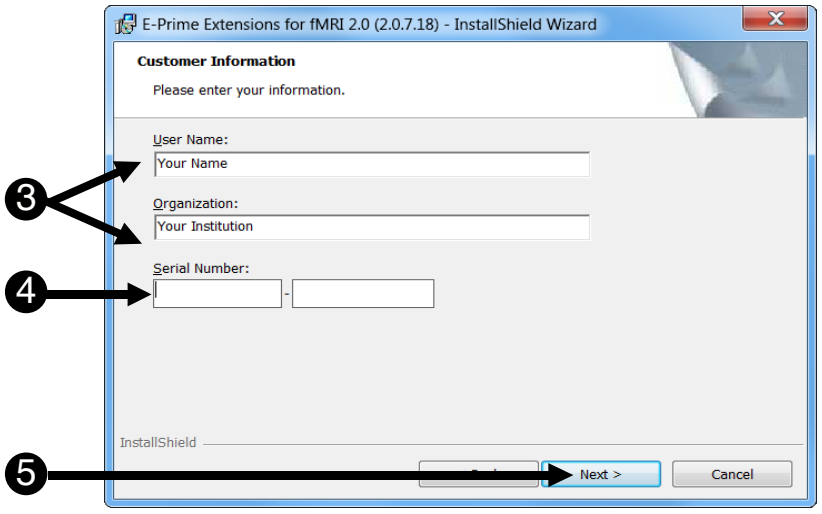
3) ***Specify* Your Name** and **Your Institution** or check with your system administrator for appropriate information.

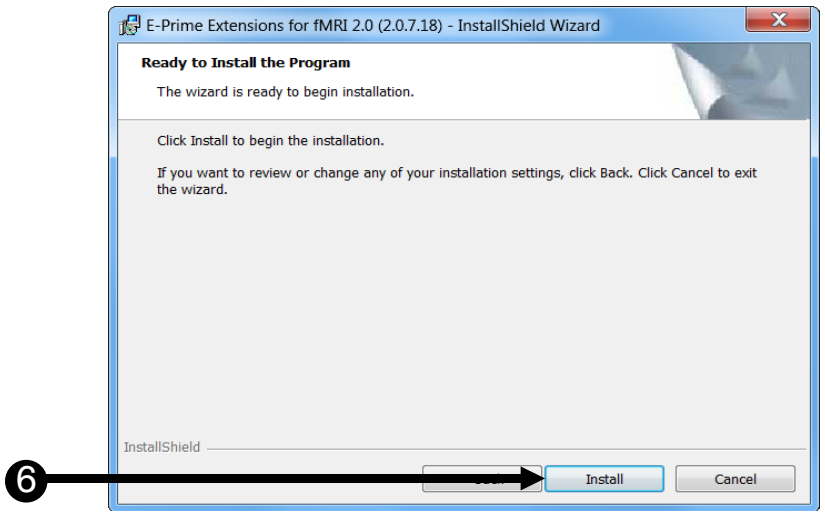
4) The **Serial Number field** must be ***complete*** to obtain access to online **Product Service and Support**.

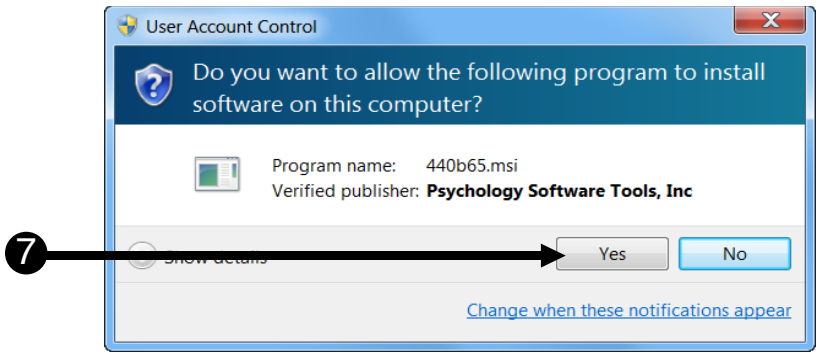
5) ***Click* Next** to begin transferring files to your computer.

6) ***Click* Install**.

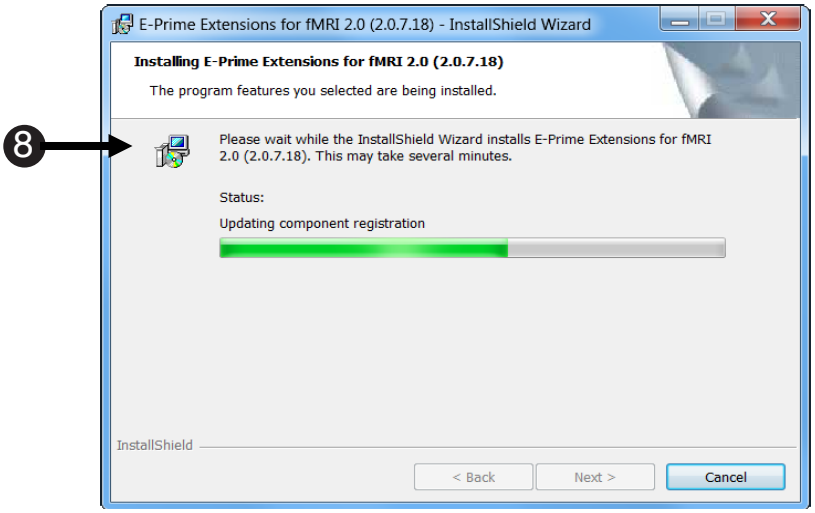
7) ***Click* Yes** to continue the installation.





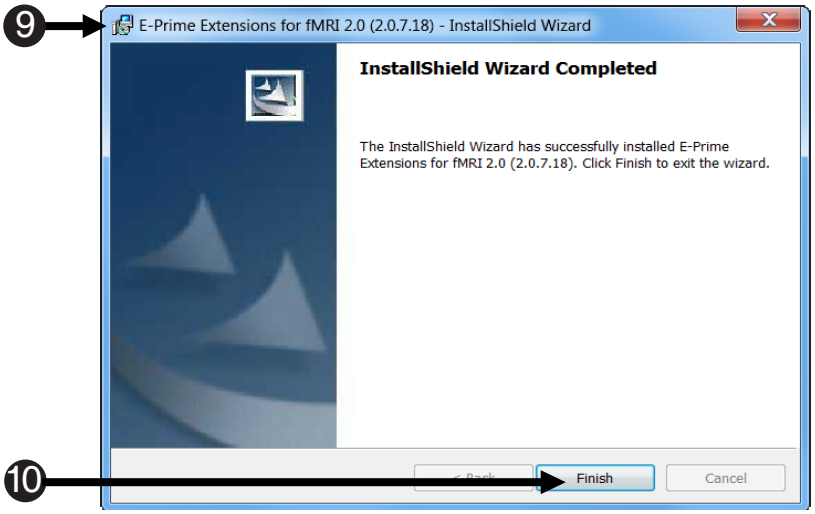


8) ***Wait*** while the installer configures the software.



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

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



1.3 Software Installation (continued)

Finding the "My Experiments" Folder

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

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

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

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

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

1.4 Product Service and Support

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

https://support.pstnet.com

Registration requires a valid E-Prime serial number, EEfMRI serial number, and e-mail address. At the support site, you will also find a Knowledge Base including release notes and a compilation of frequently asked questions. The support site also includes E-Prime sample paradigms that are available for you to download. There is an additional support forum you can use to post general questions about E-Prime 2.0 SP1 and EEfMRI. For additional details: https://support.pstnet.com.

1.5 Resources

1) Check PST's web site for additional resources: http://www.pstnet.com/eefmri.
2) EEfMRI users are entitled to one year of Silver Support E-Prime technical support via PST's Product Service and Support web site.
   For additional details: http://www.pstnet.com/eprime.cfm?tabID=Support

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

2.1 Overview

This chapter consists of tutorials that will guide you through the steps necessary to add the most commonly used features of the EEfMRI software to an existing E-Prime experiment.

In order to complete the tutorials, you will need a computer with E-Prime and the EEfMRI software already installed. If you have not installed E-Prime please stop now and complete the installation, and if you have not installed EEfMRI, please refer to section **1.3 Software Installation, (Page 7)** of this manual.

The tutorials assume you are familiar with using E-Prime to build behavioral experiments. If you are new to using E-Prime, we suggest that you work through all tutorials included in the *E-Prime Getting Started Guide* prior to beginning this tutorial series.

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

Summary:

Incorporating EEfMRI support into an existing E-Prime experiment primarily involves adding the EEfMRI Package File to the experiment and placing the EEfMRI PackageCalls at the appropriate locations experiment structure.

During this tutorial, you will add EEfMRI support to the Press.es2 sample experiment that is installed by default with the EEfMRI package. The Press.es2 sample 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.es2 experiment, it is recommended that you load, review and run the sample experiment before beginning the tutorial (e.g. in order to develop an understanding of the overall design of the experiment and how it operates prior to further modifications).

Goal:

This tutorial illustrates how to add the EEfMRI PackageCalls into the Press.es2 sample experiment included with EEfMRI. When you have completed this tutorial, you will have a basic “EEfMRI-enabled” paradigm.

Overview of Tasks:

- Load Press.es2 and resave it as fMRIPress.es2.
- Add the EEfMRI Package File to the Experiment Object.
- (Optional): Add SRBox to Experiment device properties to enable responses from the FOBRS button unit.
- (Optional): Add Celeritas to Experiment device properties to enable responses from Celeritas.
- 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 and sync the onset of the stimulus presentation sequence with the scanner trigger pulse.
- Add the fMRIRunEnd PackageCall to designate the end of the functional scan.
- Verify the overall experiment structure and run the experiment.

Estimated Time: 20-30 minutes

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

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

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

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

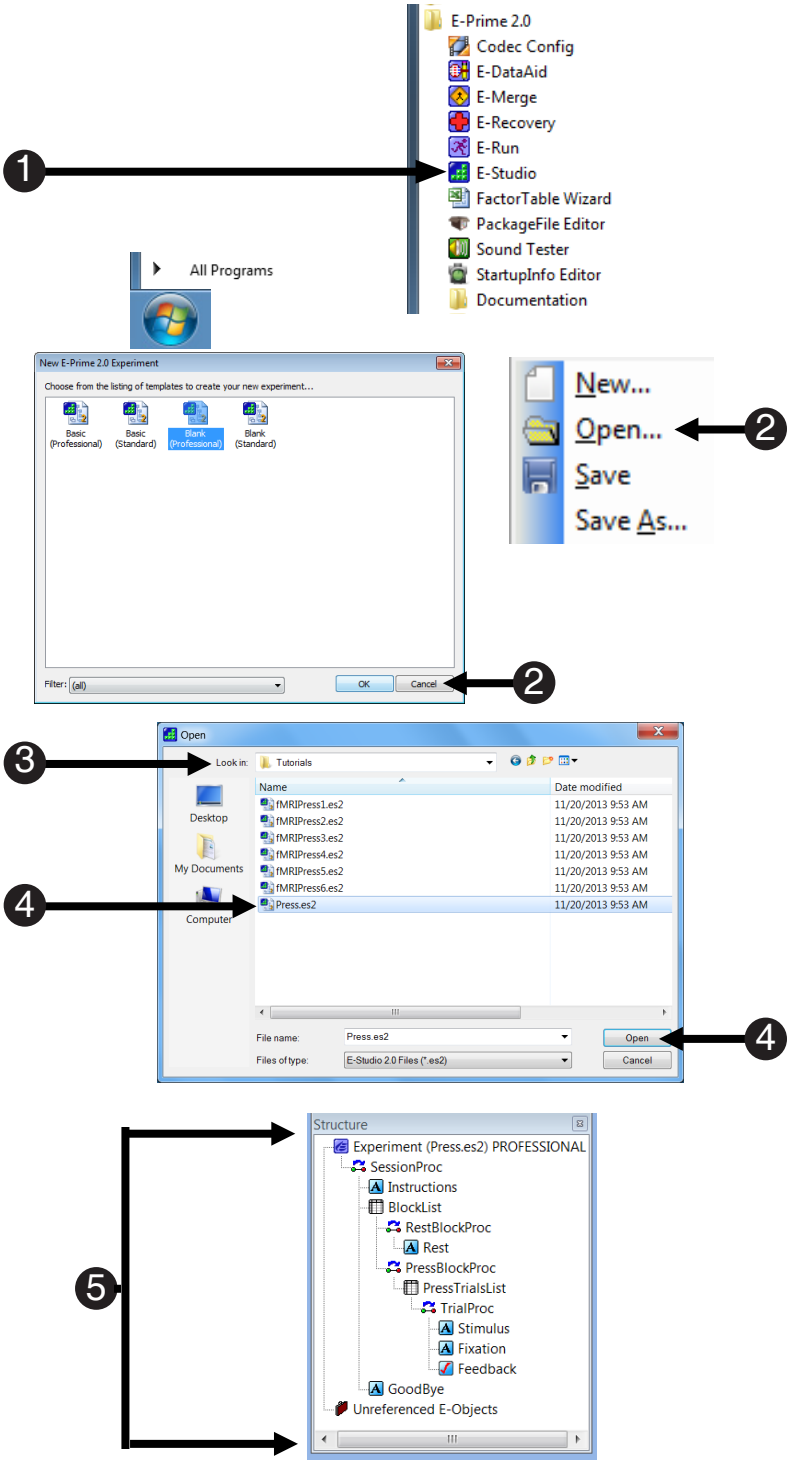
2) ***Click*** the **Cancel** button. ***Select*** **File** > **Open**.

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

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

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

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



Task 2: Save the experiment under a new name

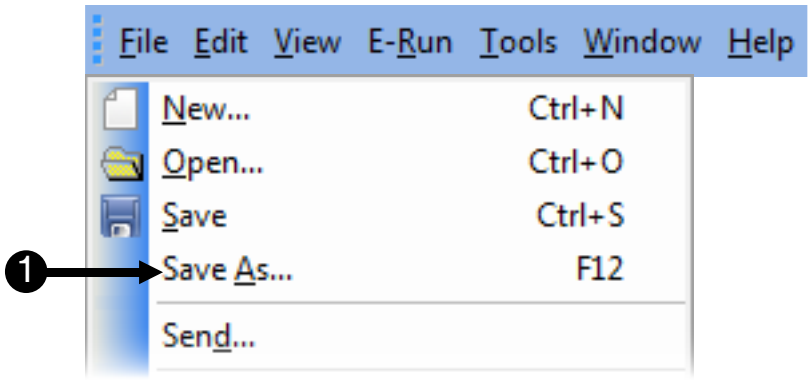
Save the Press.es2 experiment in the same folder under the new name "fMRIPress.es2."

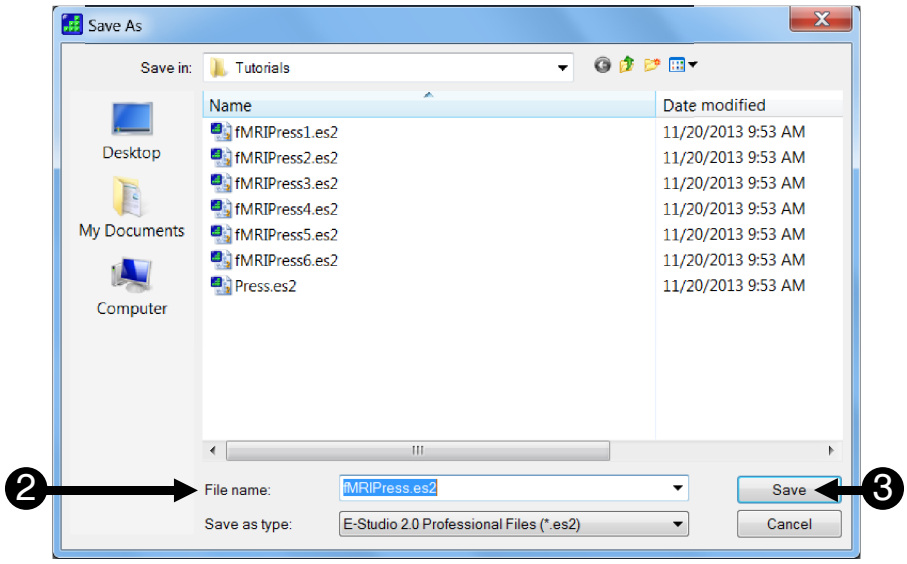
Rename the experiment and save it in the same folder ("…\My Experiments\fMRI\Tutorials") so that any resources references within the experiment will remain valid and can be reused.

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

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

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





Task 3: Add the EEfMRI Package to the Experiment Properties

Open the Properties dialog for the Experiment Object and use the Packages tab to add the EEfMRI Package File to the experiment.

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

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

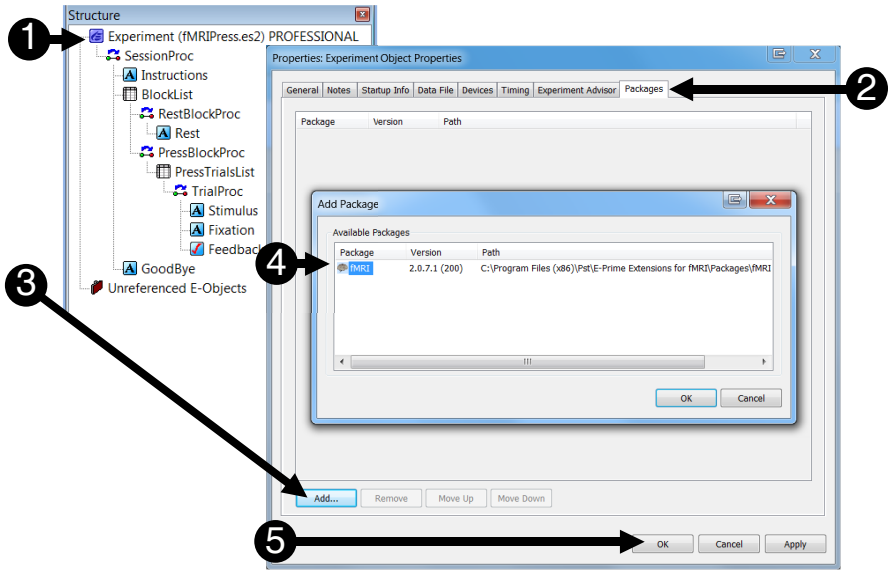
2) ***Click*** on the **Packages** tab of the **Experiment Object Property Pages**.

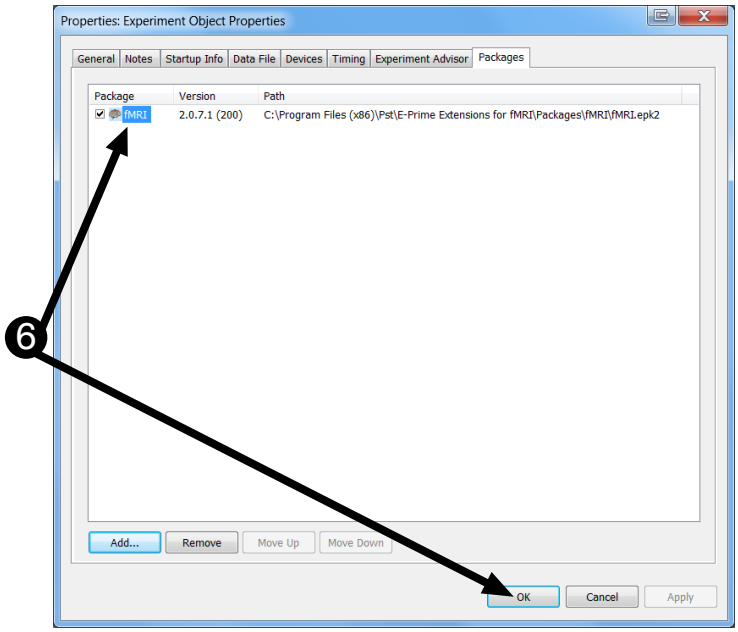
3) ***Click*** the **Add…** button.

4) ***Select*** the **fMRI Package File** in the **Add Package Property Pages.**
The version number may differ from what is shown here.

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

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





Task 4: (optional): Add a response device to the Experiment Properties

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. 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 two MR-compatible hardware systems for capturing button presses, Celeritas and FOBRS. In order to use either device with the experiment, the devices must first be added to the experiment.

The tutorials and/or sample experiments can be completed as written and tested using the computer keyboard only. No additional steps are required beyond what is provided in the tutorials. Alternatively, if you want to use your Celeritas or FOBRS device with the tutorials and/or samples,or if you want to learn more about these response devices, please see **Appendix C: Celeritas and FOBRS Response Devices, (Page 79)**. The Appendix provides a brief overview of what steps are necessary to add these devices, and directs you to the appropriate place in the respective device's Operator Manual for detailed instructions.

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

Task 5: Add an EEfMRI PackageCall to initialize the system

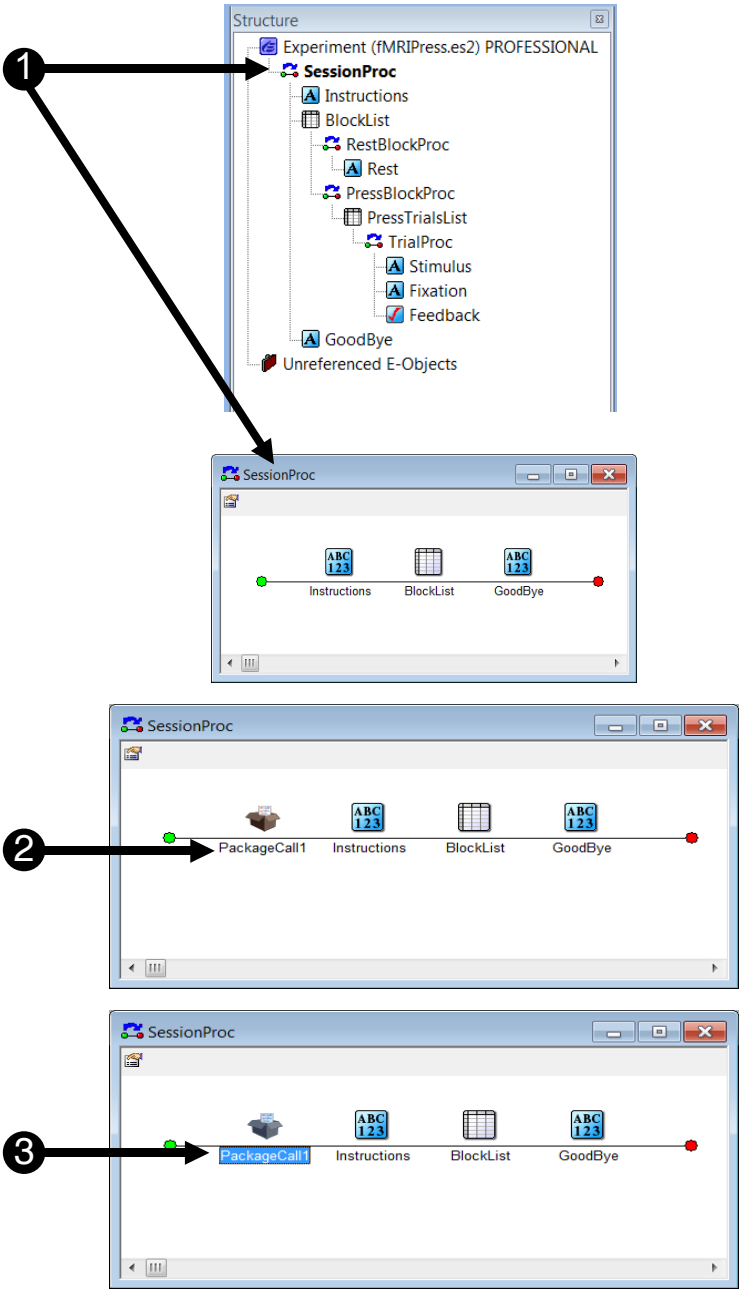
Add a PackageCall at the beginning of the SessionProc to initialize the system for use with EEfMRI. Name the object "fMRISessionInit."

The EEfMRI Package File must be initialized at the start of the experiment by making a call to the fMRI_SessionInit Routine. The preferred method to call a routine in a Package File is to drag a PackageCall from the E-Studio Toolbox, drop it at the desired location within the experiment, and edit its properties. (You can also make the call directly using E-Basic script within an InLine object.) When using the PackageCall method, we strongly recommended that you rename the object to reflect the specific Package File and routine that is being referenced within the object. When creating paradigms for use with EEfMRI, a common convention is to combine the prefix "fMRI" with the name of the routine being called. In this example, we name the PackageCall object "fMRISessionInit" because it calls the routine "fMRI_SessionInit".

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

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

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



Task 5 (continued): Add an EEfMRI PackageCall to initialize the system

Configure the fMRISessionInit PackageCall to call the SessionInit Routine of the fMRI Package File and accept the default Parameters list.

The Properties dialog of the PackageCall is used to specify which Package File and Routine are to be called in each instance. After a Package is selected within the interface, the Routine dropdown list will be populated with all of the routines contained within the package. After you select a Routine, the Parameters field will be set to the default parameters and the Description field will be filled with the text that the Package File author included for the selected routine. You can refer to the Description field for information about each parameter in the list (any parameter in double quotes indicates string data). The fMRISessionInit PackageCall includes parameters that allow you to turn EEfMRI support on/off and to create a .PDAT file as output that is named by default. This name can be user defined, but the default should be sufficient for nearly all experiments (details of the .PDAT file will be discussed in **Appendix B: .PDAT File Format, (Page 78)**.

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

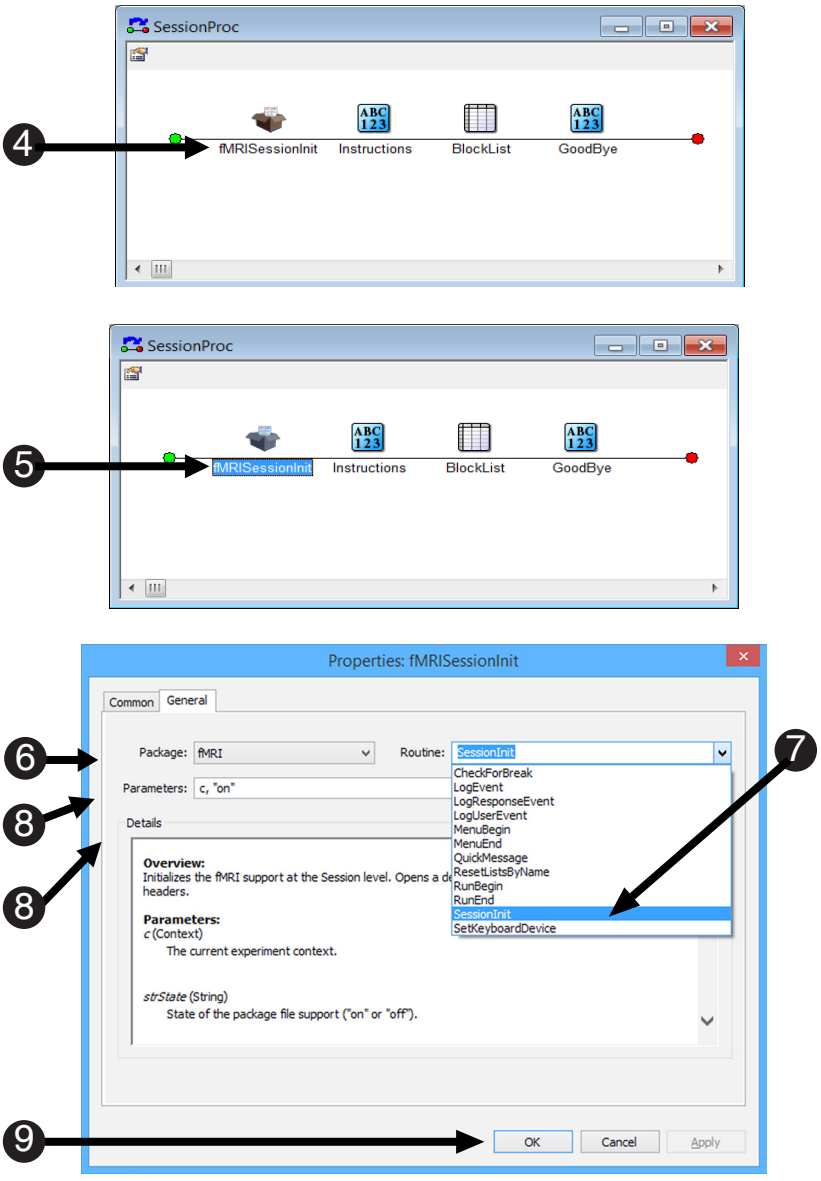
5) ***Double click*** the **fMRISessionInit** PackageCall object on the **SessionProc** to display its **Properties** dialog.

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

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

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

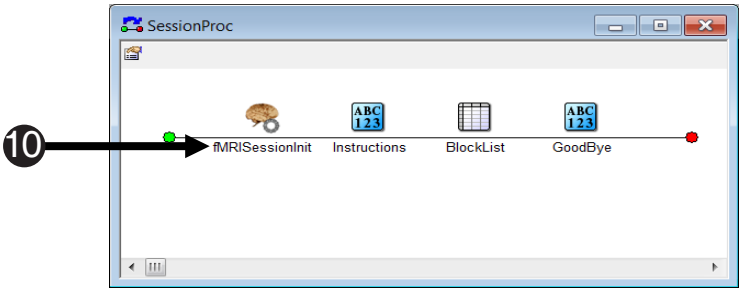
9) ***Click*** the **OK** button to accept the changes and dismiss the dialog.



Task 5 (continued): Add an EEfMRI PackageCall to initialize the system

Configure the fMRISessionInit PackageCall to call the SessionInit Routine of the fMRI Package File and accept the default Parameters list.

10) If the Package File defines an icon for the routine it will replace the default PackageCall icon, otherwise the icon associated with the entire Package File itself will be used.



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. Name the object "fMRIRunBegin."

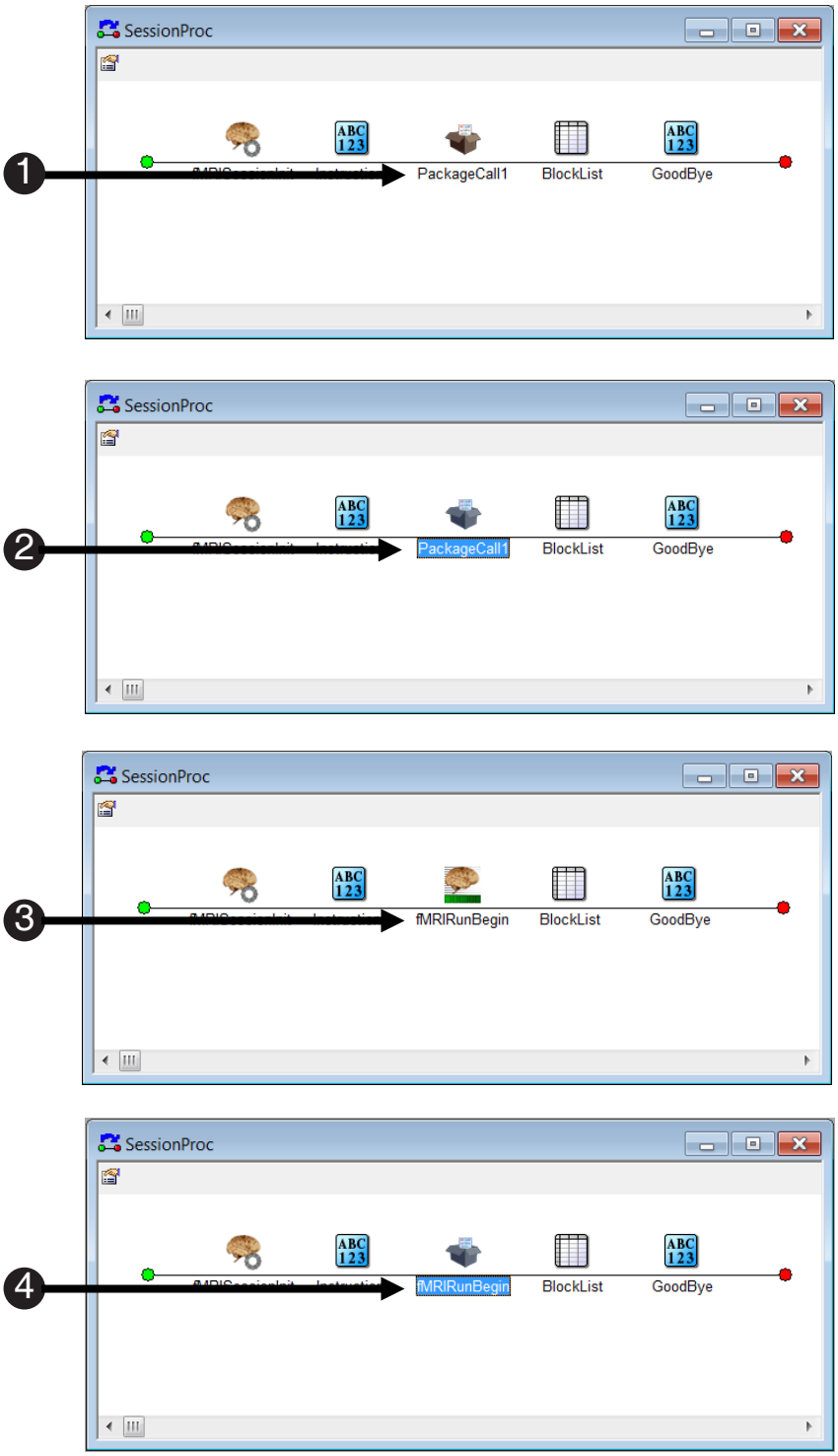
The fMRIRunBegin PackageCall marks the beginning of the functional scan and when properly placed, will synchronize the onset of the stimulus presentation with the scanner trigger pulse (assuming appropriate interface hardware is in use). This call will write critical timing information to the .PDAT file and must be matched with a corresponding fMRIRunEnd PackageCall for proper operations.

1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it in the **SessionProc** procedure ***after*** the **Instructions** object and ***before*** the **BlockList** object.

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

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

4) ***Double click*** the **fMRIRunBegin** PackageCall to display its **Properties** dialog.



Task 6 (continued): Add the fMRIRunBegin PackageCall

Configure the fMRIRunBegin PackageCall to identify the beginning of the functional scanning run and the start of the task stimulus presentation sequence. Name the object "fMRIRunBegin."

Edit the Run Condition to correspond to the name of the Run. In this example we will use "Press".

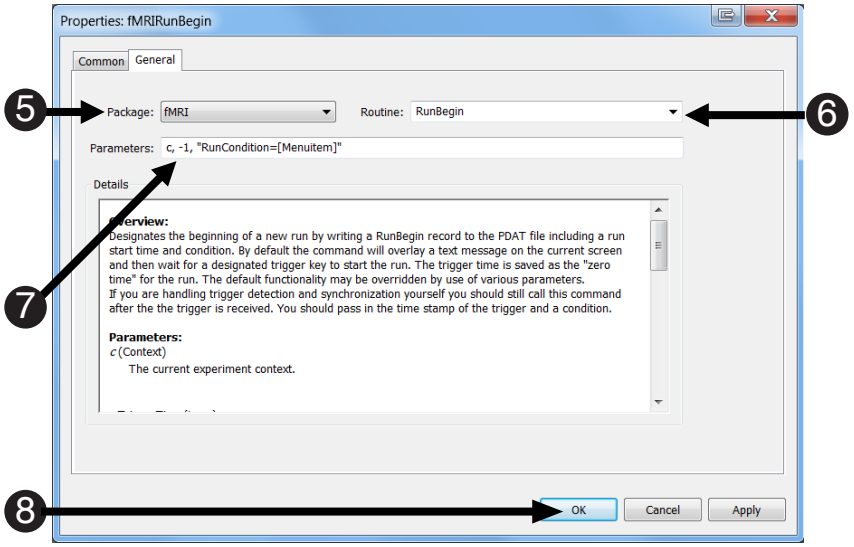
5) ***Select*** **fMRI** from the **Package** dropdown list.

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

7) ***Edit*** the **Parameters** to read *c, -1, "RunCondition=[MenuItem]" The last parameter is used to assign a name to the current Run to assist in later post processing.*

   Refer to the text in the ***fMRIRunBegin reference*** *on* ***Page 66*** *for more information about the parameters and options of this routine.*

8) ***Click*** the **OK** button.

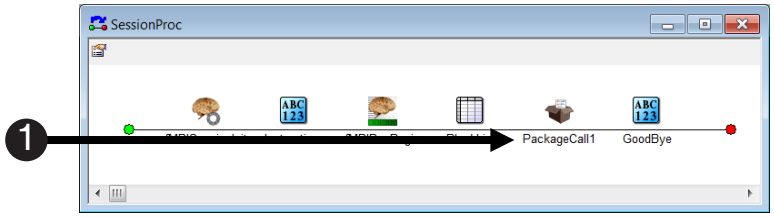


Task 7: Add the fMRIRunEnd PackageCall

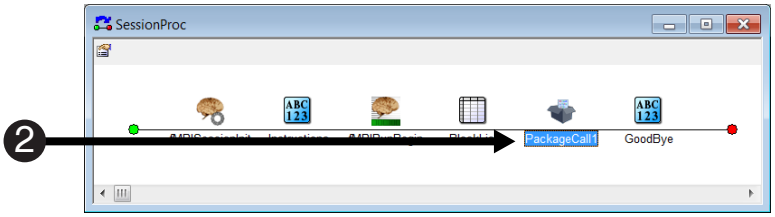
Add a PackageCall to identify the end of the functional scanning run. Name the object "fMRIRunEnd." Configure the PackageCall to accept the default parameters.

The fMRIRunEnd PackageCall is used to mark the end of the functional scanning run. By default, the Routine will also display 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.

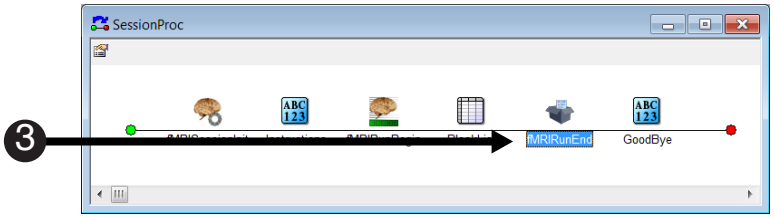
1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it on the **SessionProc** procedure ***after*** the **BlockList** object ***before*** the **Goodbye** object.



2) ***Click*** on the **PackageCall1** object to select it then ***press*** **F2** to rename the object. ***Rename*** the object to "**fMRIRunEnd**" as the new object name and ***press*** **Enter** to accept the change.



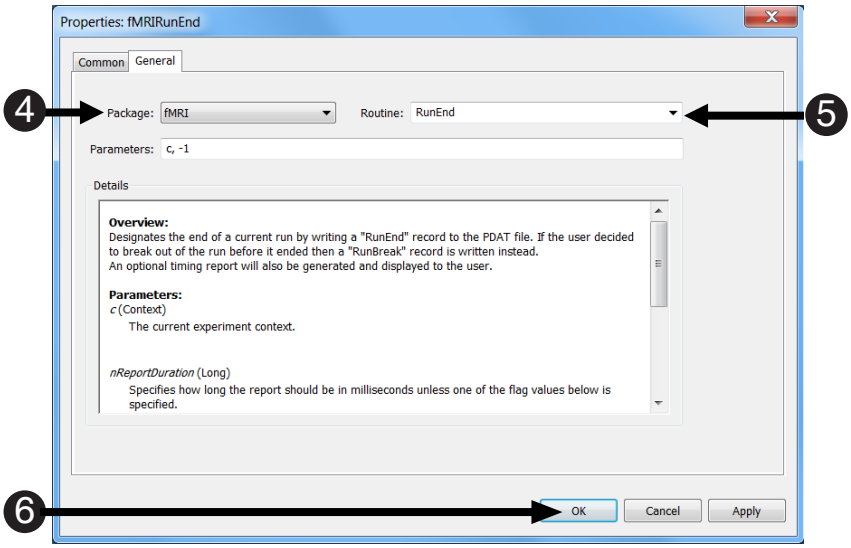
3) ***Double click*** the **fMRIRunEnd** object to display its **Properties** dialog.



4) ***Select*** **fMRI** from the **Package** dropdown list.

5) ***Select*** **RunEnd** from the **Routine** dropdown list.

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



Task 8: Run the experiment

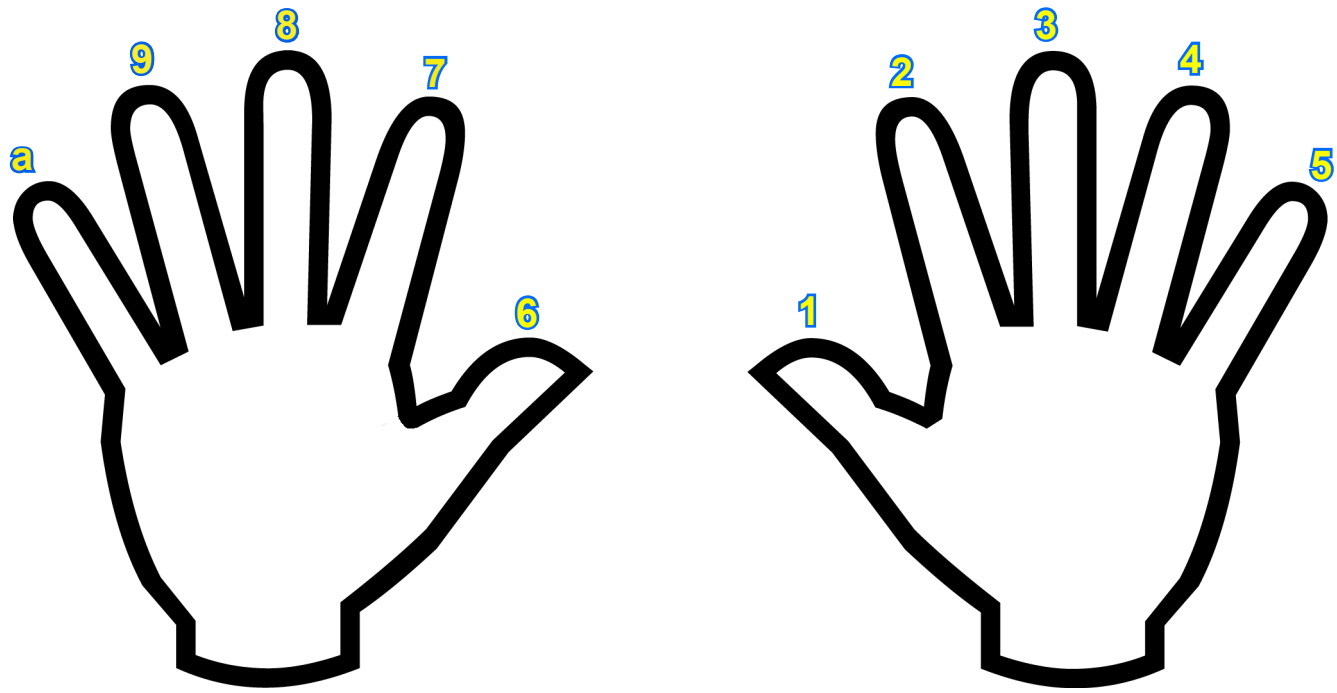
Run the experiment verifying the appropriate prompts are presented and no runtime errors are generated.

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 fully 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. The task itself is simple. 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 either the Celeritas or FOBRS response system. These MR-compatible hardware systems 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 more 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.



Task 8 (continued): Run the experiment

Run the experiment to verify that the appropriate prompts are presented and no runtime errors are generated.

The next steps will walk you through generating the experiment script, starting the experiment and entering Subject and Session numbers, and running the experiment.

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

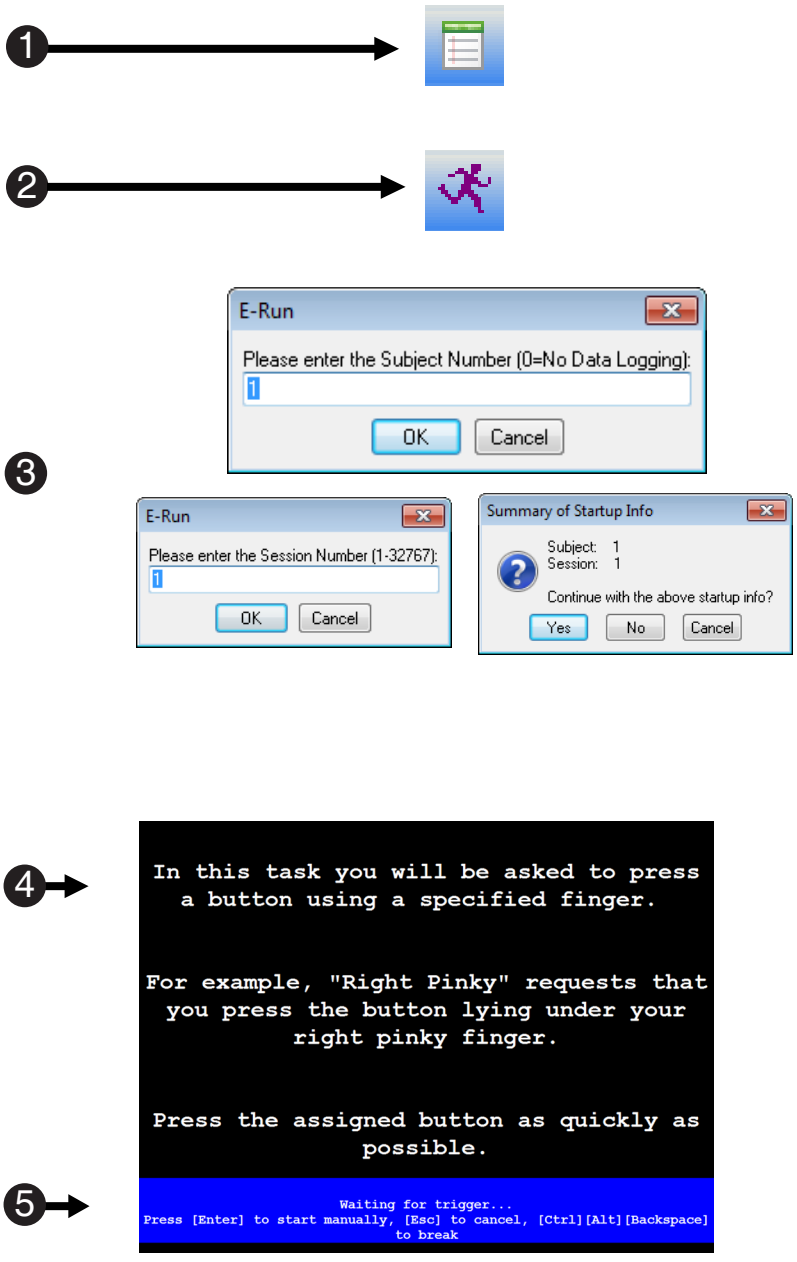
2) ***Click*** the **run icon** or ***press* F7** to ***run*** the paradigm.

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

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

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

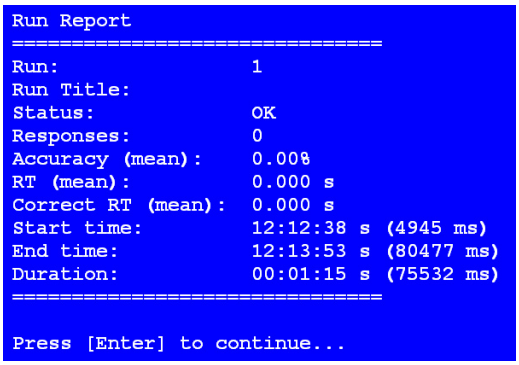
   If you test the experiment at the scanner, in conjunction with the appropriate interface hardware, the task will automatically begin when the technologist initiates the scan from the scanner console.



Task 8 (continued): Run the experiment

Review the End of Run Report to verify the task ran as expected.

After the experiment has terminated, you will see a screen like this:



This is 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 respectively.

The Accuracy field is an indication of the participant's percent accuracy. The RT field is the participant's mean response time across all trials and responses. The Correct RT field is the mean response time for the correct answers only. Note that all of the fields related to dependent measures, excluding times and duration, are displayed as zero in this example. This is because the tutorial did not yet add the fMRI PackageCalls required to collect this information. Even after these additional PackageCalls have been added you may also see these fields as zeros (0) if you do not make any responses during the task. Please refer to **Tutorial 4: Logging Block Data in .PDAT File, (Page 46)** and **Tutorial 5: Logging Stimulus and Response Data in .PDAT File, (Page 51)** in this manual. This is the Run Report, which summarizes information about the experiment run that was completed.

Some EEfMRI PackageCalls will store additional detailed information about the scanning run in a tab delimited output file. By default this output file is located in the same folder as the experiment (e.g., …\ My Experiments\fMRI\Tutorials) and will have the same name as the experiment's E-Prime data file but with the extension ".PDAT" appended (e.g., assuming you entered a one for Subject Number and a one for Session Number the file created by running the experiment in this tutorial 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. For further information about the .PDAT file, see **Appendix B: .PDAT File Format, (Page 78)**.

Tutorial 2: **Creating a Menu**

If you have not completed Tutorial 1, start by opening the experiment "…\My Experiments\fMRI\Tutorials\fMRIPress1.es2". This experiment contains all of the changes made to the Press.es2 experiment in Tutorial 1. Otherwise, this tutorial assumes you are continuing from Tutorial 1.

Resave the current experiment:

Detailed instructions on how to save an experiment under a new name are found in ***Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment,*** **Task 2: Save the experiment under a new name, (Page 16)**

1) ***Select*** the **File** > **Save As...** from the application menu bar.
2) ***Type*** "**fMRIPress2.es2**" as the new name in the **File name** field.
3) ***Click*** the **Save** button.

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 be able to quickly interrupt the experiment, reset the stimulus presentation sequence, and restart the task from a particular point in the experiment, 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 will be shown again after you return to the menu, and restart the experiment.

Goal:

When you have completed this tutorial you will have created an experiment that allows the user to choose a task to run from the menu list. It 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 and set the parameters to populate the menu with information from the "RunList".
- Add the fMRIMenuEnd PackageCall to designate the end of the menu function.
- Add a RunList and edit it to contain the "MenuItem" attribute that will populate the menu text.
- Create a PressRunProc to allow for multiple runs.
- Move PackageCalls to PressRunProc to change the structure for multiple runs.
- Relocate the BlockList to PressRunProc 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.

Estimated Time: 15-20 minutes

Task 1: Add the fMRIMenuBegin PackageCall

Add a PackageCall to the SessionProc to direct EEfMRI to create a menu. Name the object "fMRIMenuBegin."

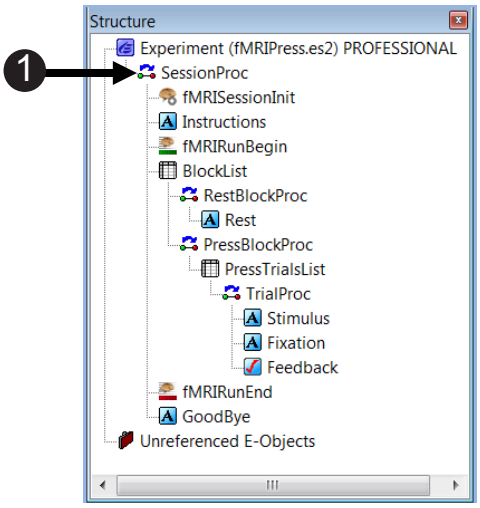
The fMRIMenuBegin and fMRIMenuEnd PackageCalls are used to create an interactive menu within the experiment. This menu system can be used by the experimenter to dynamically choose different tasks to present to the participant at runtime. The menu is created by processing a designated List object defined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

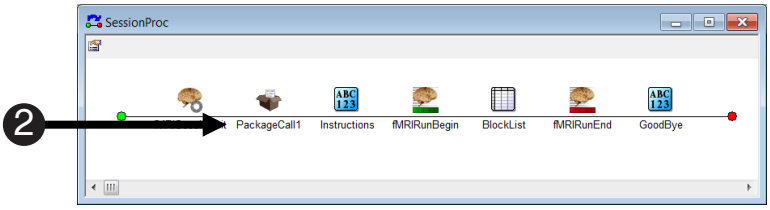
1) ***Double click*** the **SessionProc** object to open it in the workspace.

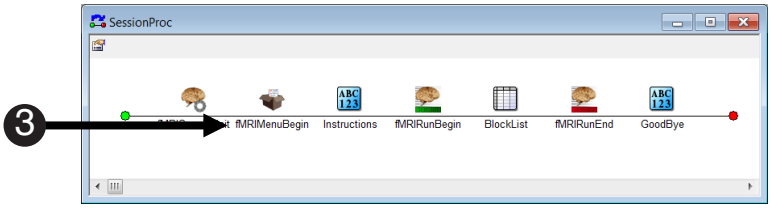
2) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it ***after*** **fMRISessionInit**. The object will be given a default name of **PackageCall1**.

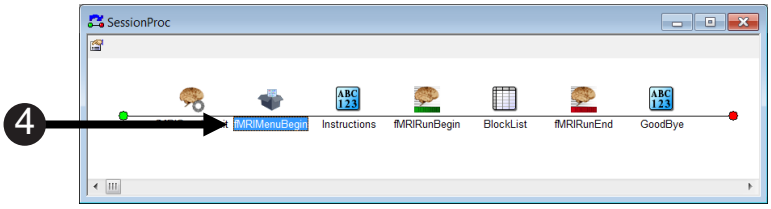
3) ***Click*** on the **PackageCall1** object to select it then ***press*** **F2** to rename the object to "**fMRIMenuBegin**". Then ***press*** **Enter** to accept the change.

4) ***Double click*** the **fMRIMenuBegin** PackageCall object located on the **SessionProc** to display its **Properties** dialog.









Task 1 (continued): Add the fMRIMenuBegin PackageCall

Configure the fMRIMenuBegin PackageCall to create a menu and set the parameters to search for the RunList object.

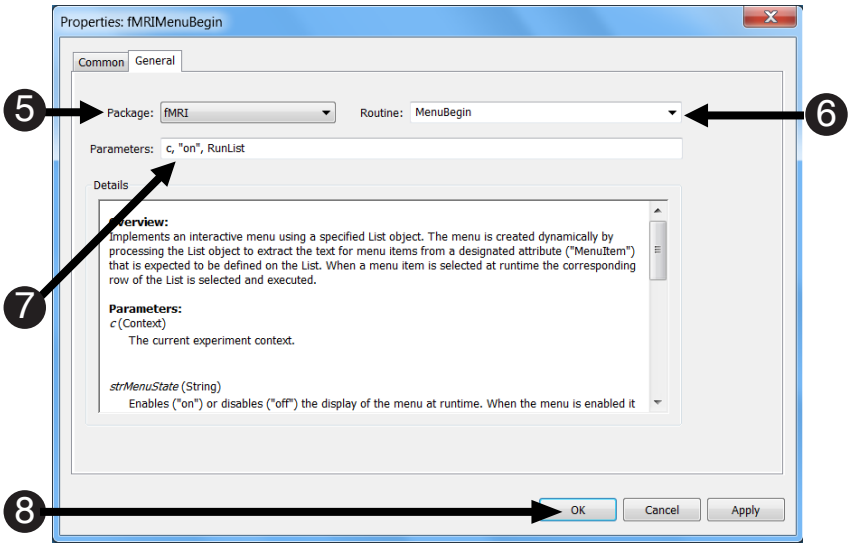
When you set Routine to fMRIMenuBegin, you will notice that the Routine's default parameters contain a reference to an object named "RunList". This is the default name of the List object that the fMRIMenuBegin PackageCall will search for to obtain the information used to populate the menu. Although the RunList object is referenced here, it will not be created until Task 3 of the tutorial.

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

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

7) ***Review*** the **Parameters:**
c, "on", RunList
The RunList object will be added to the experiment in a later step.

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



Task 2: Add the fMRIMenuEnd

Add a PackageCall to the SessionProc to complete the menu. Name the object "fMRIMenuEnd." Configure fMRIMenuEnd PackageCall to accept the default parameter.

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, before the experiment ends.

1) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it on the **SessionProc** procedure ***before*** the **GoodBye** object.

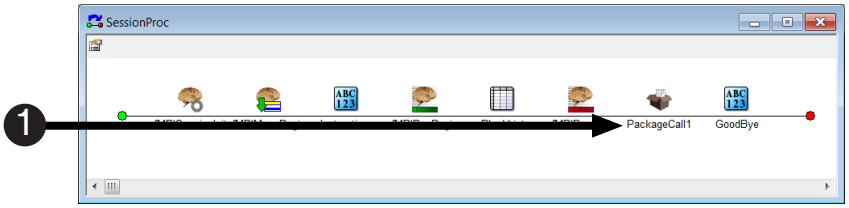
2) ***Click*** on the **PackageCall1** object to select. ***Press*** **F2** to rename the object to "**fMRIMenuEnd**". Then ***press*** **Enter** to accept the change.

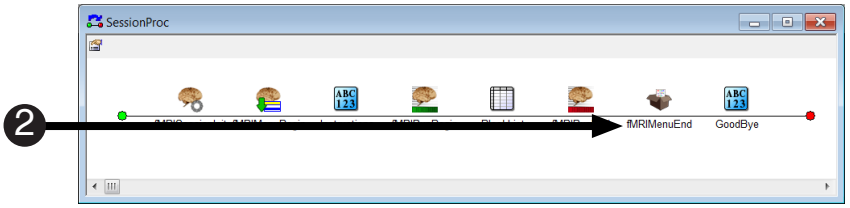
3) ***Double click*** the **fMRIMenuEnd** to open it in the workspace.

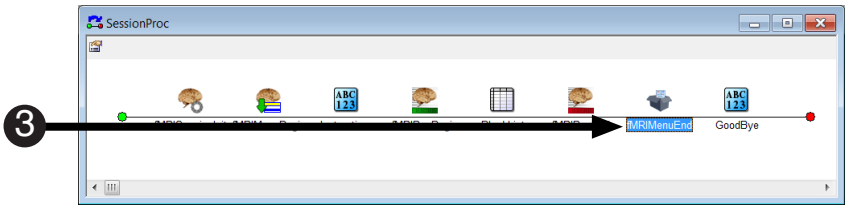
4) ***Select*** **fMRI** from the **Package** dropdown list.

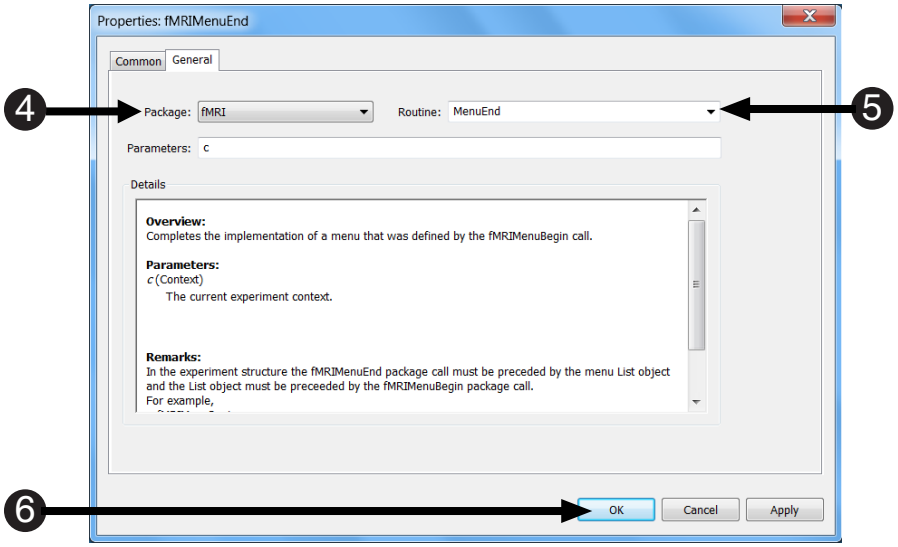
5) ***Select*** **MenuEnd** from the **Routine** dropdown list.

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









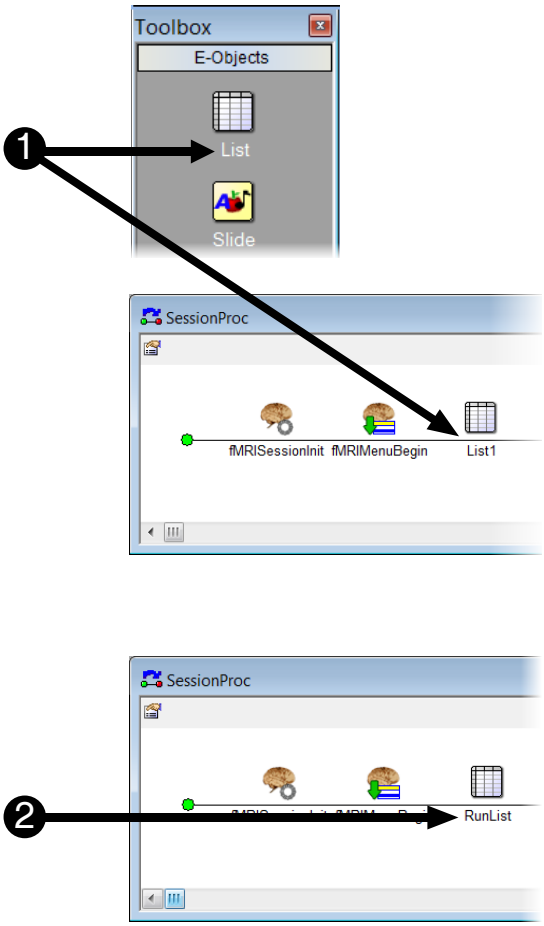
Task 3: Add the List object that will populate the menu

Add a List object to the SessionProc and rename the List object to "RunList."

The RunList object is used to define the entries that will be seen by the experimenter when the menu is displayed. The object also associates a particular Procedure with each menu entry. When the user chooses the menu item, fMRI will run 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.

1) ***Drag*** a new **List object** icon from the **Toolbox** onto the **SessionProc *after*** the **fMRIMenuBegin** object. By default the object will be given the **List1** name.

2) ***Click*** on the **List1** object to select it then ***press* F2** to rename the object as "**RunList**". Then ***press* Enter** to accept the change.



Task 4: Enable the RunList object to populate the menu

Add the "MenuItem" attribute to the RunList.

Edit the RunList object to add the "MenuItem" attribute. When the fMRIMenuBegin PackageCall creates the menu at runtime it will examine the MenuItem attribute to extract the text that will be used for each item listed on the menu.

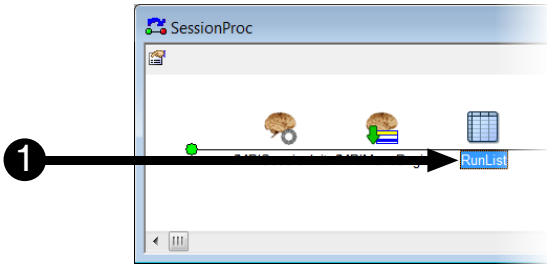
1) ***Double click*** the **RunList** object to open it in the workspace.

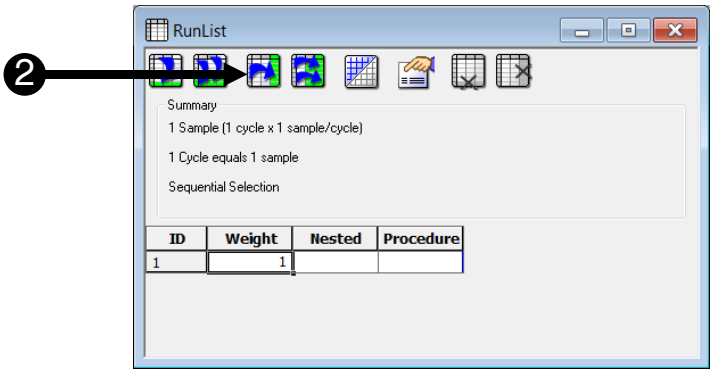
2) ***Click*** the **Add Attribute** button to open the **Add Attribute** dialog.

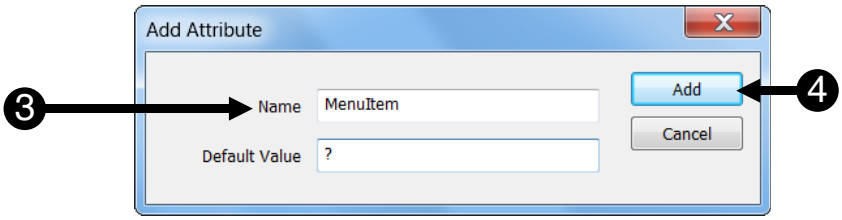
3) ***Name*** the **Attribute** "**MenuItem**".

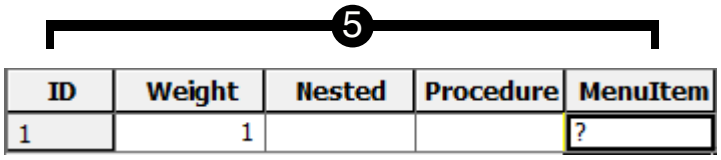
4) ***Click*** **Add.**

5) The **RunList** object should now contain the following columns.
   ID
   Weight
   Nested
   Procedure
   MenuItem









Task 4 (continued): Enable the RunList object to populate the menu

Create the PressRunProc.

Each MenuItem displayed on the menu must be associated with a Procedure object. When the experimenter selects an entry from the menu, EEfMRI will run the Procedure that is associated with the selected MenuItem. Create a new Procedure object and name it PressRunProc.

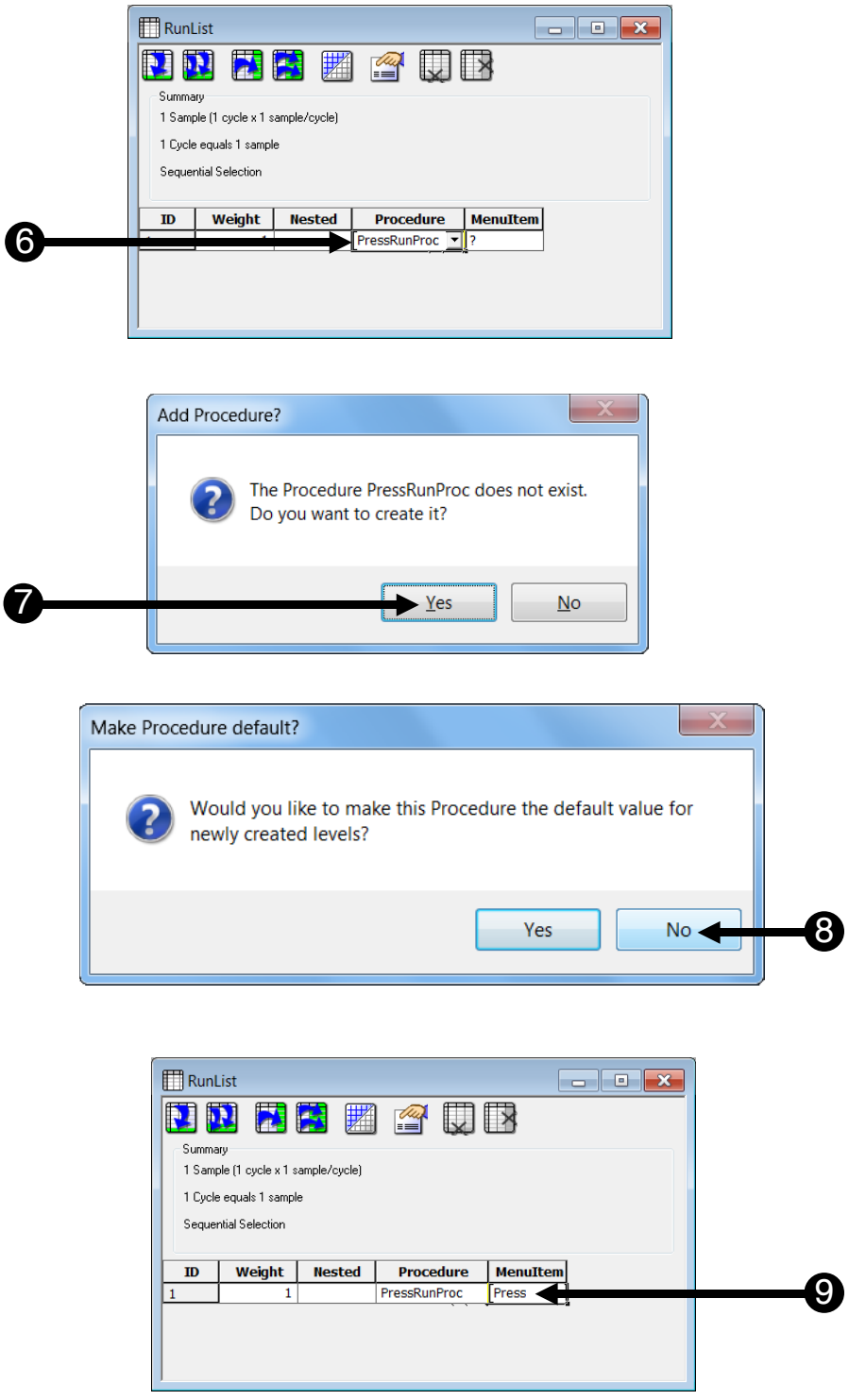
6) ***Edit*** the text cell in the **Procedure** column to read, "**PressRunProc**", and ***press*** **Enter** to accept the change.

7) ***Click*** **Yes** on the **Add Procedure** dialog to create the new **PressRunProc Procedure** object.

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

9) ***Edit*** the text cell in the **MenuItem** column to read, "**Press**".

   This text is what will be shown on screen when the menu is displayed.



Task 5: Move the Instructions object, fMRIRunBegin, and the fMRIRunEnd PackageCalls

Use the Shift key to move the Instructions, fMRIRunBegin, and fMRIRunEnd objects from the SessionProc to the new PressRunProc.

In this step, you will move the Instructions, fMRIRunBegin, and fMRIRunEnd objects from their current location on the SessionProc procedure to the new PressRunProc procedure that you created in the last step. It is important that you understand the following steps before you try to execute them. The objects need to be moved, not copied. If you drag and drop the objects without holding the Shift keys, then you will copy the objects rather than move them. Please read the next instructions carefully before trying to perform these steps.

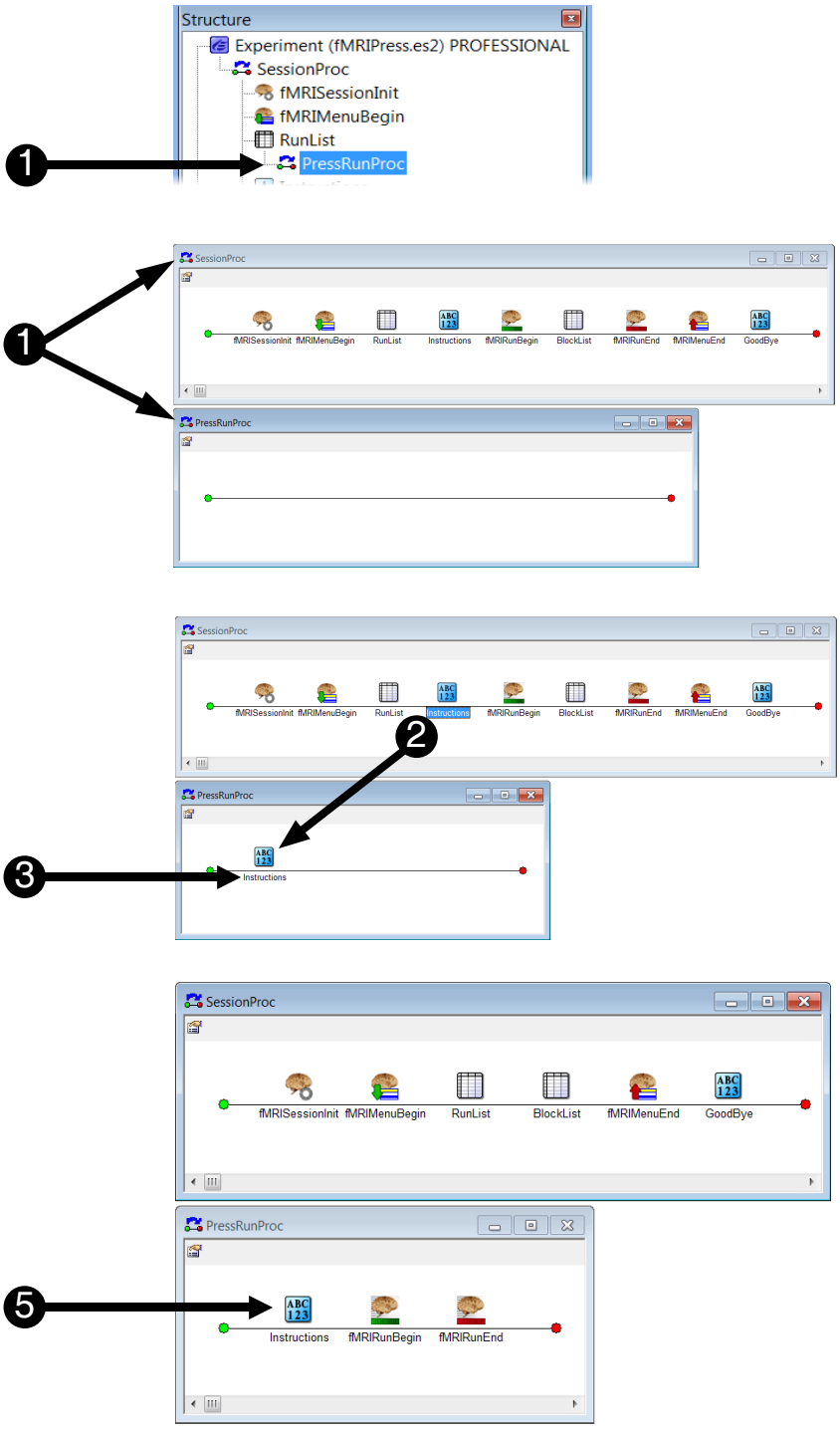
1) ***Double click*** the **PressRunProc** to open it in the workspace and ***position*** the dialog under the **SessionProc**.

2) ***Hold down*** the **Shift** key and ***click*** the **Instructions object**.
   A small rectangle will appear towards the bottom of your mouse cursor. This will allow you to move the object.

3) ***Holding*** the **Shift** key, ***drag*** and ***drop*** the **Instructions** object onto the **PressRunProc**.

4) ***Repeat*** Steps **2** and **3** for the **fMRIRunBegin** and **fMRIRunEnd** objects.

5) The **PressRunProc** should now display the **Instructions object**, **fMRIRunBegin** and the **fMRIRunEnd** PackageCalls.



Task 6: Move the BlockList to the PressRunProc

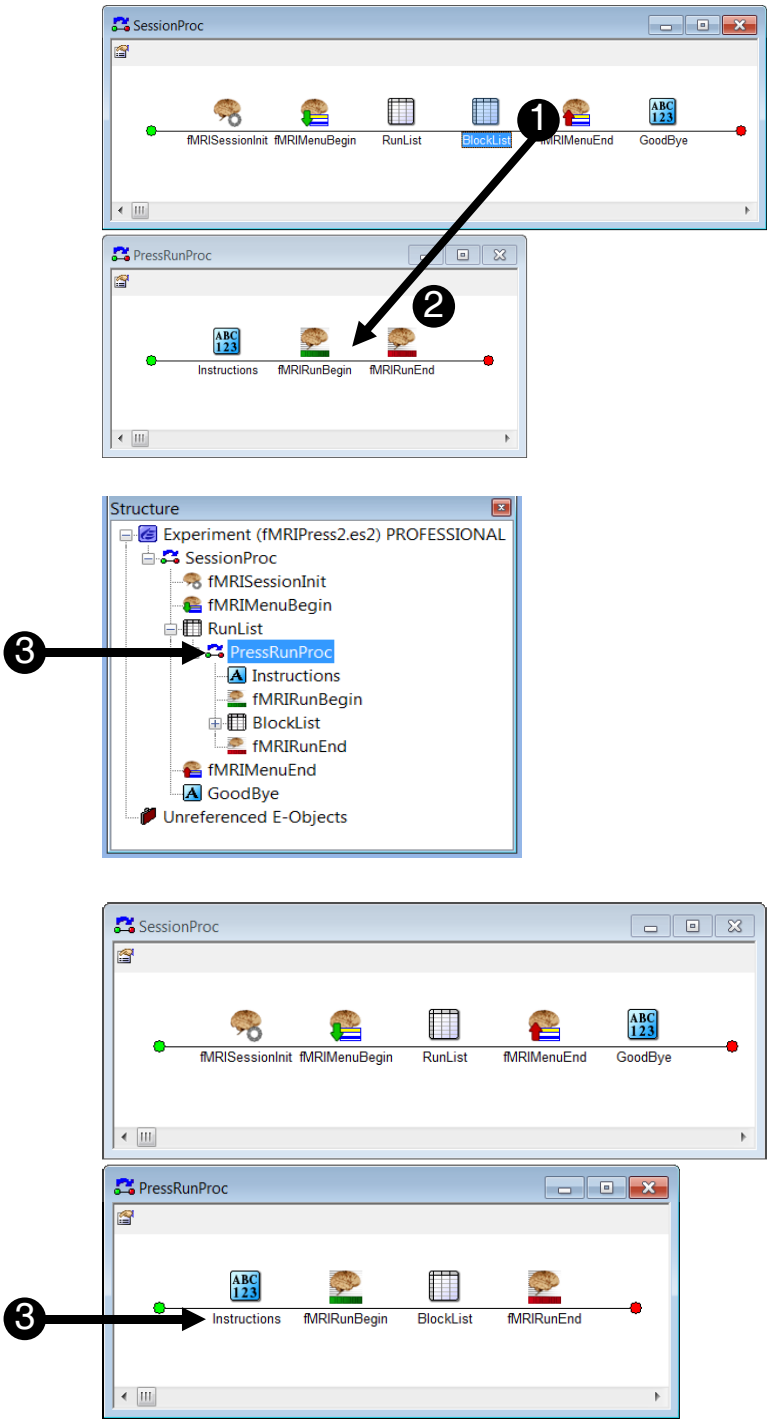
Use the Shift key to move the BlockList from the SessionProc to the PressRunProc.

In this step you will move the BlockList to the PressRunProc.

1) ***Hold down*** the **Shift** key and ***click*** the **BlockList object.**
   A small rectangle will appear towards the bottom of your mouse cursor. This will allow you to move the object.

2) ***Holding*** the **Shift** key, ***drag*** and ***drop*** the **BlockList** object onto the **PressRunProc *between*** the **fMRIRunBegin** and the **fMRIRunEnd** PackageCalls.

3) The **PressRunProc** should now contain the **Instructions, fMRIRunBegin, BlockList** and the **fMRIRunEnd** PackageCalls.
   Verify that the SessionProc and PressRunProc procedures in your experiment match those shown before continuing.



Task 7: Run the experiment

Run the experiment to verify that the appropriate prompts are presented and no runtime errors are generated.

You have now completed the basic steps necessary to create a functional menu.

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

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

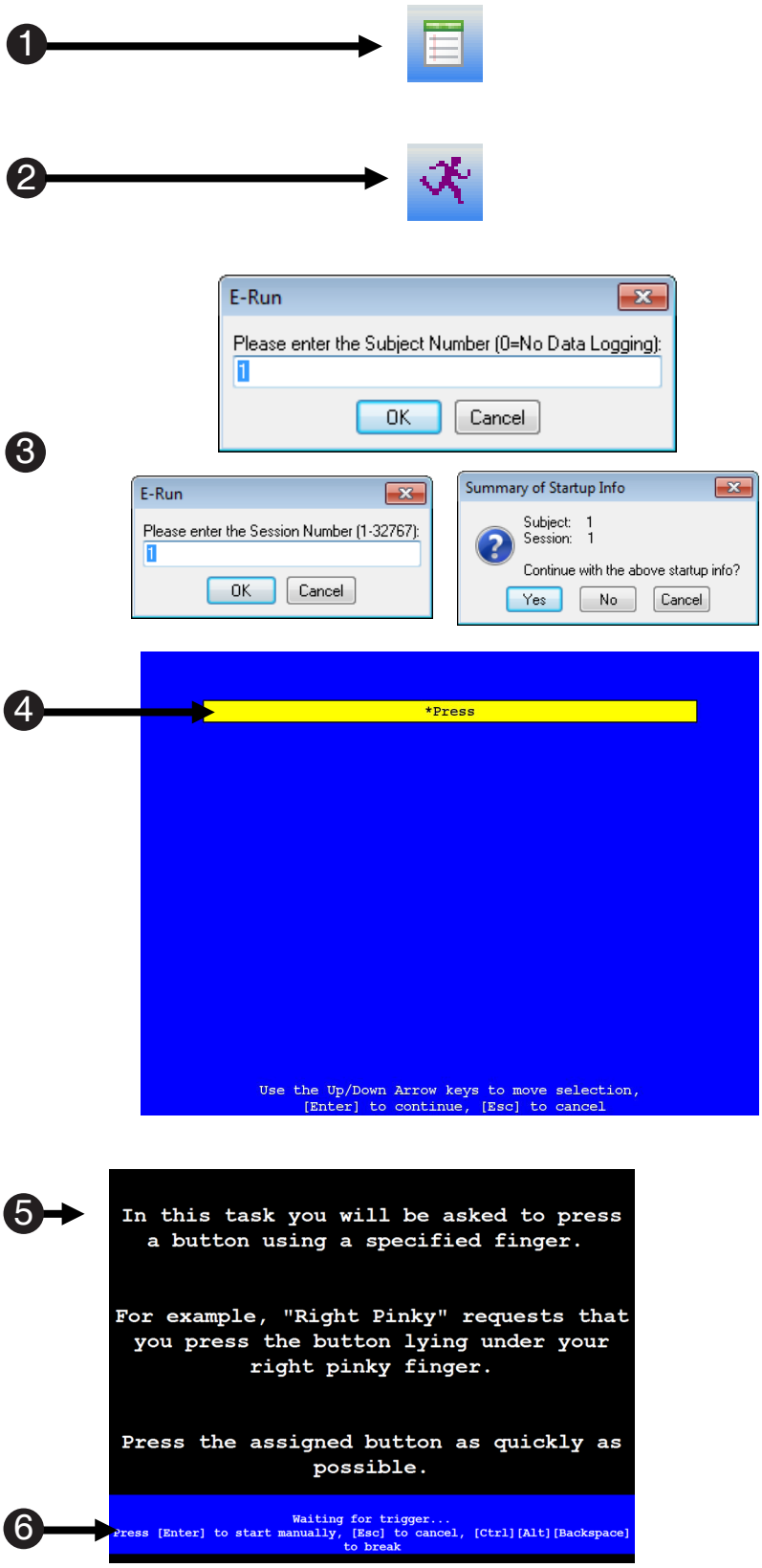
3) ***Click* OK** to accept the default values for **Subject Number**, **Session Number** and **Summary of Startup Info**.

4) ***Press* Enter** to ***select*** the highlighted menu option.

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

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

   If you test the experiment at the scanner, in conjunction with the appropriate interface hardware, the task will automatically begin when the technologist initiates the scan from the scanner console.



Task 8: Add the fMRIResetListsByName PackageCall

Add a PackageCall to the SessionProc that will direct EEfMRI to reset and reuse all of the stimulus items in a given list. Name the object "fMRIResetListsByName".

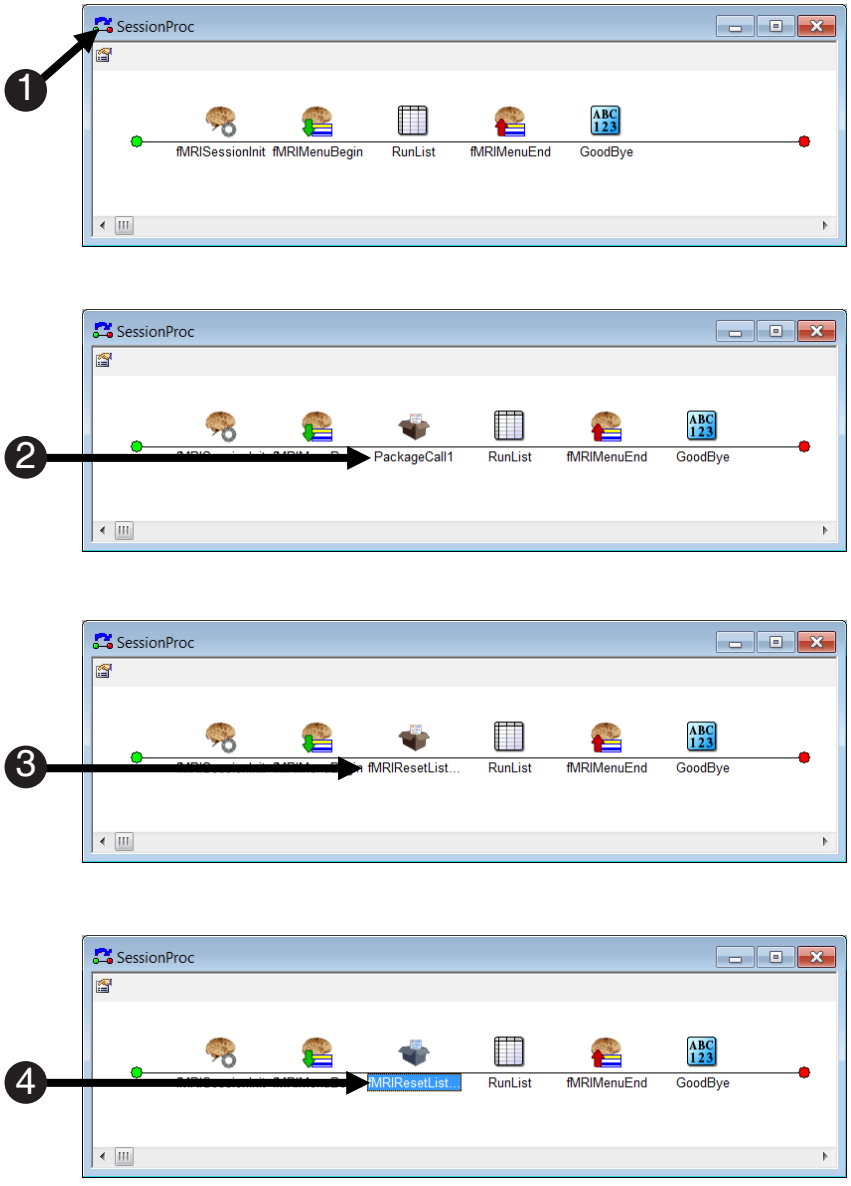
In experiments where a menu system is in use, it is likely you will want to reset the stimuli in most List objects before the next run begins. For the purposes of this manual, a run is defined by the start and subsequent stop of the scanner. Unless you take care to assure that stimulus lists are properly 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 insures that stimulus lists will be reset consistently and behave as expected. Generally it is recommended 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.

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

2) ***Drag*** a new **PackageCall** from the **Toolbox** and ***drop*** it on the SessionProc ***after*** **fMRIMenuBegin.**

3) ***Click*** on the **PackageCall1** object to select it then ***press*** **F2** to rename the object to **fMRIResetListsByName**. Then ***press*** **Enter** to accept the change.

4) ***Double click*** the **fMRIResetListsByName** PackageCall to display its **Properties** dialog.



Task 8 (continued): Add the fMRIResetListsByName PackageCall

Add a PackageCall to the SessionProc that will direct fMRI to reset and reuse all of the stimulus items in a given list. Name the PackageCall "fMRIResetListsByName".

The ResetListsByName PackageCall parameters require the names of List objects that need to be reset after the experiment has been interrupted. In this example the List object that controls the trial/stimulus presentation is the PressTrialsList.

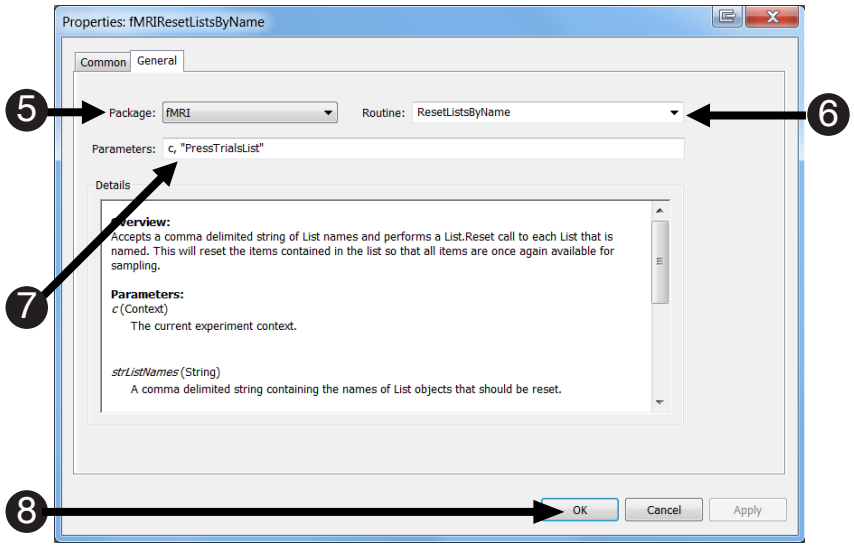
5) ***Select*** **fMRI** from the **Package** dropdown list.

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

7) ***Edit*** the **Parameters** to read:
   c, "PressTrialsList"
   In this experiment the PressTrialList Object controls the sequence and trial stimuli. The list name is passed as a parameter so that the stimuli will be reset each time the experiment is run.

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

   E-Prime Professional provides two options for interrupting an experiment. Please see **Appendix D: Interrupting an Experiment, (Page 80)** for details.



Tutorial 3: **Timing**

This tutorial will highlight timing issues surrounding an fMRI experiment and introduce some methods that E-Prime employs to deal with them. For a detailed discussion of these issues please refer to Chapter 4: Critical Timing in the *E-Prime User's Guide*.

If you have not completed Tutorial 2, start by opening the experiment "…\My Experiments\fMRI2\Tutorials\fMRIPress2.es2". This experiment contains all of the changes made to the Press.es2 experiment in Tutorials 1 and 2. Otherwise, this tutorial assumes you are continuing from Tutorial 2. Before you start, save the experiment under the name "fMRIPress3.es2" in the same folder.

Resave the current experiment:

Detailed instructions on how to save an experiment under a new name are found in ***Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment,*** **Task 2: Save the experiment under a new name, (Page 16)**

1) ***Select*** the **File** > **Save As...** from the application menu bar.
2) ***Type*** "**fMRIPress3.es2**" as the new name in the **File name** field.
3) ***Click*** the **Save** button.

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 experiment object to free some available time for the next objext in the execution sequence to prepare/ready itself for execution. This tutorial will take 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.

Goal:

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

Overview of Tasks:

- Modify Rest Object timing.
- Modify Stimulus Object timing.
- Modify Fixation Object timing.

Estimated Time: 10-15 minutes

Task 1: Modify Rest Object Timing

Open the Rest 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. If used correctly, in conjunction both techniques serve as powerful tools to correct for timing error. The observed effect of cumulative error in an fMRI experiment is typically that the scanner stops running before the experiment is finished presenting stimuli.

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

2) ***Click*** the **Properties** button to open the **Rest** object **Properties** dialog.

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

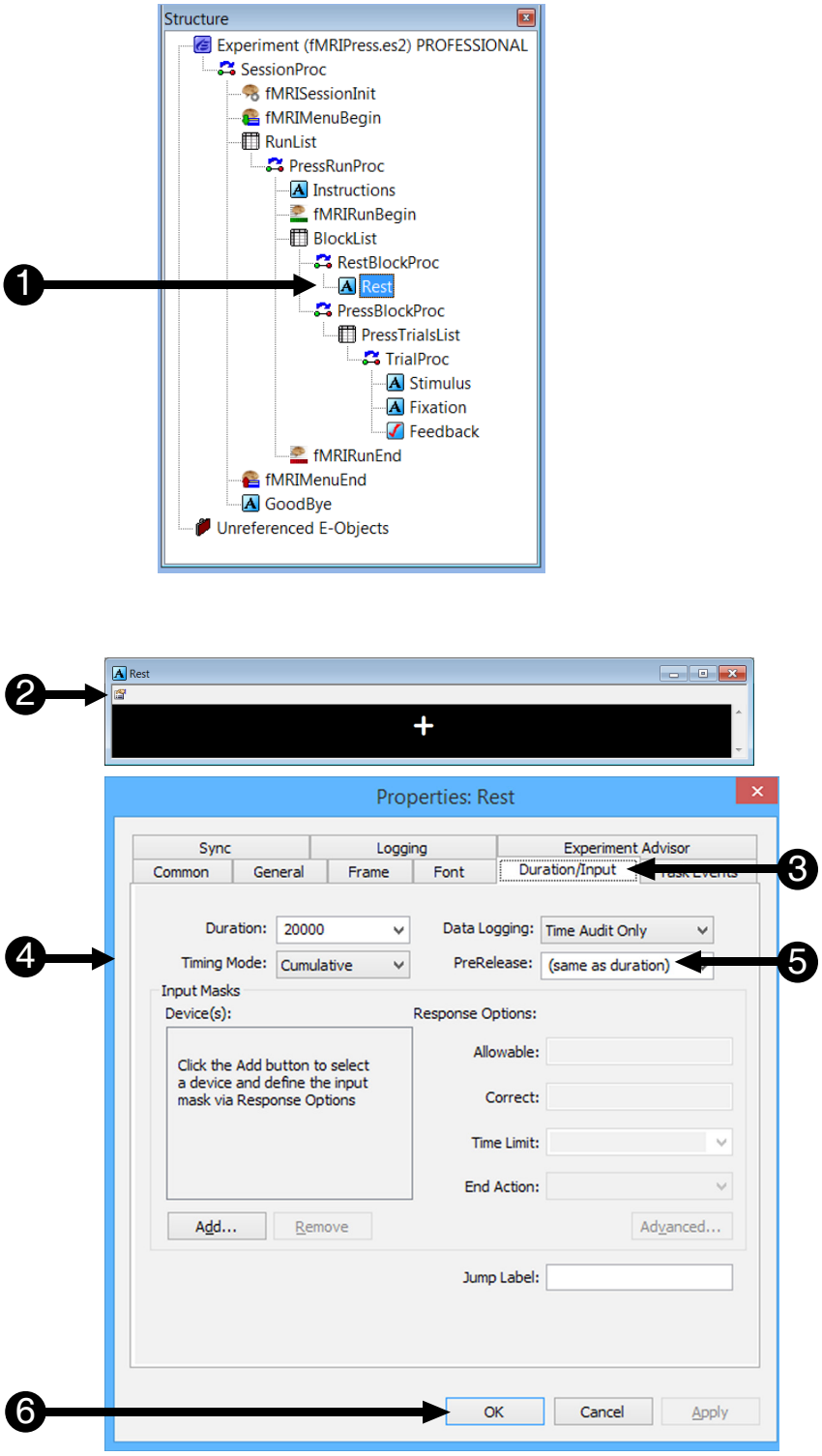
⚠

4) ***Select*** **Cumulative** from the **Timing Mode** dropdown.

   NOTE: *If there is an error in the stimulus onset time of one stimulus, this will minimize delays in stimulus presentation by absorbing the error.*

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

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



Task 2: Modify Stimulus Object Timing

Open the Stimulus Object properties and modify the Pre-Release, Timing Mode and Time Limit.

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. If used correctly, in conjunction both techniques serve as powerful tools to correct for timing error. The observed effect of cumulative error in an fMRI experiment is typically that the scanner stops running before the experiment is finished presenting stimuli.

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

2) ***Click*** the **Properties** button to open the **Stimulus** object **Properties** dialog.

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

⚠

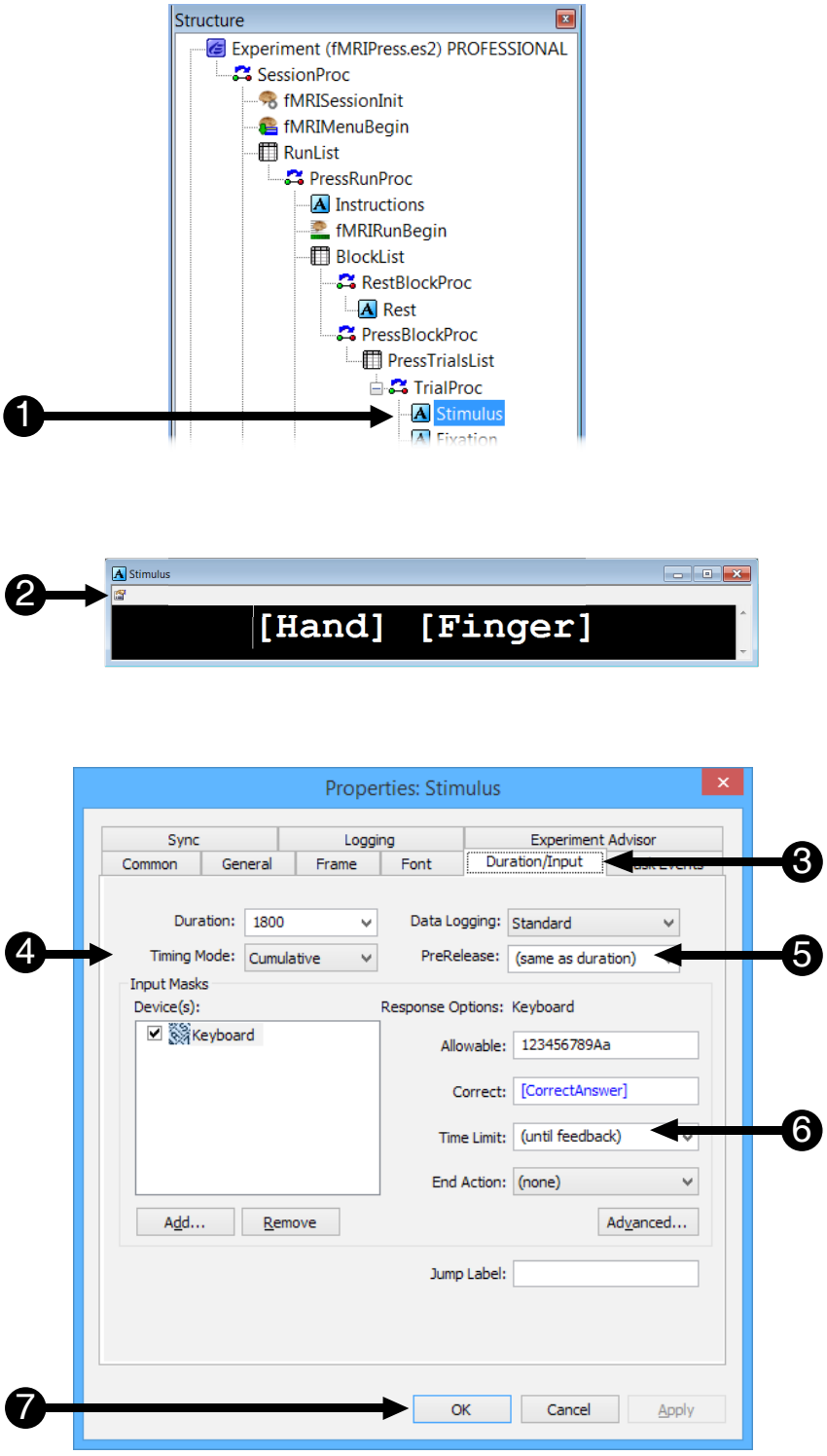
4) ***Select*** **Cumulative** from the **Timing Mode** dropdown.

   NOTE: *If there is an error in the stimulus onset time of one stimulus, this will minimize delays in stimulus presentation by absorbing the error.*

5) ***Verify*** that ***PreRelease*** is set to ***(same as duration).***

6) ***Select*** **2000 ms** from the **Time Limit** dropdown list.

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



Task 3: Modify Fixation Object Timing

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

The PreRelease option allows the computer to free resources to prepare for the next trial. Cumulative Timing Mode corrects for delays introduced into the paradigm by delays in stimulus onset by altering the presentation time of the stimulus. This minimizes the delay of subsequent events due to error. If used correctly, in conjunction both techniques serve as powerful tools to correct for timing error.

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

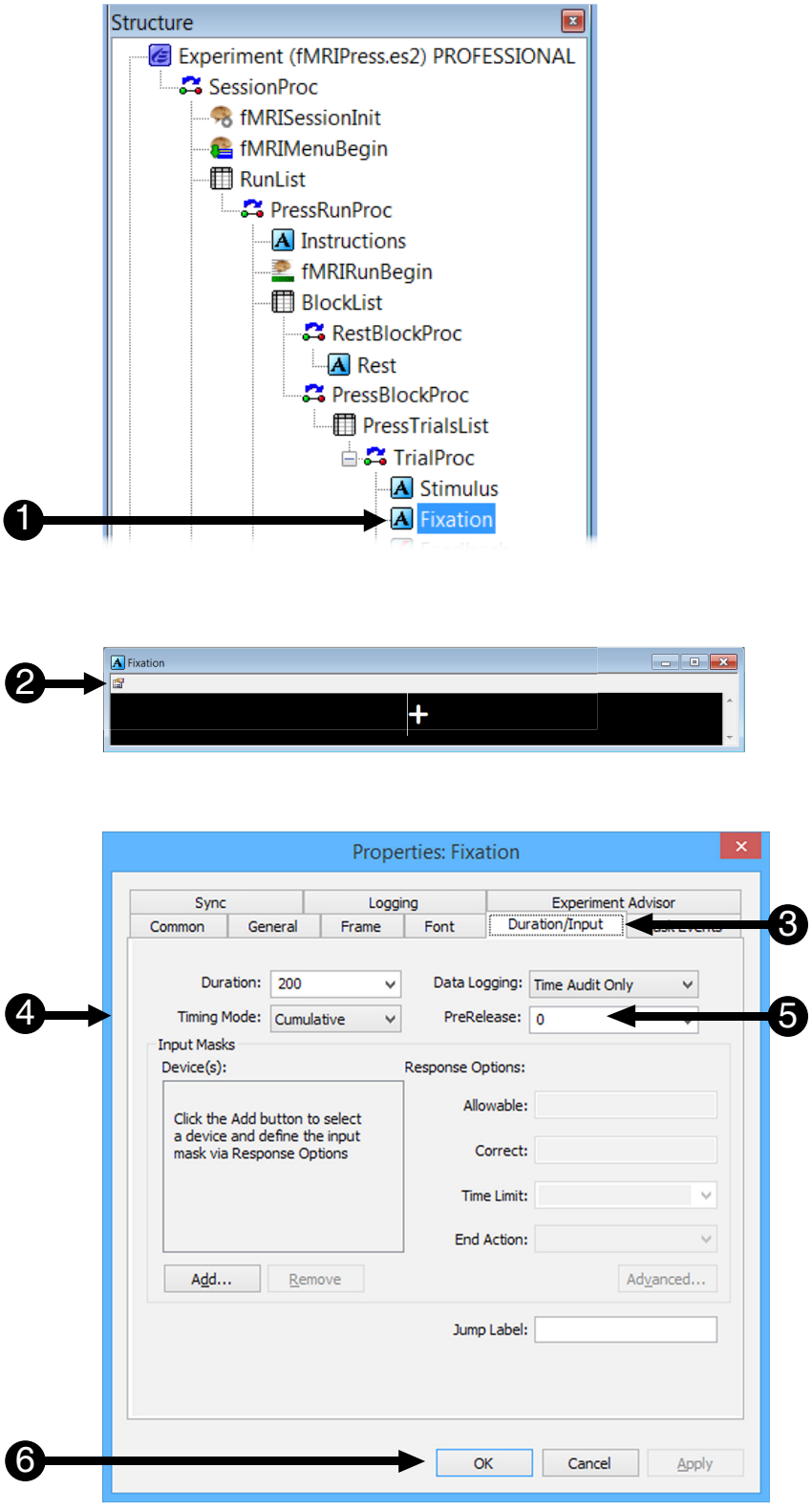
2) ***Click*** the **Properties** button open the Fixation object properties dialog.

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

4) ***Select*** **Cumulative** from the **Timing Mode** dropdown.
   ⚠ **NOTE:** *If there is an error in the stimulus onset time of one stimulus, this will minimize delays in stimulus presentation by absorbing the error.*

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

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



Task 4: Run the experiment

Run the experiment to verify that the appropriate prompts are presented and no runtime errors are generated.

You have now completed the basic steps necessary to modify the timing of this experiment.

⚠ **NOTE:** *There will be no observable difference in the timing. If you desire to see the result of the changes you made, compare the .PDAT timestamps.*

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

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

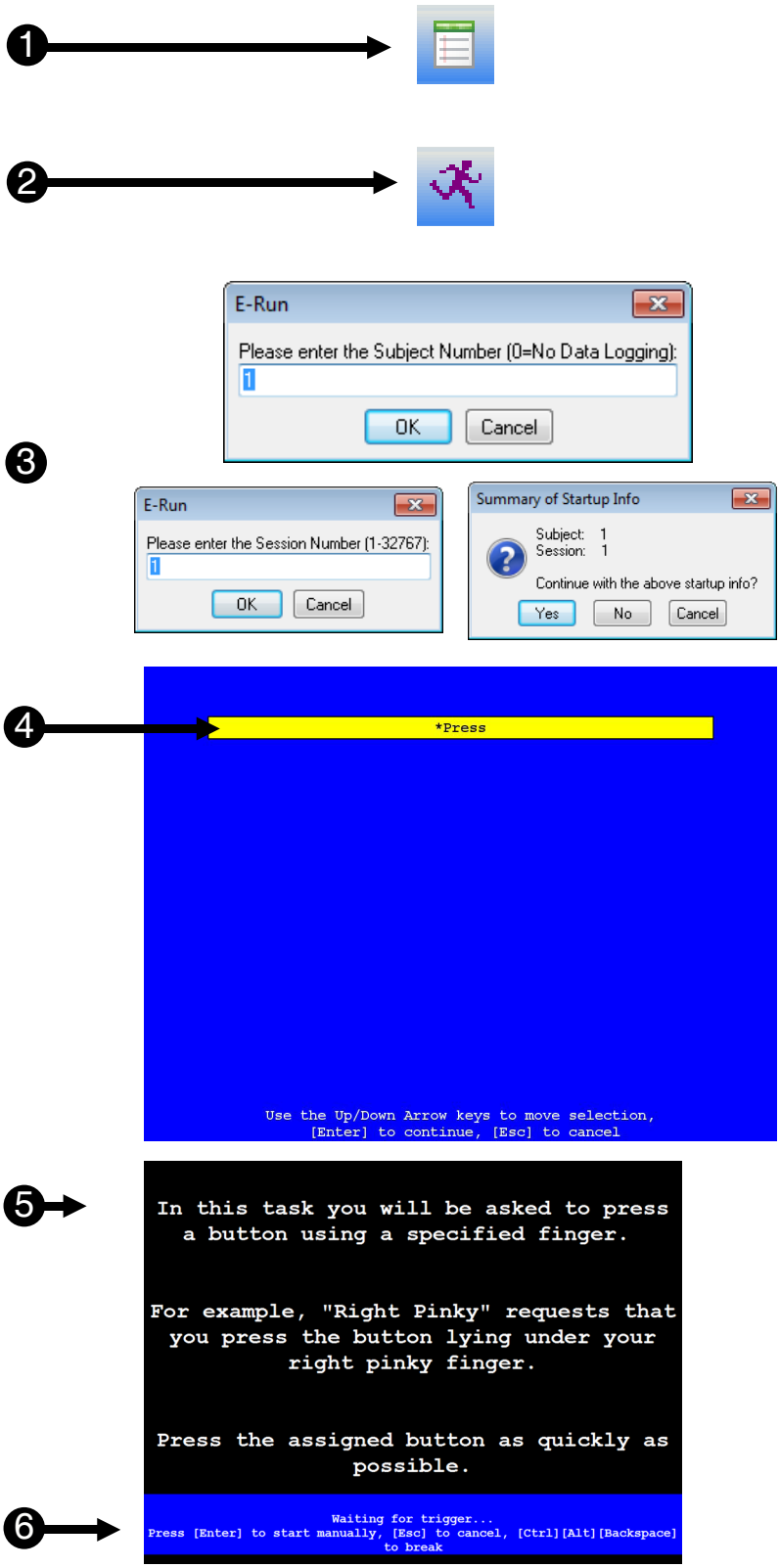
3) ***Click* OK** to accept the default values for **Subject Number**, **Session Number** and **Summary of Startup Info**.

4) ***Press* Enter** to ***select*** the highlighted menu option.

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

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

   If you test the experiment at the scanner, in conjunction with the appropriate interface hardware, the task will automatically begin when the technologist initiates the scan from the scanner console.



2.2 Analysis Issues: Tutorials 4-6

The following group of tutorials includes a brief discussion of experiment designs, how to use fMRI PackageCalls to pass timestamps to the .PDAT file to create a list of events that occurred during the experiment, and how to create a practice run.

Tutorial 4: **Logging Block Data in .PDAT File**

If you have not completed Tutorial 3, start by opening the experiment "…\My Experiments\fMRI\ Tutorials\ fMRIPress3.es2." Before you start, save the experiment under the name "fMRIPress4.es2" in the same folder.

Resave the current experiment:

Detailed instructions on how to save an experiment under a new name are found in ***Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment,*** **Task 2: Save the experiment under a new name, (Page 16)**

1) ***Select*** the **File** > **Save As...** from the application menu bar.
2) ***Type*** "**fMRIPress4.es2**" as the new name in the **File name** field.
3) ***Click*** the **Save** button.

Summary:

An fMRI experiment can contain several runs (time between each scanner start and end) consisting of different experiments made up 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.

Goal:

This tutorial will illustrate a method used to log the block condition and block onset time in an experiment. The structure of this experiment is optimal for EEfMRI package to log data.

Overview of Tasks:

- Add the fMRILogUserEventBlock PackageCall to the experiment.
- Edit the fMRILogUserEventBlock PackageCall.
- Insert fMRILogUserEventBlock PackageCall in the appropriate places in the experiment.

Estimated Time: 10-15 minutes

Task 1: Add the fMRILogUserEventBlock PackageCall

Add a PackageCall to the RestBlockProc and name the PackageCall fMRILogUserEventBlock.

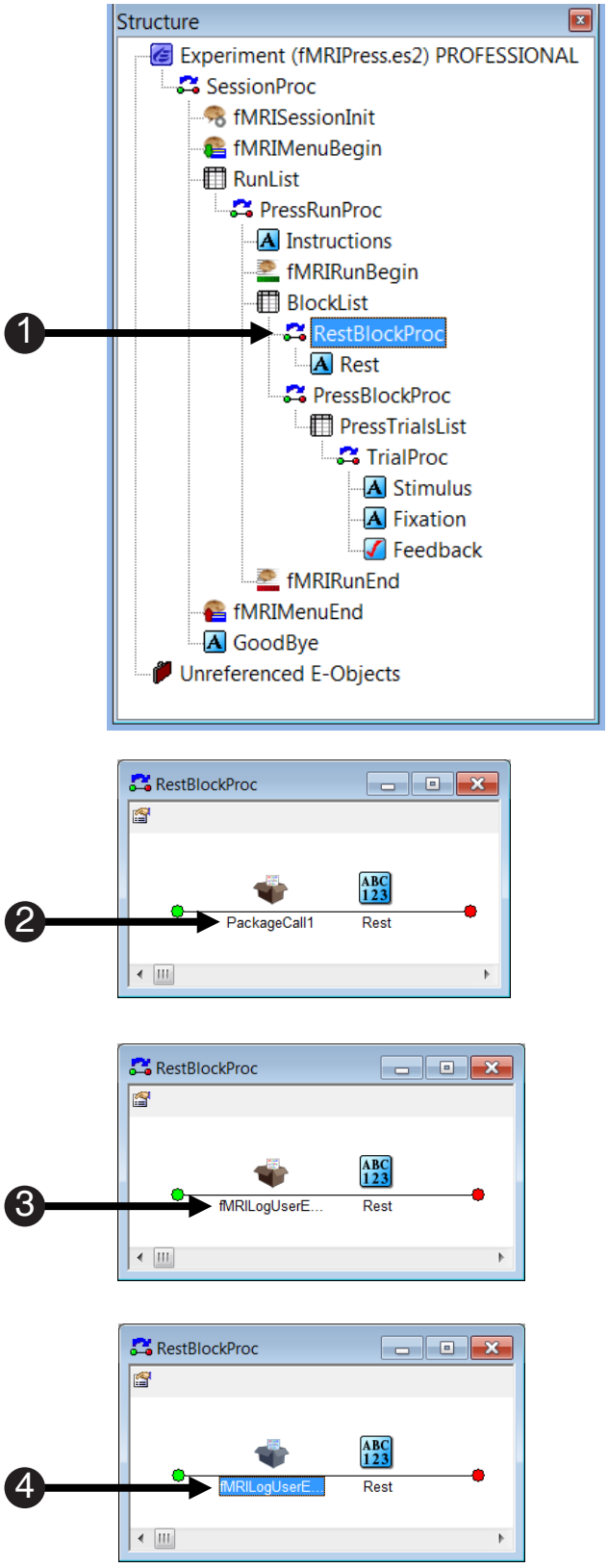
The fMRILogUserEventBlock PackageCall directs EEfMRI to log timing information and a user defined condition ID into the .PDAT file for later use during analysis.

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

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

3) ***Click*** on the **PackageCall1** object to select it then ***press*** **F2** to rename the object to **fMRILogUserEventBlock.** Then ***press*** **Enter** to accept the change.

4) ***Double click*** the **fMRILogUserEventBlock** to open the properties dialog.



Task 1 (continued): Add the fMRILogUserEventBlock PackageCall

Edit the PackageCall to direct EEfMRI to log the block condition and block timestamps in the .PDAT file. Configure the fMRILogUserEvent PackageCall to record the Block Condition.

During analysis it is necessary to know what condition is active at any given point throughout the duration of the experiment. In order to achieve this, the fMRILogUserEvent PackageCall needs to be placed at the beginning of each block procedure.

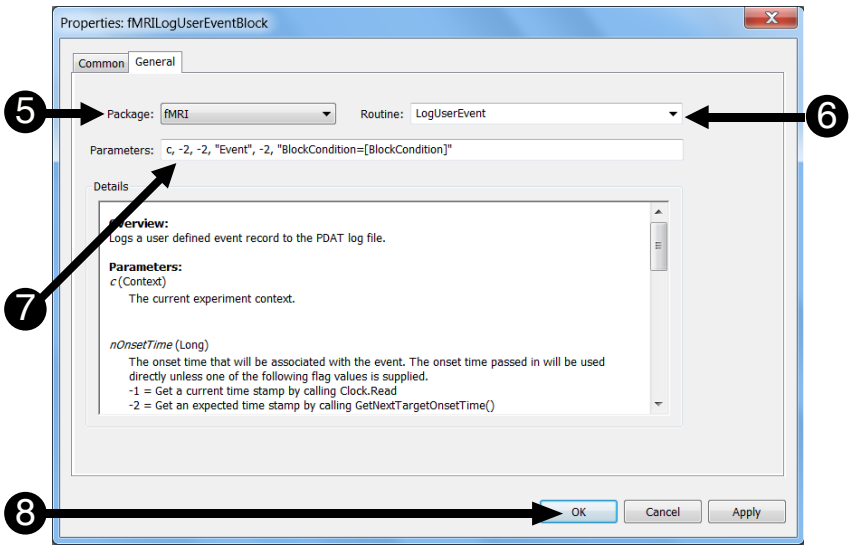
5) ***Select*** **fMRI** from the **Package** dropdown list.

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

7) ***Edit*** the **Parameters** to read:
c, -2, -2, “Event”, -2, “BlockCondition= [BlockCondition]”

⚠ **NOTE:** *The squared brackets denote an Attribute in E-Basic. Please refer to the E-Prime User’s Guide for more information.*

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



Task 2: Add the fMRILogUserEventBlock PackageCall to PressBlockProc

Add the fMRILogUserEventBlock PackageCall to the beginning of the PressBlockProc.

This PackageCall requires the user to designate how timing information about a particular event (e.g. onset vectors) is logged in the .PDAT file and what name is associated with that timing information. In this example, we will use the -2 option to record onset times and the BlockCondition attribute to rename the condition based on the procedure the experiment calls.

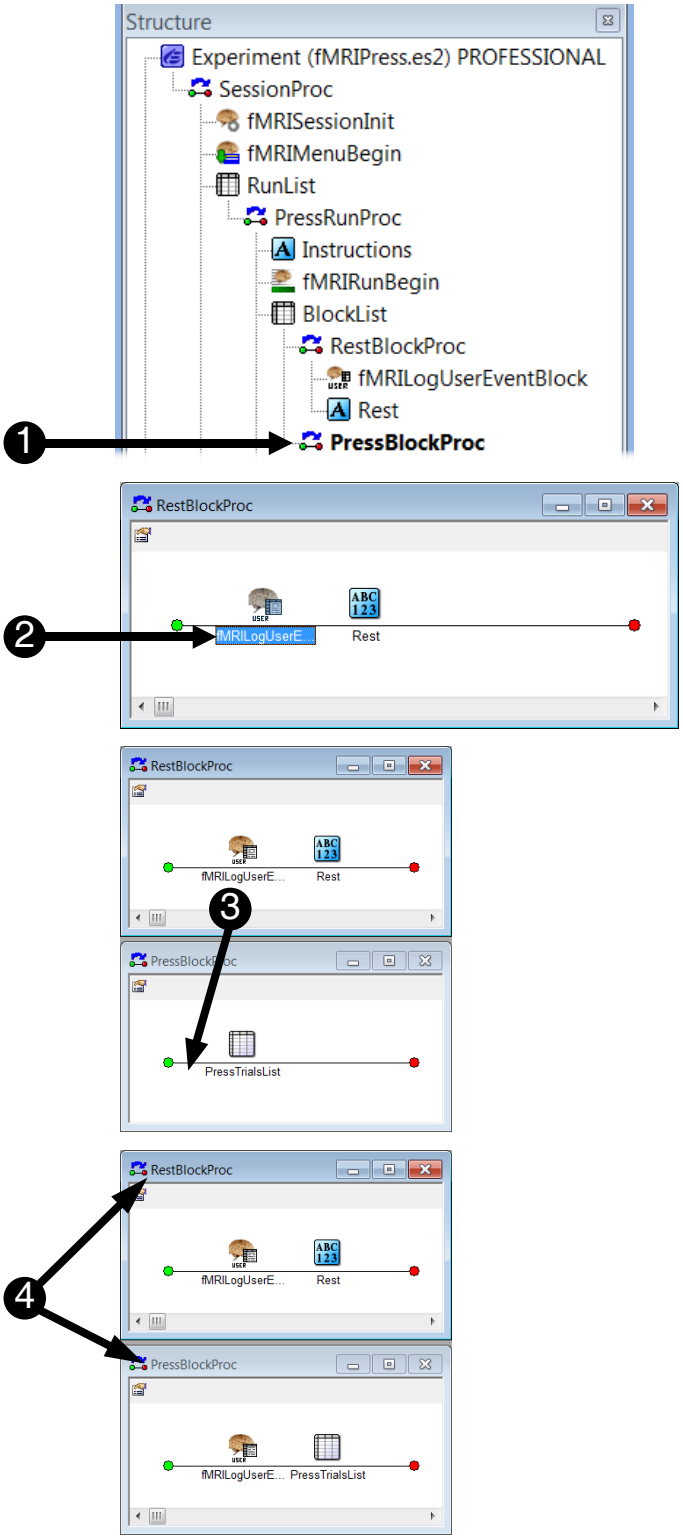
1) ***Double click*** the **PressBlockProc** to open the **Properties** dialog.

2) ***Select*** the **fMRILogUserEventBlock** located on the **RestBlockProc.**

3) ***Drag*** and ***drop*** the **fMRILogUserEventBlock PackageCall** at the beginning of the **PressBlockProc**.

4) ***Compare*** the **RestBlockProc** and the **PressBlockProc** shown with what your experiment.

   The fMRILogUserEventBlock PackageCall should be the first object listed in both procedures.



Task 3: Test

Run the experiment to verify that the appropriate prompts are presented and no runtime errors are generated.

You have now completed the basic steps necessary to log the block condition.

⚠ **NOTE:** *There will be no observable difference in the timing. If you desire to see the result of the changes you made, examine in the .PDAT file.*

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

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

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

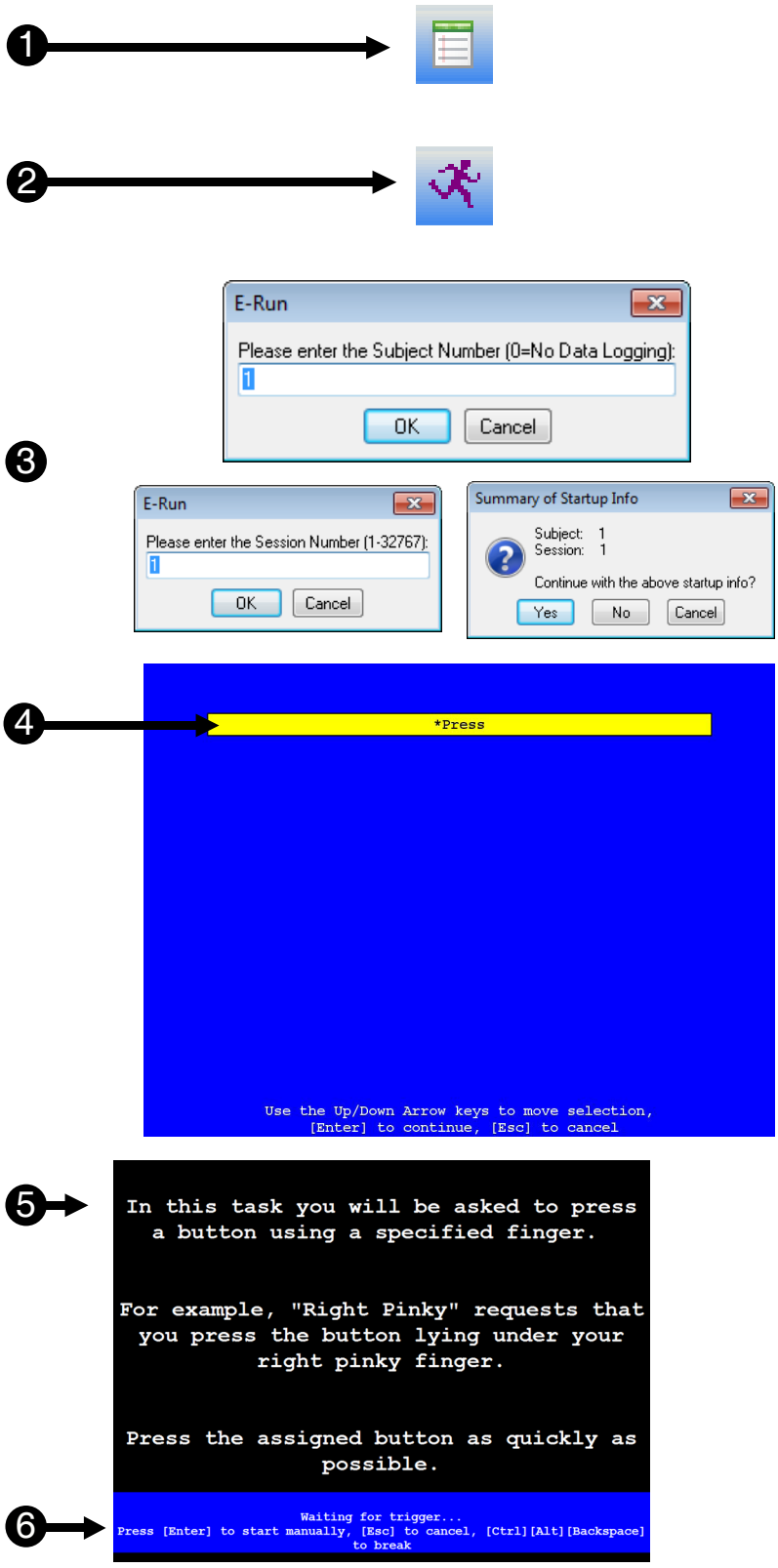
   ⚠ *If you would like to compare the new .PDAT file to the old .PDAT file, you must change the subject or run number or else the old .PDAT file will be over written.*

4) ***Press*** **Enter** select the highlighted menu option.

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

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

   If you test the experiment at the scanner, in conjunction with the appropriate interface hardware, the task will automatically begin when the technologist initiates the scan from the scanner console.



Tutorial 5: **Logging Stimulus and Response Data in .PDAT File**

If you have not completed Tutorial 4, start by opening the experiment "..\My Experiments\fMRI\Tutorials\fMRIPress4.es2." Before you start, save the experiment under the name "fMRIPress5.es2" in the same folder ("...Samples\Tutorials\fMRIPress.es2"). This ensures none of the resources in the folder becomes overwritten.

Resave the current experiment:

Detailed instructions on how to save an experiment under a new name are found in ***Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment,*** **Task 2: Save the experiment under a new name, (Page 16)**

1) ***Select*** the **File** > **Save As...** from the application menu bar.
2) ***Type*** "**fMRIPress5.es2**" as the new name in the **File name** field.
3) ***Click*** the **Save** button.

Summary:

In order to analyze fMRI 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 is able to write it to a file that is optimal for data analysis, the .PDAT file.

Goal:

This tutorial will illustrate the how to log the stimulus, participant's responses and stimulus onset times for each as well as the response for the stimulus and behavioral response data collected during an fMRI experiment in the .PDAT file.

Overview of Tasks:

- Add the fMRILogEvent PackageCall to the experiment after the event.
- Edit the fMRILogEvent PackageCall to record the stimulus condition and time.
- Add fMRILogResponseEvent PackageCall to the experiment.
- Edit the fMRILogResponseEvent Package to collect the participant's responses and time.

Estimated Time: 10-15 minutes

Task 1: Add the fMRILogEventStimulus

Add a PackageCall to the TrialProc that will direct EEfMRI to log the specified object in the .PDAT file, and name the PackageCall fMRILogEventStimulus.

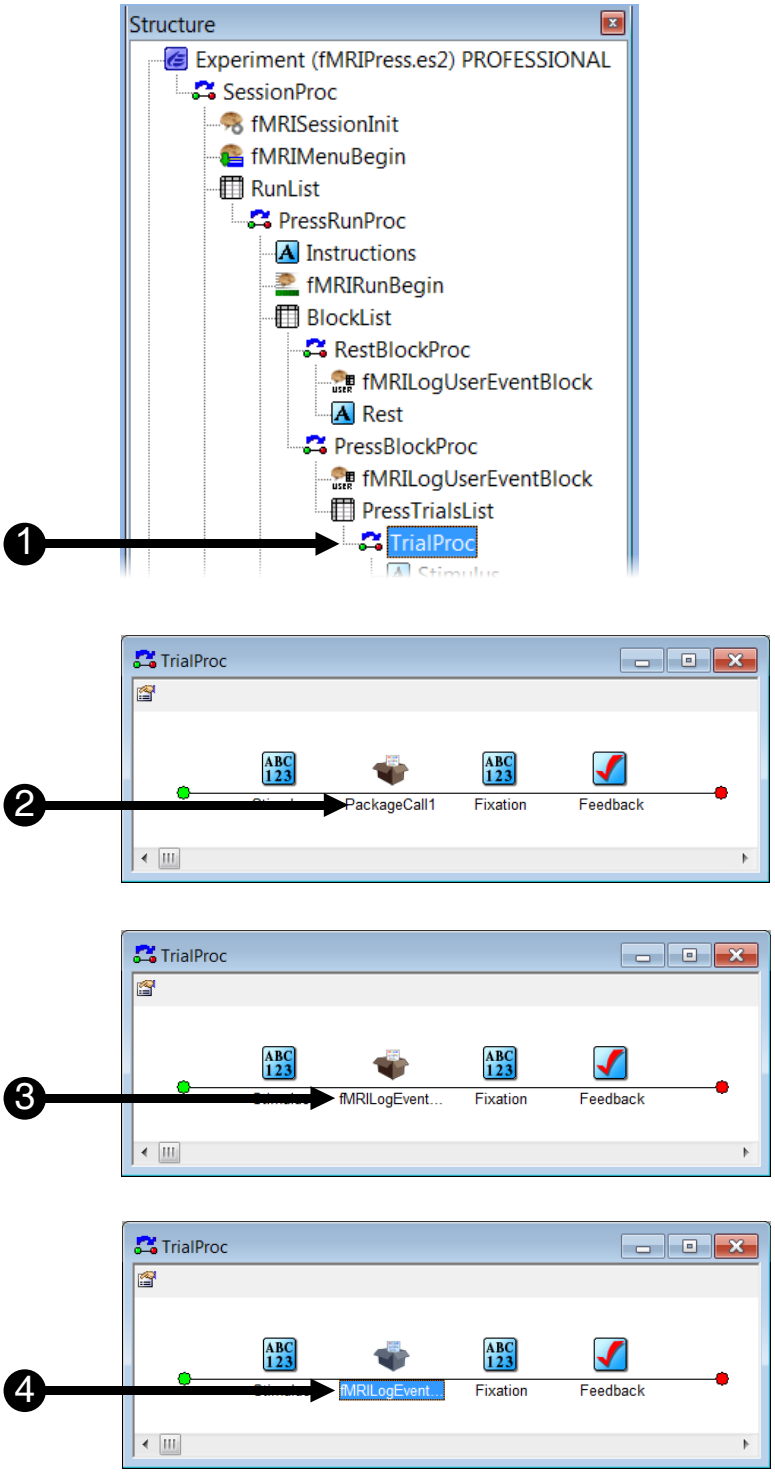
It is crucial to log the onset time and condition of each stimulus throughout the duration of an experiment. The fMRILogEvent directs EEfMRI to record this information in the .PDAT for use in later analysis.

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

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

3) ***Click*** on the **PackageCall1** object to select it then ***press*** **F2** to ***rename*** the object to **fMRILogEventStimulus.** Then ***press*** **Enter** to accept the change.

4) ***Double click*** the **fMRILogEventStimulus** to open the properties dialog.



Task 1 (continued): Add the fMRILogEventStimulus

Edit PackageCall to direct EEfMRI to log event object calls in the .PDAT file.

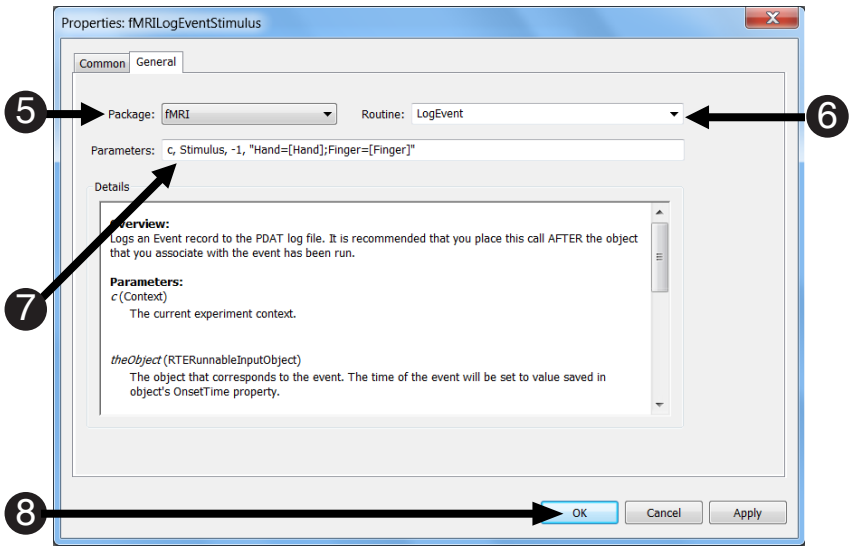
The fMRILogEventStimulus PackageCall requires the user to identify the information to be recorded in the .PDAT file. In this instance we want to document the Stimulus object. In order to accurately record what stimulus was shown we will use the hand and finger attributes from the PressTrialList and we will use the -1 option to collect the timing information.

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

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

7) ***Edit* the Parameters** to read:
c, Stimulus, -1,
"Hand=[Hand];Finger=[Finger]"

8) ***Click* OK.**



Task 2: Add fMRILogResponseEvent PackageCall and edit properties

Add PackageCall to TrialProc and name it fMRILogResponseEvent.

This PackageCall enables EEfMRI to output the participant's response and the time it occurred (i.e. its onset vector) into the .PDAT file. It is important to place the call after the response period has been collected. Like the fMRILogResponseStimulus PackageCall the fMRILogResponseEvent PackageCall is also associated with the object displaying the trials/stimuli, in this case the Stimulus object. However, the fMRILogResponseEvent is specifically designed to collect responses, label them, timestamp and record the reaction times. We will use the same parameters as the fMRILogResponseStimulus. It is the placement of the PackageCall that allows it to collect the response.

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

2) ***Click*** on the **PackageCall1** object to select it then ***press*** **F2** to rename the object to **fMRILogResponseEvent.** Then ***press*** **Enter** to accept the change.

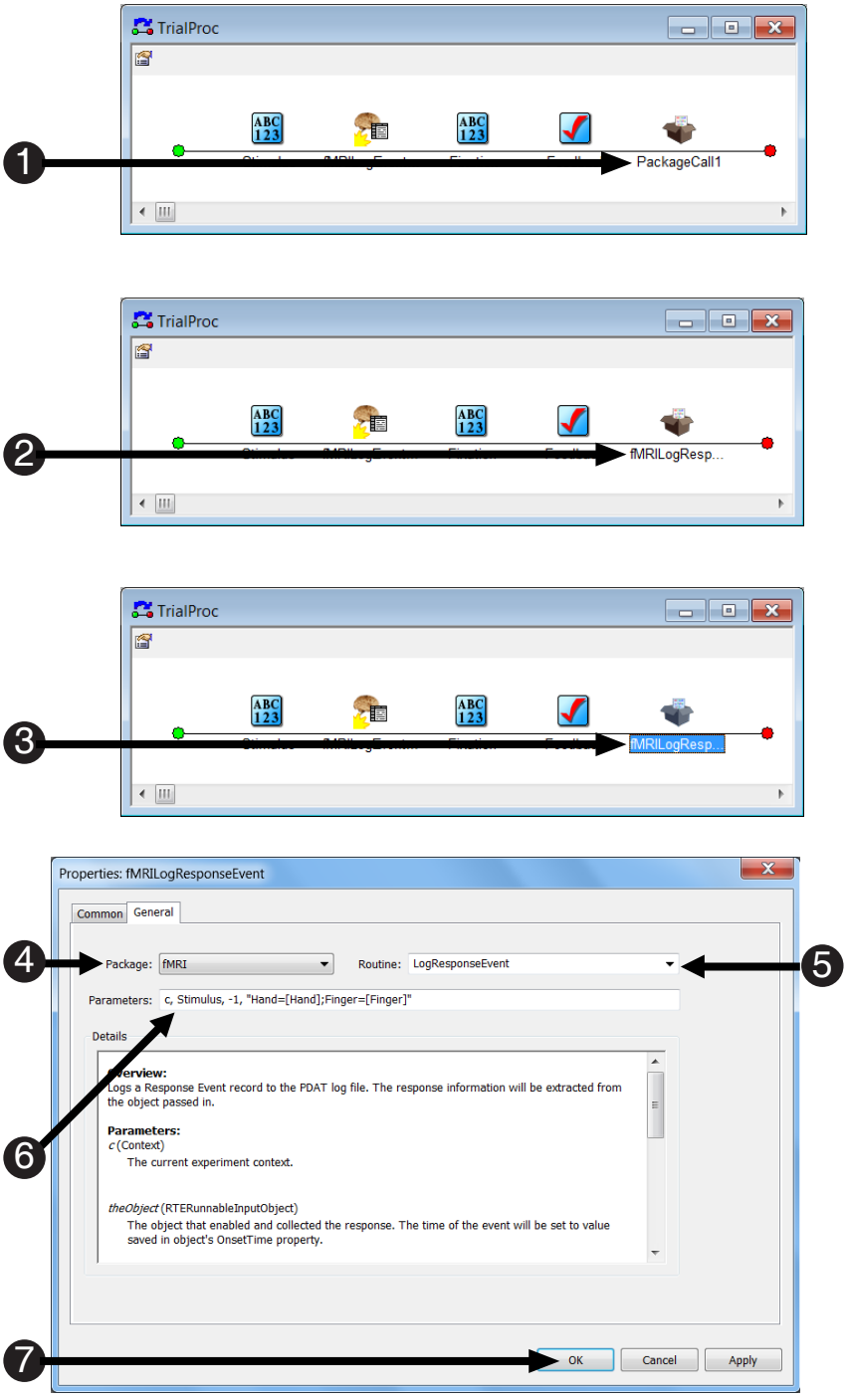
3) ***Double click*** the **fMRILogResponseEvent** to open the **Properties** dialog.

4) ***Select*** **fMRI** from the **Package** dropdown list.

5) ***Select*** **LogResponseEvent**.

6) ***Edit*** the **Parameters** to read:
c, Stimulus, -1,
"Hand=[Hand];
Finger=[Finger]"

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



Task 3: Test

Run the experiment to verify that the appropriate prompts are presented and no runtime errors are generated.

You have now completed the basic steps necessary to log timing and experiment condition information to the .PDAT file. After you run the experiment examine the .PDAT file.

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

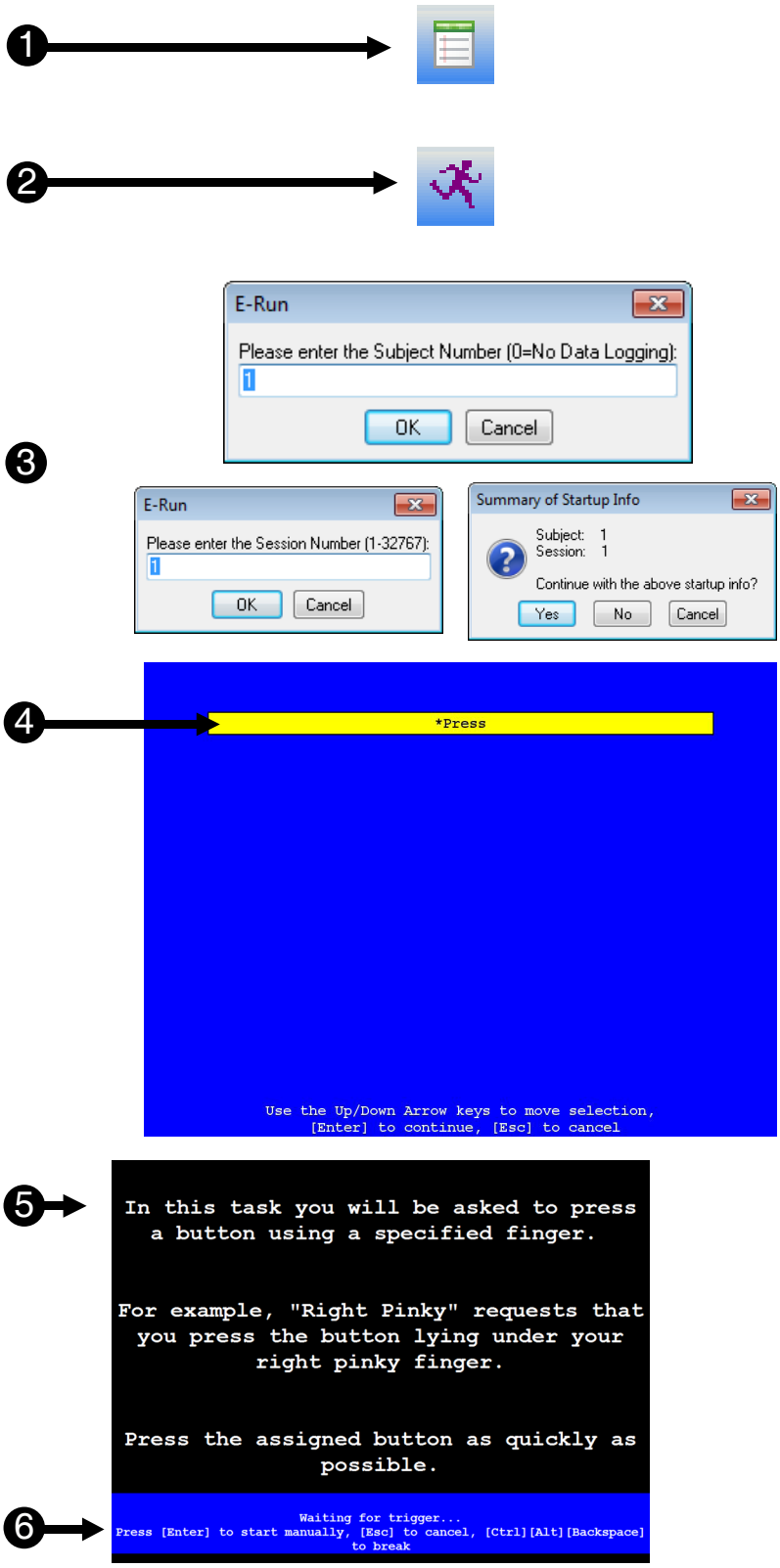
2) ***Click*** the run icon or ***press* F7** to run the paradigm.

3) ***Click* OK** to accept the **default values** for **Subject Number**, **Session Number** and **Summary of Startup Info**.
   ⚠ *If you would like to compare the new .PDAT file to the old .PDAT file, you must change the subject or run number or else the old .PDAT file will be over written.*

4) ***Press* Enter** select the highlighted menu option.

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

6) ***Press* Enter** to manually simulate the scanner trigger pulse and begin the task.
   If you test the experiment at the scanner, in conjunction with the appropriate interface hardware, the task will automatically begin when the technologist initiates the scan from the scanner console.



Tutorial 6: **Creating a Practice Run**

If you have not completed Tutorial 5, start by opening the experiment "...\My Experiments\fMRI\Tutorials\fMRIPress5.es2."

Resave the current experiment:

Detailed instructions on how to save an experiment under a new name are found in ***Tutorial 1: Adding EEfMRI Support to an E-Prime Experiment,*** **Task 2: Save the experiment under a new name, (Page 16)**

1) ***Select*** the **File** > **Save As...** from the application menu bar.
2) ***Type*** "**fMRIPress6.es2**" as the new name in the **File name** field.
3) ***Click*** the **Save** button.

Summary:

The ultimate goal of running an experiment is to collect usable 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 adequately 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.

Goal:

This tutorial will illustrate how to create a practice run and include it as a menu item.

Overview of Tasks:

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

Estimated Time: 10-15 minutes

Task 1: Edit the RunListProperties to generate a practice choice in menu

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

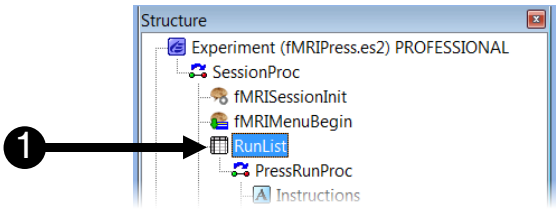
Add an Attribute to the RunList and name it Practice. The difference between a practice and experiment run is the presence or absence of feedback presentation to the participant. To enable an experiment with the ability to halt or display feedback requires several steps. The first step is to provide a way to designate when the experiment is required to run a practice or an experiment run. We will accomplish this by adding an Attribute and naming it Practice.

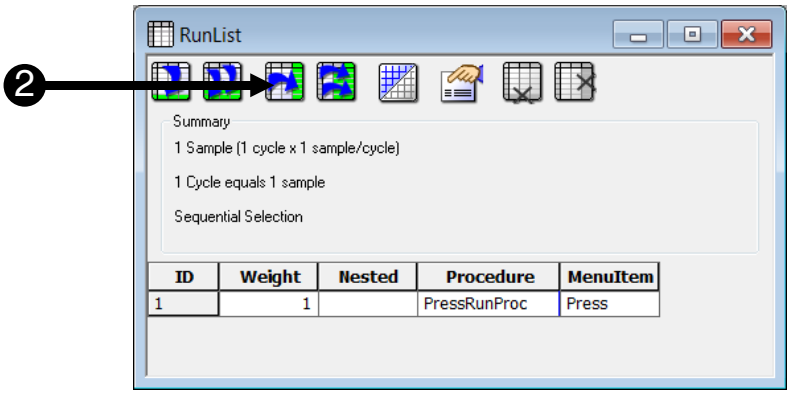
1) ***Double click*** the **RunList** icon. This will open the **RunList** in the workspace.

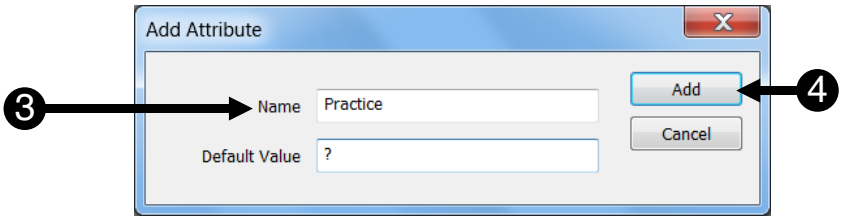
2) ***Click*** the **Add Attribute** button located at the top of the **RunList** dialog, third from the left.

3) ***Edit*** the **Name** text box to read, "**Practice**".

4) ***Click*** **Add**.







Task 1 (continued): Edit the RunListProperties to general practice choice in the menu

Add a Level to the RunList and edit it to display a practice listing in the menu.

Recall from **Tutorial 2** that the MenuItem Attribute controls the text items that populate the user menu. You will need to provide the user a way to distinguish a practice run from an experiment run. In order to this, edit the MenuItem Attribute to read “Press-Practice.” You will also need to designate a procedure for the experiment to execute during the practice run; choose PressRunProc.

5) ***Click*** the **Add Level** button located at the top of the RunList dialog, first button from the left.

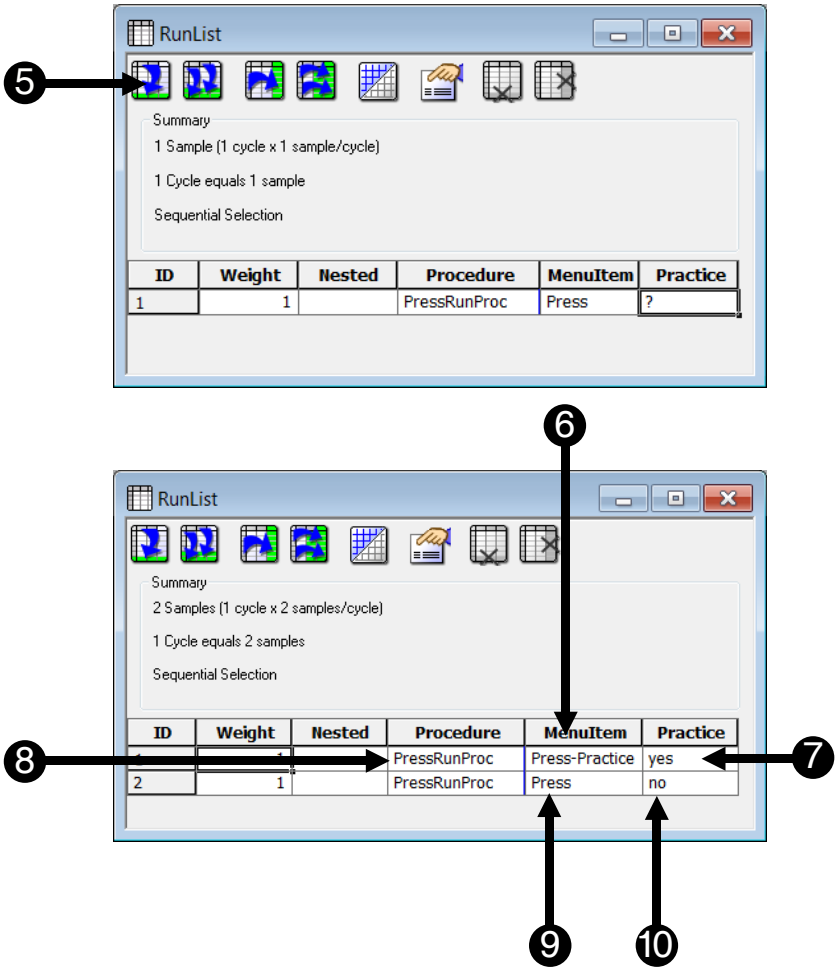
6) ***Edit*** the row one text cell**, MenuItem** column to read, *“***Press-Practice***”.*
   ⚠**NOTE:** *All text in the menu item column will be displayed on screen in the menu to the user and experiment participant.*

7) ***Edit*** row one, the practice listing, **Practice** column to read, “**yes**”.

8) ***Edit*** the **Procedure** column, row two to read, “**PressRunProc**”. This is the non-practice listing.

9) ***Confirm*** row 2, the non-practice listing, reads **Press** in the **MenuItem** column.

10) ***Confirm*** row 2, the non-practice listing, **Practice** column, reads “**no**”.



Task 2: Add and Edit an InLine Object

Add an InLine object, rename it CheckSkipFeedback, and add the code to tell E-Prime to ignore the feedback when the Press-Practice has not been selected.

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 (We will add the label in Task 3).

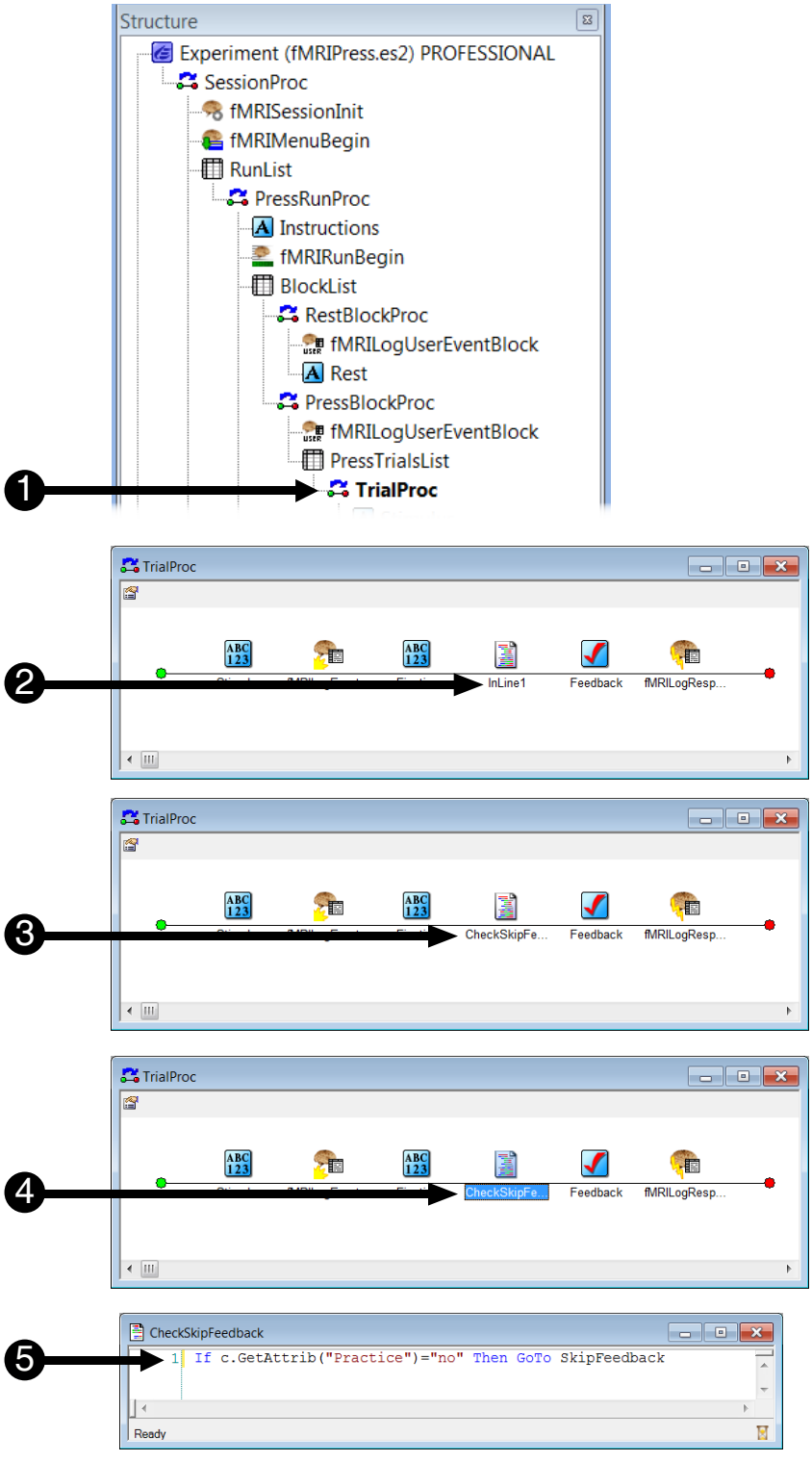
1) ***Double click*** the **TrialProc** to open its properties dialog.

2) ***Drag*** a new **InLine** object from the **Toolbox** and drop it ***after*** the **Fixation** object. The **InLine** object will be given a default name of **InLine1**.

3) ***Click*** the **InLine1** object to select it, then ***press*** **F2** to rename the object to **CheckSkipFeedback.** ***Press*** **Enter** to accept the change.

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

5) ***Edit*** the **CheckSkipFeedback InLine** to read:
   If c.GetAttrib("Practice")="no"
   Then GoTo SkipFeedback



Task 3: Add and Edit a SkipFeedback Label to TrialProc

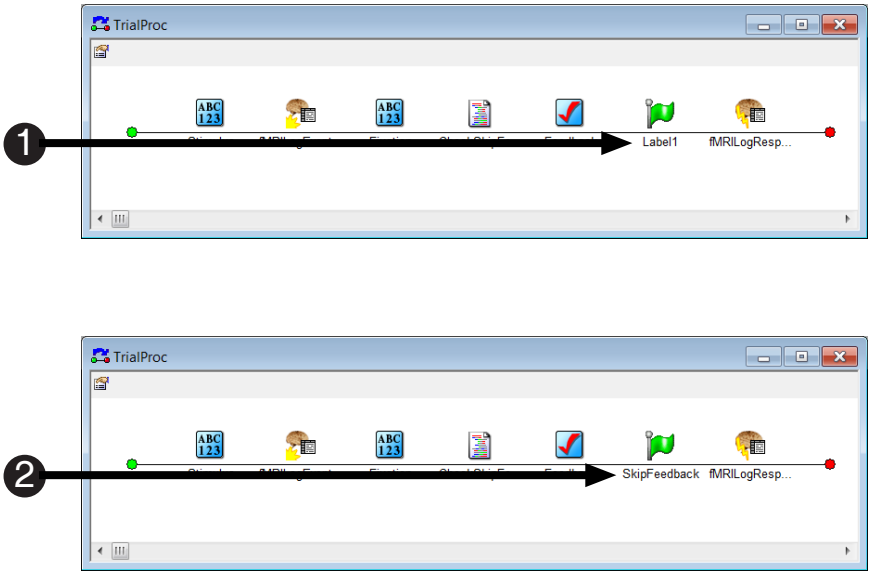
Add a label to TrialProc and name it SkipFeedback.

Designate the proper place in the experiment to resume execution once the Feedback object is skipped.

1) ***Drag*** the **Label** object from the E-Prime Toolbox and ***drop*** it on the **TrialProc *after*** the **Feedback** object.

2) ***Click*** on the **Label1** then ***press*** **F2** to ***rename*** the object. ***Rename*** the **Label1** to **SkipFeedback.**

3) ***Enter*** to accept changes.



Task 4: Run the experiment

Run the experiment to verify that the appropriate prompts are presented and no runtime errors are generated.

You have now completed the basic steps necessary to create practice run. Run the experiment in practice mode to view the trial by trial feedback and then select the Press option to compare.

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

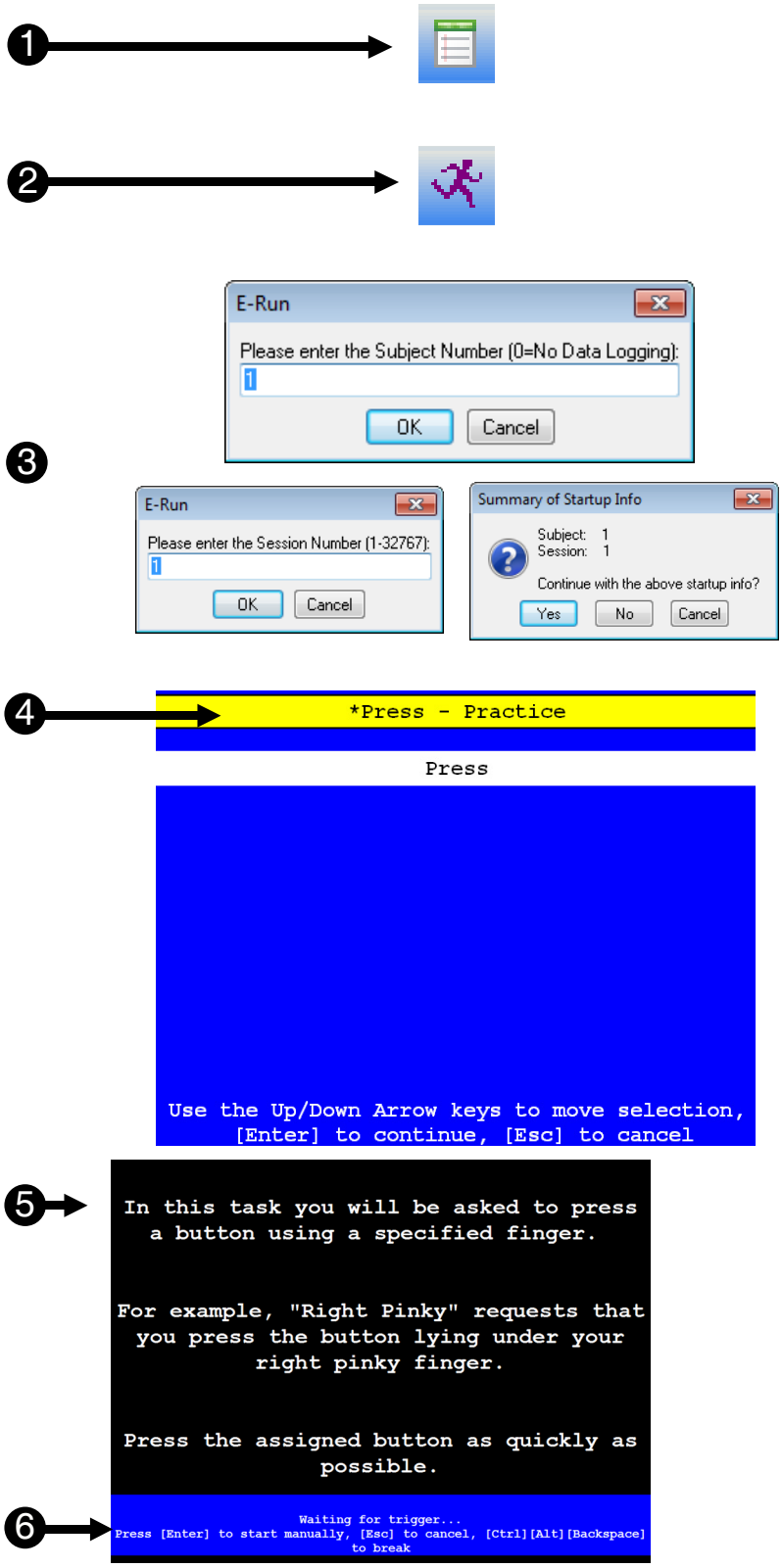
2) ***Click*** the run icon or ***press* F7** to run the paradigm.

3) ***Click* OK** to accept the **default values** for **Subject Number**, **Session Number** and **Summary of Startup Info**.
   ⚠ *If you would like to compare the new .PDAT file to the old .PDAT file, you must change the subject or run number or else the old .PDAT file will be over written.*

4) ***Press* Enter** select the highlighted menu option.

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

6) ***Press* Enter** to manually simulate the scanner trigger pulse and begin the task.
   If you test the experiment at the scanner, in conjunction with the appropriate interface hardware, the task will automatically begin when the technologist initiates the scan from the scanner console.



Chapter 3: **EEfMRI PackageCall Reference**

3.1 Introduction

The following pages describe in detail the fMRI PackageCalls. The formatting is as follows: name of the PackageCall, overview of the PackageCall, parameters of the PackageCall and any relevant examples. The PackageCalls are named by the convention the program expects. This naming convention is not necessary, but is recommended. Naming the PackageCalls otherwise will trigger a dialog box to inform you that you are not naming the PackageCall in the way the program expects.

The overview section describes the function of the PackageCall and other useful information about the PackageCall's features. The parameter section delineates the options available to the user. It is read as follows; the italicized text is the variable. Except in the case of "c" the italicized text will need to be renamed by the user. When the examples are present they serve to clarify points made in the overview and parameters.

fMRISessionInit

Description
Initializes the fMRI support at the Session level. Opens a default PDAT file and writes out the column headers.

Syntax
```
fMRI_SessionInit c, strState,[,vPDATFilename][,vPDATUserColumns]
[,vHandlePreRelease]
```

Default Parameters
```
c, "on"
```

Parameters
c As Context
Sets the current experiment context.

strState As String
Options: *on , off*
Allows user to turn on or off the fMRI Package File support, default is on.

vPDATFilename As Variant
An optional filename for the PDAT file. Specify an empty string to disable the creation of the PDAT file. Default = Same as "EDAT" file with a "PDAT" extension.

vPDATUserColumns As Variant
An optional tab delimited string of column names for user defined columns in the PDAT file. Specify an empty string to disable user defined columns. Default = "User1\tUser2\tUser3\tUser4\tUser5".

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
This is a required call that must exist at the Session level.

fMRIMenuBegin

Description
Implements an interactive menu using a specified List object. The menu is created dynamically by processing the List object to extract the text for menu items from a designated attribute (“MenuItem”) that is expected to be defined on the List. When a menu item is selected at runtime the corresponding row of the List is selected and executed.

Syntax
```
fMRI_MenuBegin c, strMenuState, theMenuList[,vMenuItemAttribute]
[,vHandlePreRelease]
```

Default Parameters
```
c, “on”, RunList
```

Parameters
c As Context

The current experiment context.

strMenuState As String

Enables (“on”) or disables (“off”) the display of the menu at runtime. When the menu is enabled it will be displayed to the user and the Selection, Reset/Exit related properties specified on the List will be ignored. When the menu is disabled the List object will execute normally.

theMenuList As List

The List object that will be used to create the menu. The List object is required to contain an attribute whose contents defines the text that will be used for the menu items that are displayed to the user. By default the attribute is expected to be named “MenuItem”, but you may change this using the vMenuItemAttribute parameter.

vMenuItemAttribute As Variant

Optional name of an attribute on the List that will be used to obtain the text displayed for each menu item. Default = “MenuItem”

vHandlePreRelease As Variant

Optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine’s script. If not specified this will default to True.

Remarks
In the experiment structure the fMRIMenuBegin package call must be followed by the menu List object and the List object must be followed by the fMRIMenuEnd PackageCall.

For example:

fMRIMenuBegin
MenuList
fMRIMenuEnd

It is recommended that this call is placed at the Session level.
See fMRIResetListsByName for information on resetting lists every time the menu is displayed.

fMRIMenuEnd

Description
Completes the implementation of a menu that was defined by the fMRIMenuBegin call.

Syntax
```
fMRI_MenuEnd c, [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context.

Remarks
In the experiment structure the fMRIMenuEnd package call must be preceded by the menu List object and the List object must be preceded by the fMRIMenuBegin PackageCall.

For example,
fMRIMenuBegin
MenuList
fMRIMenuEnd

It is recommended that this call is placed at the Session level.

fMRIRunBegin

Description

Designates the beginning of a new run by writing a RunBegin record to the PDAT file including a run start time and condition. By default the command will overlay a text message on the current screen and then wait for a designated trigger key to start the run. The trigger time is saved as the “zero time” for the run. The default functionality may be overridden by use of various parameters.
If you are handling trigger detection and synchronization yourself you should still call this command after the trigger is received. You should pass in the time stamp of the trigger and a condition.

Syntax

```
fMRI_RunBegin c, nTriggerTime, strConditionId [,vTriggerMessage]
[,vTriggerKeys][,vHandlePreRelease]
```

Default Parameters

```
c, -1, “”
```

Parameters

c As Context

The current experiment context.

nTriggerTime As Long

A millisecond time stamp to associate with the trigger. If -1 then this call will wait until one of the designated trigger keys is received from the Keyboard device before returning. If non-zero, then this value will be used directly as the trigger time and the system will NOT wait for a trigger.

strConditionId As String

A string that identifies the current run condition.

vTriggerMessage As Variant

An optional text message that will be overlaid on the screen when waiting for the trigger. You should pass in an empty string (“”) to disable the message.

vTriggerKeys As Variant

An optional set of keys that will be accepted as the trigger. By default the allowable keys are “{=}”, “{^}”, “{ENTER}”, and “{ESCAPE}”.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine’s script. If not specified this will default to True.

Remarks

Even if you are handling trigger detection and synchronization yourself you should still call this command after the trigger is received to get the RunBegin record written to the PDAT file with the appropriate condition and start time.

fMRIRunEnd

Description
Designates the end of a current run by writing a “RunEnd” record to the PDAT file. An optional timing report will also be generated and displayed to the user.

Syntax
```
fMRI_RunEnd c, nReportDuration [,vReserved][,vHandlePreRelease]
```

Default Parameters
```
c, -1
```

Parameters
c As Context
The current experiment context.

nReportDuration As Long
Specifies how long the report should be in milliseconds unless one of the flag values below is specified.
0 = disable the report (do not show the report)
-1 = leave the report displayed to the user until the “{ENTER}” key is pressed.

vReserved Variant
(none)

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine’s script. If not specified this will default to True.

Remarks
If fMRIRunBegin was called to begin the run then this call is required to complete the run.

fMRILogEvent

Description
Logs an Event record to the PDAT log file. It is recommended that you place this call AFTER the object that you associate with the event has been run.

Syntax
```
fMRI_LogEvent c, theObject, nDuration, strConditionId [,vUserDefinedData]
```

Default Parameters
```
c, ObjectPresentingStimulus, -1, ""
```

Parameters
c As Context
The current experiment context.

theObject As RTERunnableInputObject
The object that corresponds to the event (e.g. Stimulus). The time of the event will be set to the value saved in object's OnsetTime property.

nDuration As Long
The duration that will be associated with the event. The duration passed in will be used directly unless one of the following flag values is supplied.
-1 = Set the duration based on value of the theObject.Duration property.
-2 = Flag the value to be calculated via post processing.

strConditionId As String
A string that describes an experiment condition to associate with the event.

vUserDefinedData As Variant
An optional tab delimited string that provides data for user defined columns.

Remarks
The record ID is hardcoded as "Event".

fMRILogResponseEvent

Description
Logs a Response event record to the PDAT log file. The response information will be extracted from the object passed in. It is recommended that you place this call AFTER the object that you associate with the event has been run and the allowable response time period has expired.

Syntax
```
fMRI_LogResponseEvent c, theObject, nDuration, strConditionId
[,vUserDefinedData] {,vHandlePreRelease}
```

Default Parameters
```
c, ObjectCollectingResponse, -1, ""
```

Parameters
c As Context
The current experiment context.

theObject As RTERunnableInputObject
The object that enabled and collected the response. The time of the event will be set to value saved in object's OnsetTime property.

nDuration As Long
The duration that will be associated with the response event. The duration passed in will be used directly unless one of the following flag values is supplied.
-1 = Set the duration to the value of object's RT property.
-2 = Flag the value to be calculated via post processing.

strConditionId As String
A string that describes an experiment condition to associate with the event.

vUserDefinedData As Variant
An optional tab delimited string that provides data for user defined columns.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
The record ID is hardcoded as "Response".
The RT will be logged as the duration for the event.
Make sure that the allowable response period for your response event has completed before calling this routine.

fMRILogUserEvent

Description
Logs a user defined event record to the PDAT log file.

Syntax
```
fMRI_LogUserEvent c, nOnsetTime, nTargetOnsetTime, strRecordId, nDuration,
strConditionId [,vUserDefinedData]
```

Default Parameters
```
c,-2,-2,“Event”,-2, “”
```

Parameters
c As Context
The current experiment context.

nOnsetTime As Long
The onset time that will be associated with the event. The onset time passed in will be used directly unless one of the following flag values is supplied.
-1 = Get a current timestamp by calling Clock.Read.
-2 = Get an expected timestamp by calling GetNextTargetOnsetTime().

nTargetOnsetTime As Long
The target onset time that will be associated with the event. The target onset time passed in will be used directly unless one of the following flag values is supplied.
-1 = Get a current timestamp by calling Clock.Read
-2 = Get an expected timestamp by calling GetNextTargetOnsetTime()

strRecordId As String
A string id that describes the type of event record described. The fMRI system reserves some default id strings, but the user may define their own as needed to assist in post processing activities.

nDuration As Long
The duration that will be associated with the event. The duration passed in will be used directly unless one of the following flag values is supplied.
-1 = Unused.
-2 = Flag the value to be calculated via post processing.

strConditionId As String
A string that describes an experiment condition to associate with the event.

vUserDefinedData As Variant
An optional tab delimited string that provides data for user defined columns.

Remarks
(none)

fMRIResetListByName

Description
Accepts a comma delimited string of List names and performs a List.Reset call to each List that is named. This will reset the items contained in the list so that all items are once again available for sampling.

Syntax
```
fMRI_ResetListByName c, strListNames, [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context

The current experiment context.

strListNames As String

A comma delimited string containing the names of List objects that should be reset.

vHandlePreRelease As Variant

An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
If you want to reset list every time the menu is displayed, do the following:

For example,
fMRIMenuBegin
fMRIResetListsByName
MenuList
fMRIMenuEnd

It is recommended that this call is placed at the Session level.

fMRICheckForBreak

Description
Checks the UserBreakState. If a break request is detected, the currently running List will be terminated. This call is used to define a location for breaking out of a sequence.

If an immediate break from execution is desired, use Conditional Exit (Ctrl+Alt+Backspace), which will terminate all objects that have HandleConditionalExit=True. The Run Report is unaffected by ConditionalExitState. ConditionalExitState is always cleared by the Menu.

Syntax
```
fMRI_CheckForBreak c [,vHandlePreRelease]
```

Default Parameters
```
c
```

Parameters
c As Context
The current experiment context.

vHandlePreRelease As Variant
An optional parameter that specifies if the HandlePreRelease Routine should be called prior to the execution of this Routine's script. If not specified this will default to True.

Remarks
(none)

Appendix A: **MapperOne Experiment Description**

Introduction

This section of the manual is designed to describe the EEfMRI sample paradigms included with the package installation. The sample experiment is located \My Experiments\fMRI\Samples\ MapperOne. The MapperOne.es2 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 640 X 480, and the included image resources are the same resolution, in order 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 of 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 interstimulus interval (ISI) of two seconds.

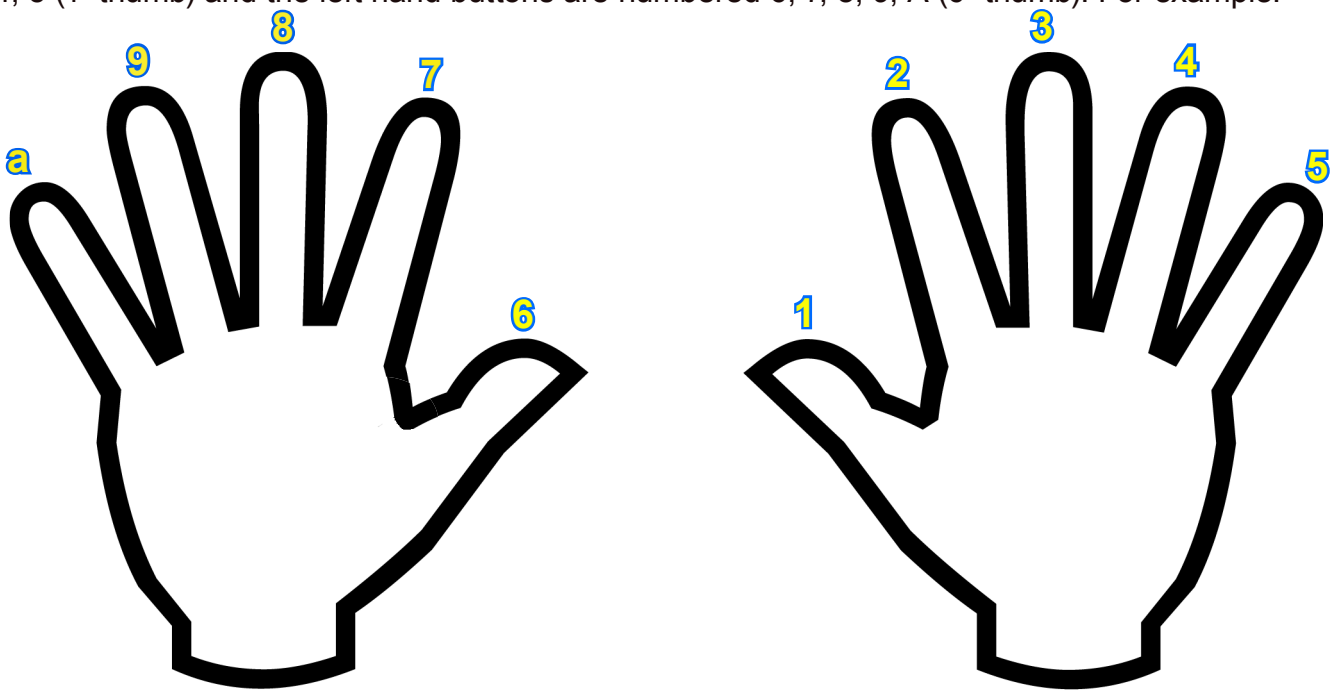
Each run consists of eight blocks and lasts for two minutes and 30 seconds (including two discarded acquisitions Our recommended scanning parameters for 1.5T scanners are:
TE = 35 ms, FOV = 20 cm, TR = 3000 ms, 37 slices, and the discard of the two initial volumes collected per sequence. These timing perimeters will 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. It is highly recommended that the participant practice the task prior to the actual scan.

The emotional pictures task consists of a the 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.

The paper can be found here:
http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1620013

Visual - Motor Task

In this task, the participant will passively observe a fixation for a period of time, observe a series of single digit clues and, then see a flashing checkerboard. Instruct the participant to press the corresponding button on their response unit. 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). For example:



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 will alternate eight times.

6 sec Fixation	18 sec Motor	18 sec Checkerboard

Language Task (Verb Generation)
In this task the participant will passively observe a control stimulus (####) for a period of time and then observe 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.

For example:

Trial	1	2	3	4
Noun	truck	apple	key	dog
Silent Verb	drive	eat	unlock	bark

In trials one and three, the verb could have also 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.

The control blocks and verb generation blocks will alternate several times.

6 sec Fixation	18 sec '###'	18 sec See noun Silently say verb	18 sec '###'	…

Working Memory (Verbal N-Back)

In this task, the participant will perform two alternating letter comparison tasks. During the first comparison task, Zero-Back or target letter 'X', the participant will see 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 will perform a slightly harder task called Two-Back or letter skips task. During the two-back task the participant will need 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.

For example:

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 table above 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.

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

6 sec Fixation	**18 sec Zero-Back**	**18 sec Two-Back**	…

Emotional Pictures

In this task the participant will first passively observe a fixation for six seconds. Then the participant will view a block of control pictures lasting 18 seconds. This will be followed by another rest of six seconds. Then a novel picture block will be presented for 18 seconds. This pattern will repeat for three minutes and 12 seconds.

6 sec Baseline	18 sec Control Pictures	6 sec Rest	18 secs Novel Pictures	6 sec Rest	18 sec Control Pictures	6 sec Rest	18 sec Novel Pictures	6 sec Rest	...

We are not legally able to distribute the control and emotional pictures actually 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 web site:
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 B: **.PDAT File Format**

The .PDAT is the output file produced by the EEfMRI Package File. 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	User defined column 1. These are created by the fMRI_LogUserEvent PackageCall

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

Appendix C: **Celeritas and FOBRS Response Devices**

As mentioned in Section 2.1, while keyboards are often used to collect responses during the experiment development process, they are not compatible for use in the magnet room. Psychology Software Tools offers MR-compatible response devices, utilizing non-ferrous components and fiber optic cabling, that can be used in conjunction with your EEfMRI-enabled experiment.

Celeritas, as well as its predecessor FOBRS, enables a participant in the scanner room to make responses using button response units and/or joysticks. A separate console provides real-time visual confirmation of the responses, enabling a staff member in the control room to monitor the participant's responses. Some of the Celeritas components are shown below.

Celeritas Interface Console

Celeritas Response Unit Options:

Five Button Response Unit

Joystick

To configure your existing experiment to accept responses from Celeritas, refer to the following sections in the *Celeritas Operator Manual*:

1) Chapter 4 Software Installation
2) Section 7.2 Using Celeritas in your E-Prime Experiment.

Note that there is a Celeritas-enabled version of each sample and tutorial experiments.
The Celeritas versions of the sample and tutorial experiments have the suffix of "_Celeritas".
For example, the …Documents\My Experiments\fMRI\Samples\ folder contains the files:
fMRIPress.es2 and fMRIPress_Celeritas.es2.

To configure your existing experiment to accept responses from FOBRS, refer to Chapters 4 and 5 of the *FOBRS Operator Manual*:

Note that if you modify a sample or tutorial experiments to work with one of these response devices, you will most likely want to modify the Instruction slide object. This object accepts the "Enter" key from the participant, since the experiment runs with typical keyboard input.
The text that is shown to the participant as well as the Input Mask that is defined on the Duration/ Input tab of the Instruction slide object should both be modified when using one of the MR-compatible response devices; the "Enter" key should be replaced with one of the buttons that is available to the response device.

Appendix D: **Interrupting an Experiment**

E-Prime 2.0 provides two methods for interrupting an experiment. The first method, available only in E-Prime 2.0 Professional, is the Conditional Exit/Graceful Abort option. This escape sequence is invoked by pressing Ctrl-Alt-Backspace simultaneously. Pressing this key sequence will immediately end the current trial. The Conditional Exit/Graceful Abort option is recommended for use with EEfMRI-enabled experiments. When used with EEfMRI, the Conditional Exit key sequence returns the user to the Menu, assuming that the MenuBegin and MenuEnd package calls are being used.

The second method, checking the User Break State, is invoked with the Ctrl-Shift key sequence. This method sets a user break state variable, accessed via the GetUserBreakState() method. When used with advanced E-Basic scripting techniques, the currently executing list objects can be terminated at the end of the trial or block, and all of the accumulated data that had been collected to the current point of the experiment can be saved. Details on using this method are provided in the sample experiment “Safe Exit of Running Experiment”, which is available on the Samples page of the PST Product Services and Support site. Since the use of the GetUserBreakState() method requires additional E-Prime scripting, the EEfMRI PackageFile includes a PackageCall to assist with this task. See the CheckForBreak routine in the EEfMRI PackageFile for details.

⚠ **NOTE:** *Do not confuse either these methods with the Abort prompt method. This method, invoked by pressing Ctrl-Alt-Shift, will abort an experiment but it does not close out devices or create a data file. A data file can usually be retrieved by running the utility E-Recovery. The Abort method prompt is designed for use during initial experiment development and testing, not during data collection.*

Appendix E: **Technical Support**

Psychology Software Tools, Inc. provides technical support for E-Prime via the PST Product Service and Support web site. You must register online at https://support.pstnet.com to receive technical support. To register you simply need a valid serial number. At the support site, you will also find a Knowledge Base including release notes and a compilation of frequently asked questions. In addition, the support site also includes E-Prime sample paradigms that are available for you to download.

Appendix F: **Contact Information**

For additional information or support

PSYCHOLOGY SOFTWARE TOOLS
Solutions for Research, Assessment, and Education

Contact us at
Psychology Software Tools, Inc
311 23rd Street Extension, Suite 200
Sharpsburg, PA 15215-2821
Phone: 412-449-0078
Fax: 412-449-0079
www.pstnet.com

For Product Service and Support:
Please visit us at https://support.pstnet.com