\README – E-Prime 3.0 LSL PackageFile

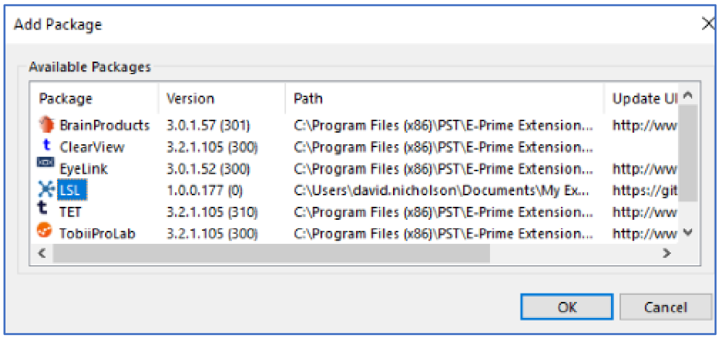
2022-08-25

INSTALLATION

1. Uncompress the distribution file.
2. Navigate to *"...\Documents\My Experiments\3.0\Packages"* and create the Packages folder if necessary.
3. Move the *LSL* folder to the folder *"...\Documents\My Experiments\3.0\Packages"*
    a. The LSL folder contains the LSL PackageFile and related resources.
    b. By default, it must exist in the E-Prime 3.0 user Packages folder to be located by E-Studio 3.0.
    c. To keep the LSL folder in an alternate location, select Tools > Options > Packages from the E-Studio menu to add a specific folder to the list of PackageFile Search Folders that E-Studio uses to locate for PackageFiles. For more information, see PACKAGEFILE: Loading PackageFiles in E-Studio [22899].
4. Move the *LSL Samples* folder to the folder *"...\Documents\My Experiments\3.0"*
    a. The *LSL Samples* folder contains any sample experiments related to the LSL PackageFile.
    b. Each sample experiment subfolder contains the *lsl.dll* file. This dynamic link library contains all the library routines that can be accessed from E-Basic script.
    c. The lsl.dll file MUST exist in the experiment folder (e.g., where the *.es3 and *.ebs3 experiment files exist), or must exist in a folder that is listed in the system PATH environment variable.
    d. If the LSL PackageFile cannot load the *lsl.dll* successfully when the experiment is launched, a notification dialog will be shown.
    e. The most recently updated lsl.dll can be found at https://github.com/sccn/liblsl/releases.
    NOTE: *E-Prime requires the use of the 32-bit dll (i.e. liblsl-1.xx-Win_i386.zip). The current documentation regarding the routines contained in the library can be obtained from https://github.com/sccn/labstreaminglayer*.
    f. The most recent LabRecorder can be found at https://github.com/labstreaminglayer/App-LabRecorder/releases.
    NOTE: *The lsl.dll may also be copied and used in the LabRecorder release with E-Prime.*

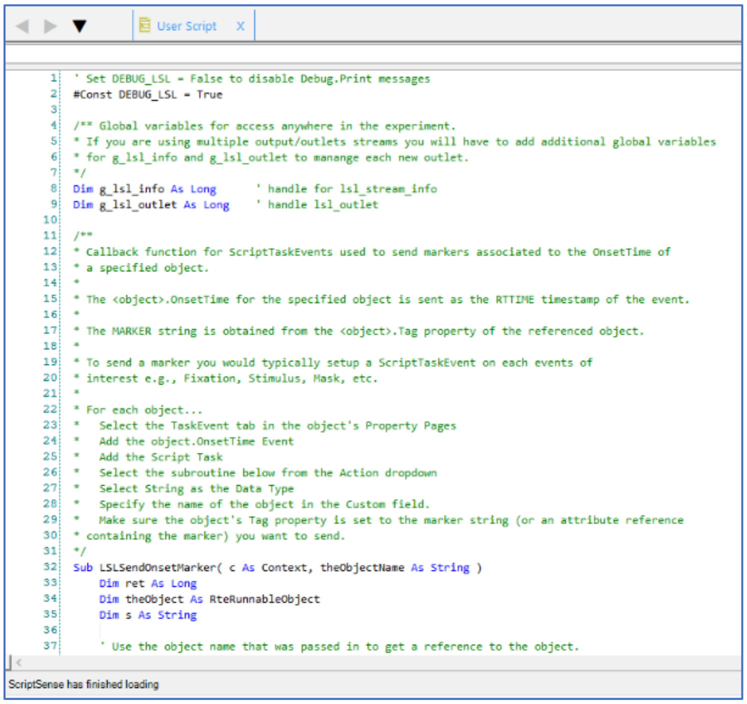
ADDING LSL SUPPORT TO AN E-PRIME EXPERIMENT

1. Follow the instructions above to setup the appropriate folder structure.
2. Copy the *lsl.dll* file from one of the *LSL Samples* folder to the folder containing the E-Prime 3.0 to be used.
3. Add the LSL PackageFile to the experiment by navigating to the Experiment Object Property Pages, selecting the Packages tab, and selecting the LSL PackageFile. For more information, see the “Adding PackageFiles” section of E-STUDIO: PackageCall Object [22713].



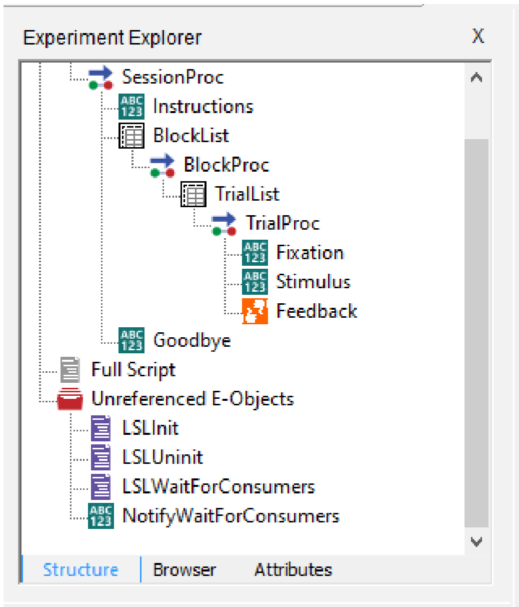
Package	Version	Path	Update UI
BrainProducts	3.0.1.57 (301)	C:\Program Files (x86)\PST\E-Prime Extension...	http://ww
ClearView	3.2.1.105 (300)	C:\Program Files (x86)\PST\E-Prime Extension...	
EyeLink	3.0.1.52 (300)	C:\Program Files (x86)\PST\E-Prime Extension...	http://ww
LSL	1.0.0.177 (0)	C:\Users\david.nicholson\Documents\My Ex...	https://git
TET	3.2.1.105 (310)	C:\Program Files (x86)\PST\E-Prime Extension...	http://ww
TobiiProLab	3.2.1.105 (300)	C:\Program Files (x86)\PST\E-Prime Extension...	http://ww

4. Launch a second instance of E-Studio and open the LSLOddBall.es3 from the ...\Documents\My Experiments\3.0\LSL Sample experiments folder.
5. Copy the contents of the User Script from the LSLOddBall experiment into the User Script section of the experiment.
    a. This User Script defines three example Subroutines that can be called from a Script Task event (or optionally from InLine script).
        i. The *LSLSendOnsetMarker* subroutine can be called (typically from a Script Task Event) to send a marker via LSL at an object’s OnsetTime. The string name of the object is passed as a parameter. The object’s .OnsetTime is sent as the timestamp for the event, and the contents of object’s .Tag property is sent as the marker.
        ii. The *LSLSendStimulusRTTimeMarker* subroutine can be called (typically from InLine Script) to send a marker via LSL at the Stimulus response time. The RTTime and is sent as the timestamp. The contents of the object’s .Tag property is sent as the marker.
        iii. The *LSLSendTimestampMarker* subroutine can be called (typically from InLine Script) to send a marker via LSL at the current time. The current time (read from the E-Prime clock) is sent as the timestamp. A string or attribute reference is sent as the marker.



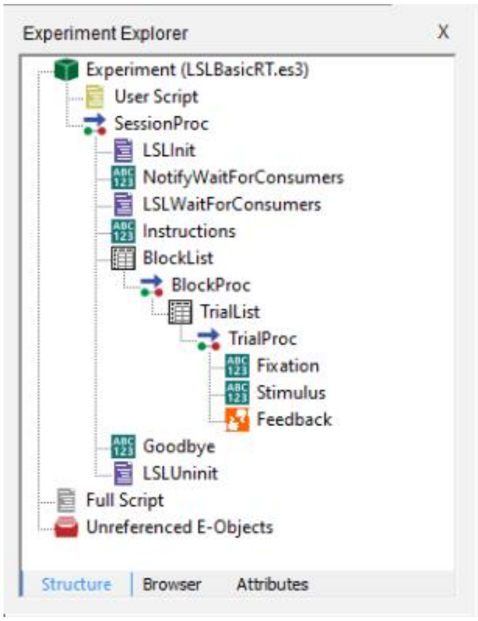
```
' Set DEBUG_LSL = False to disable Debug.Print messages
#Const DEBUG_LSL = True

/** Global variables for access anywhere in the experiment.
* If you are using multiple output/outlets streams you will have to add additional global variables
* for g_lsl_info and g_lsl_outlet to manange each new outlet.
*/
Dim g_lsl_info As Long       ' handle for lsl_stream_info
Dim g_lsl_outlet As Long     ' handle lsl_outlet

/**
* Callback function for ScriptTaskEvents used to send markers associated to the OnsetTime of
* a specified object.
*
* The <object>.OnsetTime for the specified object is sent as the RTTIME timestamp of the event.
*
* The MARKER string is obtained from the <object>.Tag property of the referenced object.
*
* To send a marker you would typically setup a ScriptTaskEvent on each events of
* interest e.g., Fixation, Stimulus, Mask, etc.
*
* For each object...
*   Select the TaskEvent tab in the object's Property Pages
*   Add the object.OnsetTime Event
*   Add the Script Task
*   Select the subroutine below from the Action dropdown
*   Select String as the Data Type
*   Specify the name of the object in the Custom field.
*   Make sure the object's Tag property is set to the marker string (or an attribute reference
* containing the marker) you want to send.
*/
Sub LSLSendOnsetMarker( c As Context, theObjectName As String )
    Dim ret As Long
    Dim theObject As RteRunnableObject
    Dim s As String

    ' Use the object name that was passed in to get a reference to the object.
```

6. Copy the following objects from the LSLOddBall experiment into the Unreferenced E-Objects section of the experiment.



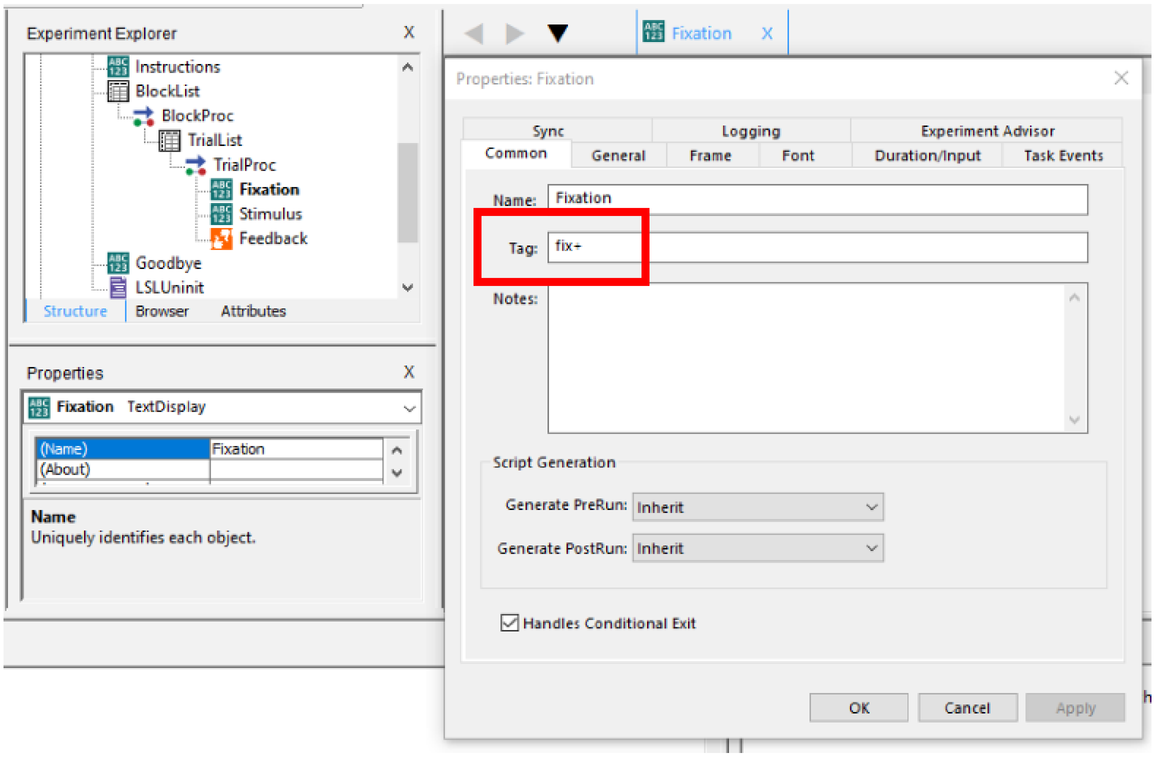
   a. *LSLInit* – An InLine object with script that initializes LSL
   b. *LSLUninit* – An InLine object with script that uninitializes LSL
   c. *LSLWaitForConsumers* – An InLine object with script that waits (30 seconds) for any LSL Consumers connect to the experiment
   d. *NotifyWaitForConsumers* - (optional) TextDisplay to display a message while waiting for LSL Consumers to connect to the experiment

7. Move the above objects into the following equivalent locations in the experiment structure.



CONFIGURING THE EXPERIMENT'S MARKERS

1. Set the <object>.Tag property to the marker string that should be sent (e.g., "fix+" to mark the onset of the Fixation object).



Properties: Fixation

Name: Fixation

Tag: fix+

Notes:

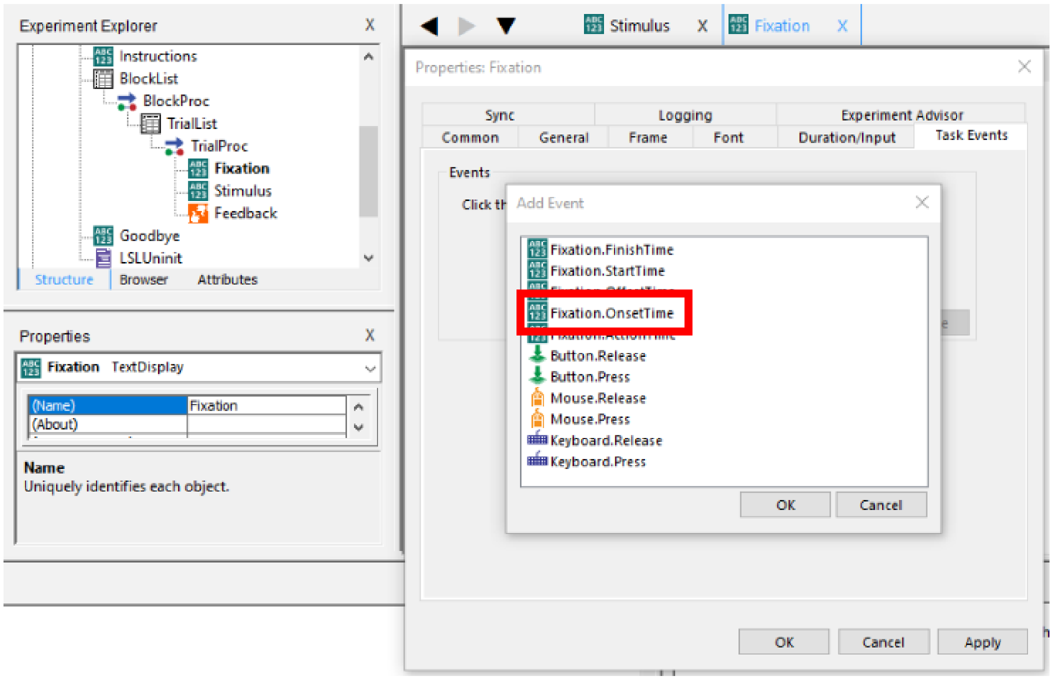
Script Generation

Generate PreRun: Inherit

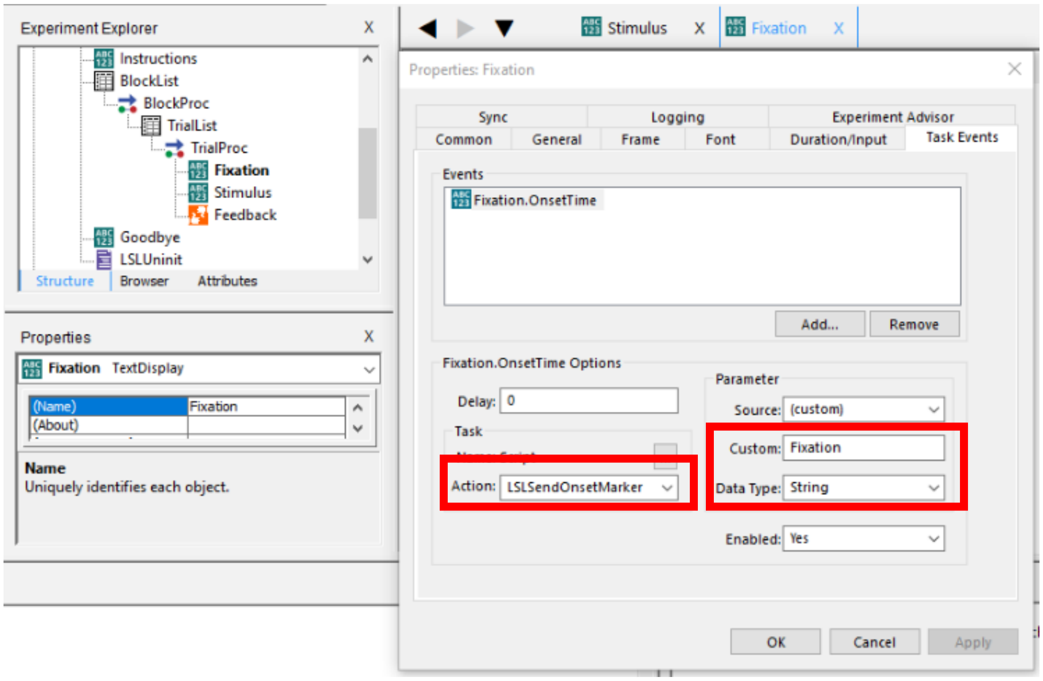
Generate PostRun: Inherit

☑ Handles Conditional Exit

2. Create a Script Task Event on the <object>.OnsetTime Event.



3. Select the *LSLSendOnsetMarker* subroutine from the *Action* dropdown. Specify the name of the object in the *Custom* field. Specify String as the Data Type.



(Name): Fixation

Events: Fixation.OnsetTime

Delay: 0

Action: LSLSendOnsetMarker

Source: (custom)

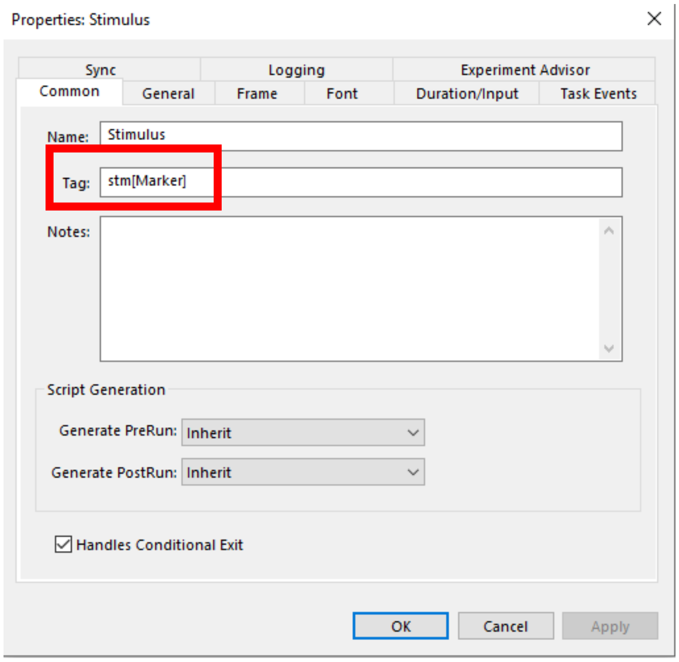
Custom: Fixation

Data Type: String

Enabled: Yes

4. Repeat steps 1-3 to create Task Events for both the Stimulus and Feedback objects.

NOTE: *The <object>.Tag property can be set to an Attribute reference if the marker value varies trial by trial (e.g., "[Marker]"). This sends a marker unique to each type of stimulus presented by the Stimulus object on each trial.*



Name: Stimulus

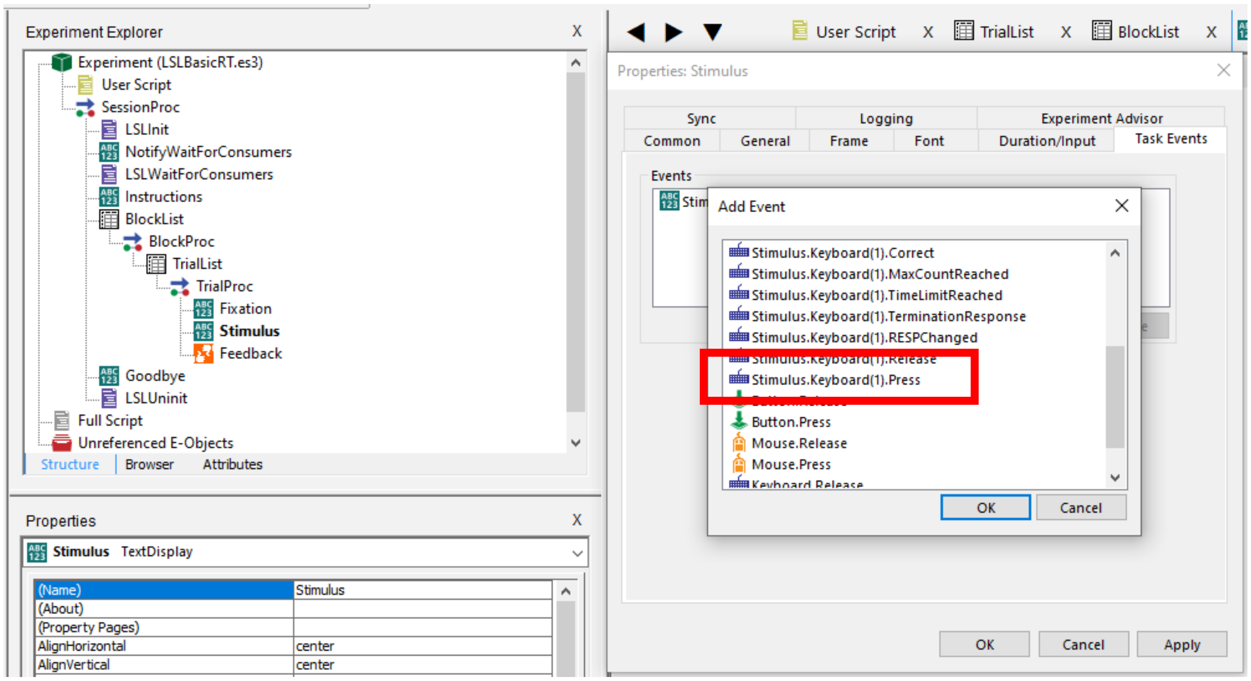
Tag: stm[Marker]

Generate PreRun: Inherit

Generate PostRun: Inherit

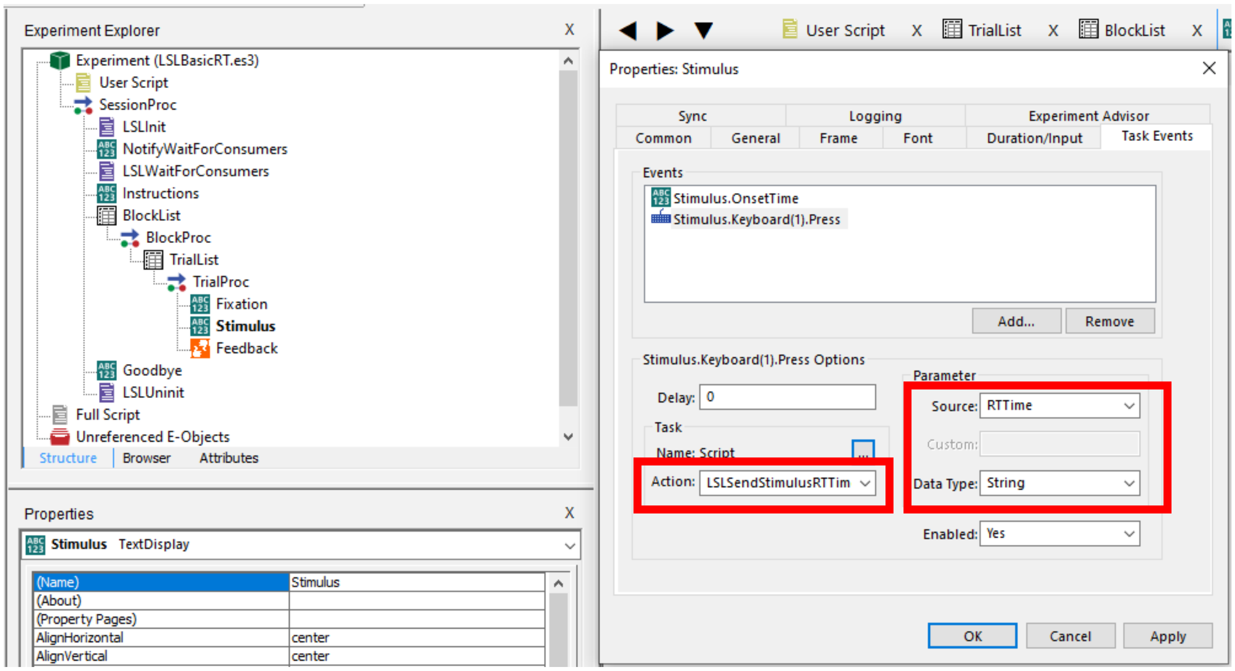
☑ Handles Conditional Exit

5. Create a Script Task Event on the Stimulus.Keyboard(1).Press Event.



(Name)	Stimulus
(About)	
(Property Pages)	
AlignHorizontal	center
AlignVertical	center

6. Select the *LSLSendStimulusRTTime* subroutine from the Action dropdown. Set the Source to RTTime. Specify String as Data Type.



Events:
Stimulus.OnsetTime
Stimulus.Keyboard(1).Press

Stimulus.Keyboard(1).Press Options

Delay: 0

Task

Name: Script

Action: LSLSendStimulusRTTim

Parameter

Source: RTTime

Custom:

Data Type: String

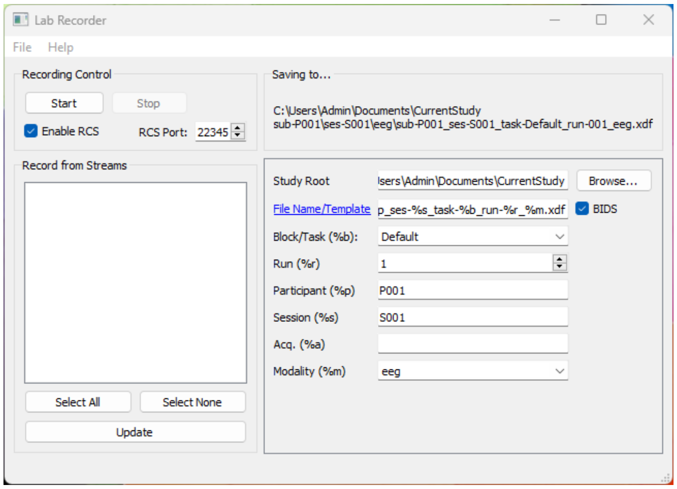
Enabled: Yes

(Name)	Stimulus
(About)	
(Property Pages)	
AlignHorizontal	center
AlignVertical	center

NOTE: *This subroutine is specific to an E-Object named Stimulus that contains an input mask. If your object is not named Stimulus, make the appropriate changes to the subroutine in the User Script.*

RUNNING AN EPRIME EXPERIMENT WITH LSL SUPPORT

1. Launch the LSL Consumer application(s). The LSL Lab Recorder is used in this example as a consumer.



☑ Enable RCS

RCS Port: 22345

Saving to...

C:\Users\Admin\Documents\CurrentStudy
sub-P001\ses-S001\eeg\sub-P001_ses-S001_task-Default_run-001_eeg.xdf

Study Root: Isers\Admin\Documents\CurrentStudy

File Name/Template: p_ses-%s_task-%b_run-%r_%m.xdf ☑ BIDS

Block/Task (%b): Default

Run (%r): 1

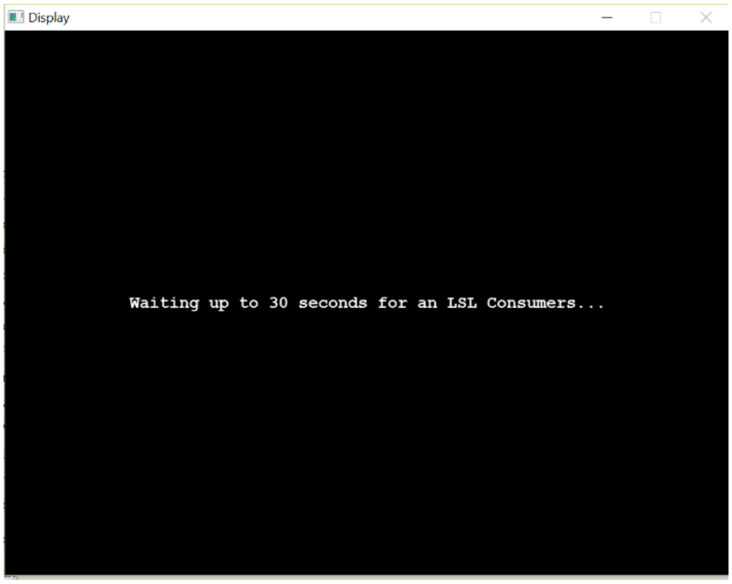
Participant (%p): P001

Session (%s): S001

Acq. (%a):

Modality (%m): eeg

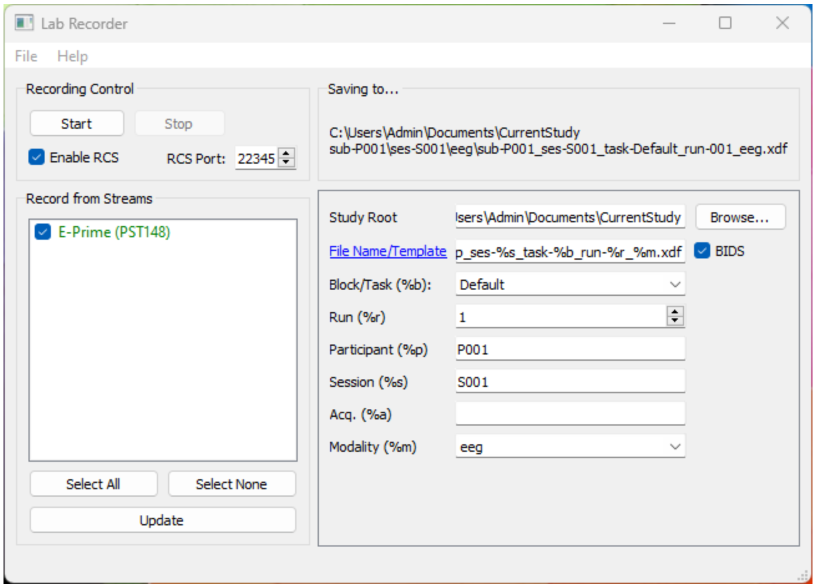
2. Launch the E-Prime experiment.



Waiting up to 30 seconds for an LSL Consumers...

NOTE: *We recommended launching/testing the experiment in Windowed mode while Lab Recorder is running on the E-Prime machine. This allows switching between the applications when testing the experiment on a single machine.*

3. After the experiment launches, configure the LSL Consumer application to record from the “E-Prime” stream and start the recording.



Saving to...

C:\Users\Admin\Documents\CurrentStudy
sub-P001\ses-S001\eeg\sub-P001_ses-S001_task-Default_run-001_eeg.xdf

☑ Enable RCS

RCS Port: 22345

☑ E-Prime (PST148)

Study Root: Isers\Admin\Documents\CurrentStudy

File Name/Template: p_ses-%s_task-%b_run-%r_%m.xdf

☑ BIDS

Block/Task (%b): Default

Run (%r): 1

Participant (%p): P001

Session (%s): S001

Acq. (%a):

Modality (%m): eeg

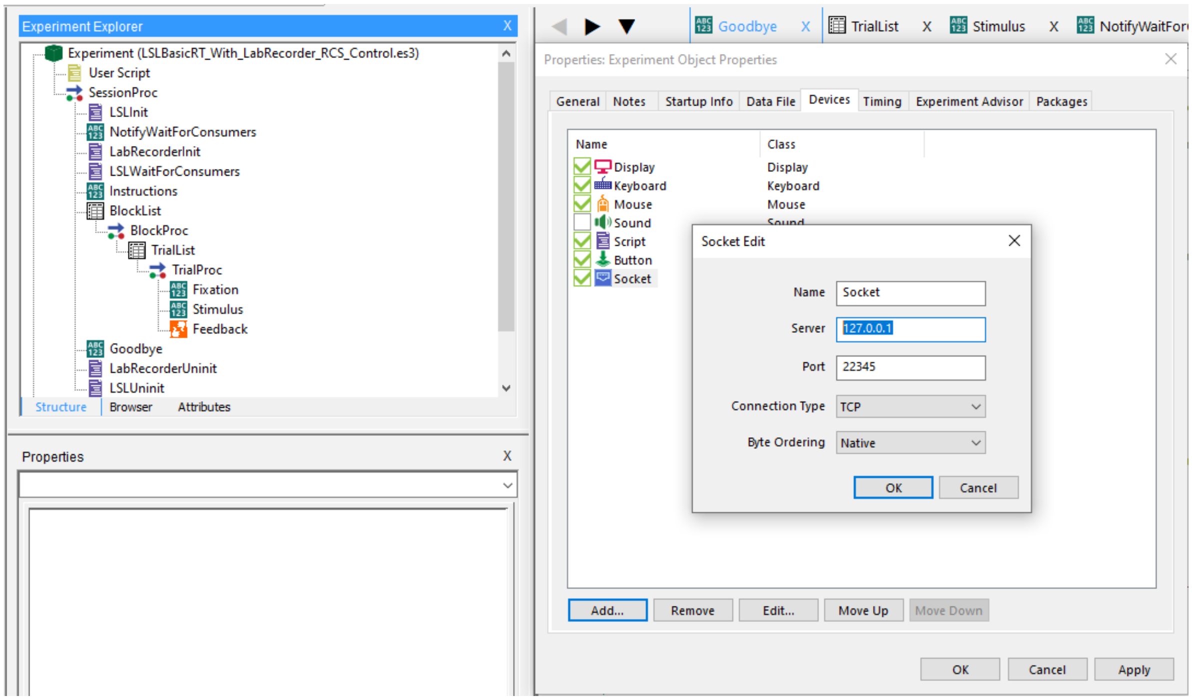
NOTE: *The Update button may need to be pressed to list the E-Prime stream.*

4. Stop the recording after the experiment completes. When using Lab Recorder, the contents can be reviewed in the *.xdf file to confirm the markers.

USING RCS CONTROL IN LABRECORDER

LabRecorder can be controlled by RCS commands. These commands are documented at https://github.com/labstreaminglayer/App-LabRecorder. The "BasicRT With LabRecorder RCS Control" sample demonstrates how to control LabRecorder from E-Prime. Follow the steps below to enable RCS control in an experiment:

1. Add a Socket device to the experiment by navigating to the Experiment object Property Pages > Devices tab > selecting Add… > and clicking the Socket device.
2. Open its properties and set the Server field to the IP address of the LabRecorder machine. Verify the Port number matches the one set in LabRecorder as well.



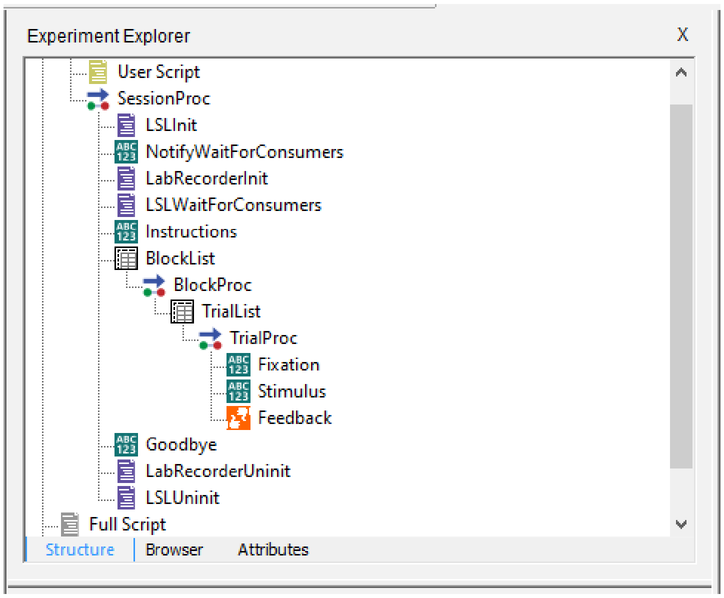
Name	Class
☑ Display	Display
☑ Keyboard	Keyboard
☑ Mouse	Mouse
□ Sound	Sound
☑ Script	
☑ Button	
☑ Socket	

Socket Edit

Name	Socket
Server	127.0.0.1
Port	22345
Connection Type	TCP
Byte Ordering	Native

NOTE: *For assistance on locating an IP address see: INFO: Locate Your IP Address and Ping Another Computer [31363].*

3. Copy the LabRecorderInit and LabRecorderUninit to the appropriate locations in the experiment.



NOTE: *"Enable RCS" must be checked in LabRecorder.*

TROUBLESHOOTING LSL

1. For specific usage information of Lab Recorder, see https://github.com/labstreaminglayer/App-LabRecorder#usage.
2. Unable to connect to a stream?
   a. Check the connection and firewall settings on the LSL machine: https://labstreaminglayer.readthedocs.io/info/network-connectivity.html
   b. Check the connection and firewall settings on the E-Prime machine: INFO: Windows Firewall Permissions with E-Prime [26269]
   c. For additional E-Prime support, submit a request at https://support.pstnet.com/hc/en-us/requests/new