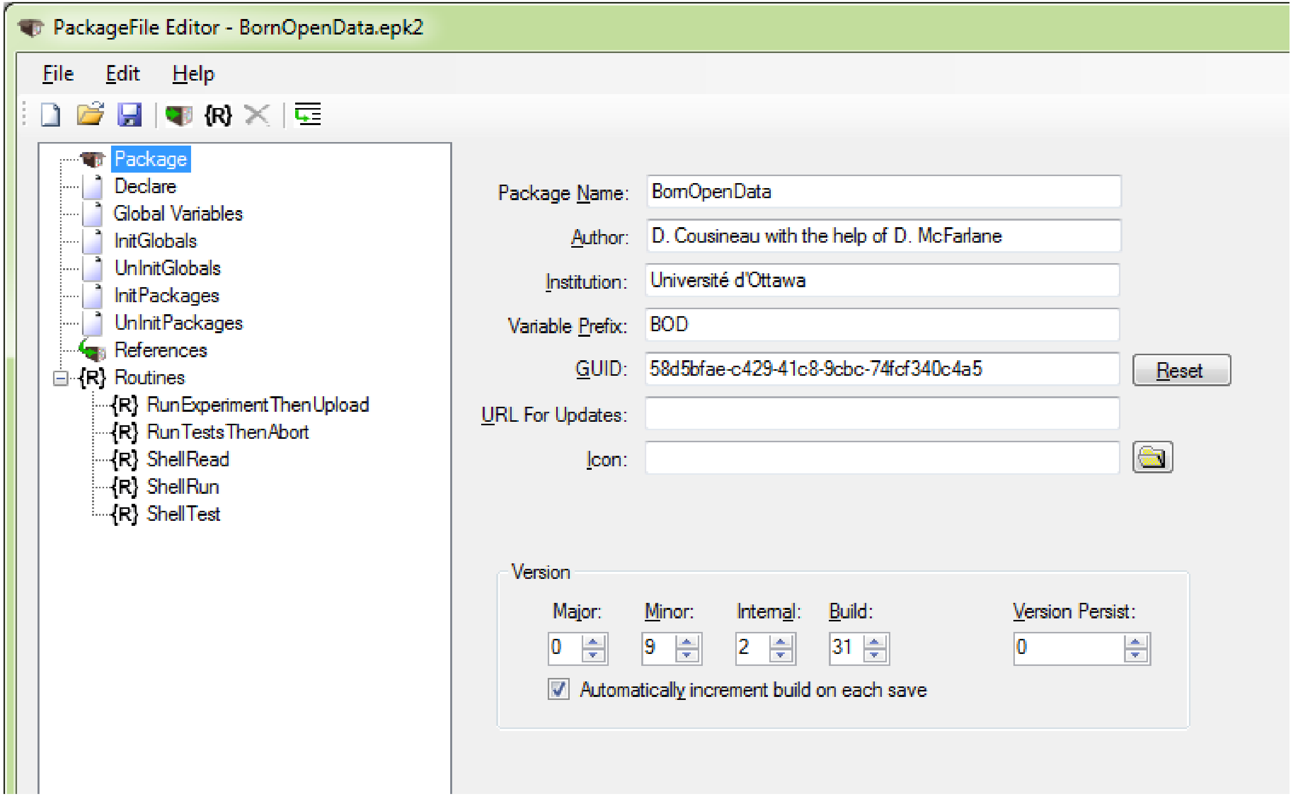
Born-Open Data for E-Prime: Source code of the package

Denis Cousineau

This document is just a series of cut-and-paste of the code found in the BornOpenData package for E-Prime version 2 (file BornOpenData.epk2). It is accessible to those who have an E-Prime license, but for the benefit of all, I reproduce it here. The code refers to the sections seen in Figure 1, left column. Note that some of the sections are empty; only comments are present.

Figure 1: Screen capture of the main page of the package.



Package Name: BornOpenData

Author: D. Cousineau with the help of D. McFarlane

Institution: Université d'Ottawa

Variable Prefix: BOD

GUID: 58d5bfae-c429-41c8-9cbc-74fcf340c4a5

URL For Updates:

Icon:

Major: 0 Minor: 9 Internal: 2 Build: 31 Version Persist: 0

☑ Automatically increment build on each save

Listing 1: Code for the ‘Declare’ section

```
'********************************************************
' The following code is from the package BornOpenData.epk2
' this is the "Declare" declation section
'********************************************************

' No DLL to declare; no data structures needed.
```

Listing 2: Code for the ‘Global Variables’ section

```
'*****************************************************
' The following code is from the package BornOpenData.epk2
' this is the "Global Variables" declation section
'*****************************************************

'This section of script is inserted at the top section of the script.

'Store whether an upload must be done (0) of if we are in test mode (1)
Dim BOD_SkipUpload As Integer ' default 0 means that upload is performed

'Store the file base name of the data file and the experiment name
Dim  BOD_fBasename As String
Dim  BOD_eBasename As String

'Store subject and session numbers
Dim BOD_subj As Integer
Dim BOD_sess As Integer

'Store parameters from RunExperimentThenUpload
Dim BOD_User As String
Dim BOD_Password As String
Dim BOD_Experiment As String
Dim BOD_Owner As String

' two ancilliary functions
Function BOD_hr(ByVal r As Integer)
If r = 0 Then
    BOD_hr="Ok!"
Else
    BOD_hr="no..."
End If
End Function

Sub BOD_PrintDiagnosticToDebug (ByRef result() As Integer)
debug.print "========DIAGNOSTIC␣RESULTS========"
debug.print "␣␣CMD␣is␣accessible␣␣␣␣␣␣␣␣␣␣␣␣␣\t" & BOD_hr(result(1,1))
debug.print "␣␣git␣is␣installed␣␣␣␣␣␣␣␣␣␣␣␣␣␣\t" & BOD_hr(result(2,1))
debug.print "␣␣computer␣is␣online␣␣␣␣␣␣␣␣␣␣␣␣\t" & BOD_hr(result(2,2))

debug.print "␣␣user,␣owner,␣repo␣exist:␣␣␣␣␣␣\t" & BOD_hr(result(3,1))
debug.print "␣␣user␣can␣git␣add␣a␣file:␣␣␣␣␣␣\t" & BOD_hr(result(4,2))
debug.print "␣␣user␣can␣git␣committ␣a␣file:␣␣\t" & BOD_hr(result(4,3))
debug.print "␣␣user␣can␣git␣push␣a␣file:␣␣␣␣␣\t" & BOD_hr(result(4,4))
debug.print "=======DIAGNOSTIC␣COMPLETED======="
End Sub

' In the script section of E-Prime, set BOD_Debug = 1 to activate debug prints in the debug panel.
Sub BOD_PrintToDebug(ByVal stringname As String, ByVal stringcontent As String, ByVal stringresult As Variant)
If BOD_Debug =1 Then
    If stringcontent = "" Then
```

```
        debug.print stringname
    Else
        debug.print stringname & ":␣‘‘" & stringcontent & "‘‘␣responded␣" & stringresult
    End If
End If
End Sub

'********************************************************
' The above code is from the package BornOpenData.epk2
' this is the global variables declation section
'********************************************************
```

Listing 3: Code for the ‘InitGlobals’ section

```
'********************************************************
' The following code is from the package BornOpenData.epk2
' this is the "InitGlobals" declation section
'********************************************************

' Nothing to initialize prior to context setting
```

Listing 4: Code for the ‘UninitGlobals’ section

```
'********************************************************
' The following code is from the package BornOpenData.epk2
' this is the "UninitGlobals" section
'********************************************************

' This section is the very last to run in the E-Prime script

' Nothing to uninitialize
```

Listing 5: Code for the ‘InitPackages’ section

```
'********************************************************
' The following code is from the package BornOpenData.epk2
' this is the "InitPackages" declation section
'********************************************************

' This section is the second to run in the E-Prime script

' Nothing to initialize for the package
```

Listing 6: Code for the ‘UnInitPackages’ section

```
'********************************************************
' The following code is from the package BornOpenData.epk2
' this is the "UnInitPackages" declation section
'********************************************************

' This section of script is inserted InLine inside the UnInitPackages routine generated by
' E-Studio in the Full Script view.

' This section is the second to last to run in the E-Prime script
' At this time, the .edat2 datafile has been generated and closed so it can be manipulated.

' in case RunTestsThenAbort is used.
If BOD_SkipUpload = 1 Then
    GoTo BOD_end
End If

Dim   result(7) As Integer
Dim   compday  As String, comptime As String, compname As String
Dim   id       As String
Const cmdstr1a As String = "date␣/T"
Const cmdstr1b As String = "time␣/T"
Const cmdstr1c As String = "set␣computername"
Const cmdstr2  As String = "rmdir␣/s␣/q␣localRep"
Dim   cmdstr3a As String
Const cmdstr3b As String = "mkdir␣LocalRep\\rawdata"
Dim   cmdstr4  As String
Dim   cmdstr5  As String
Dim   cmdstr6  As String
Const cmdstr7  As String = "cd␣LocalRep␣&␣git␣push"


' 1- Get some identifiers
compday  = BornOpenData_ShellRead(cmdstr1a) ' date
comptime = BornOpenData_ShellRead(cmdstr1b) ' time
compname = BornOpenData_ShellRead(cmdstr1c) ' computer name
id = "BornOpenData4E-Prime␣uploaded␣SUBJECT=" + CStr(BOD_subj) + ",␣SESSION=" + CStr(BOD_sess) + "␣from␣" + compname + "␣on␣DATE=" + compday + ":" + comptime + "␣RND=" + CStr(Random(0,1000))
BOD_PrintToDebug "id\t\t", id, 0

' 2- if subfolder localRep exists, delete it
result(2) = BornOpenData_ShellTest(cmdstr2) 'rmdir ...
BOD_PrintToDebug "cmdstr2\t", cmdstr2, result(2)

' 3a- git clone ...
If BOD_Password = "" Then
    cmdstr3a = "git␣clone␣-q␣""file://" + trim(BOD_Owner) + "\\" + trim(BOD_Experiment) + """␣localRep"
Else
    cmdstr3a = "git␣clone␣-q␣""https://" + trim(BOD_User) + ":" + trim(BOD_Password) + "@github.com/" + trim(BOD_Owner) + "/" + trim(BOD_Experiment) + """␣localRep"
End If
result(3) = BornOpenData_ShellTest(cmdstr3a) ' git clone ...
BOD_PrintToDebug "cmdstr3a\t", cmdstr3a, result(3)

' 3b- if subfolder rawdata does not exist, create it
result(3) = BornOpenData_ShellTest(cmdstr3b)
result(3) = BornOpenData_ShellTest(cmdstr3b)
```

```
BOD_PrintToDebug "cmdstr3b\t", cmdstr3b, result(3) ' mkdir LocalRep\\rawdata


' 4- Do git add fBasename + .edat2
' 4a- Copy edat* file
cmdstr4 = "copy␣" + BOD_fBasename + ".edat?␣localRep\\rawdata␣&␣cd␣localRep␣&␣git␣add␣rawdata\\" + BOD_fBasename + ".edat?␣&␣cd␣.."
result(4) = BornOpenData_ShellTest(cmdstr4)
BOD_PrintToDebug "cmdstr4\t", cmdstr4, result(4) ' copy then git add
' 4b- Copy txt file
cmdstr4 = "copy␣" + BOD_fBasename + ".txt␣localRep\\rawdata␣&␣cd␣localRep␣&␣git␣add␣rawdata\\" + BOD_fBasename + ".txt␣&␣cd␣.."
result(4) = BornOpenData_ShellTest(cmdstr4)
BOD_PrintToDebug "cmdstr4\t", cmdstr4, result(4) ' copy then git add

' 5- update subjectsLog.txt
' ''''-2-''' subjectLog content
' '''' BornOpenData_ShellRun "echo Uploaded " + BOD_fBasename + ".edat2: " + BOD_hr(result(6)) + " >> Status.txt"
BornOpenData_ShellRun "echo␣" + BOD_eBasename + "\t" + CStr(BOD_subj) + "\t" + CStr(BOD_sess) + "\t" + compday + "\t" + comptime + "\t" + compname + "\tUploaded␣␣>>␣localRep\\subjectsLog.txt"
cmdstr5 = "cd␣localRep␣&␣git␣add␣subjectsLog.txt␣&␣cd␣.."
result(5) = BornOpenData_ShellTest(cmdstr5)
BOD_PrintToDebug "cmdstr5\t", cmdstr5, result(5) ' git add

' 6- Do git commit --message "adding subject, session from COMPUTER"
cmdstr6 = "cd␣localRep␣&␣git␣commit␣--message=""" + id + """␣&␣cd␣.."
result(6) = BornOpenData_ShellTest(cmdstr6) ' git commit
BOD_PrintToDebug "cmdstr6\t", cmdstr6, result(6)

' 7- Do git push
result(7) = BornOpenData_ShellTest(cmdstr7) ' cd then git push
BOD_PrintToDebug "cmdstr7\t", cmdstr7, result(7)

BOD_end:
    ''' Clean up my place
    BornOpenData_ShellRun "rmdir␣/s␣/q␣localTestRep"
    BornOpenData_ShellRun "rmdir␣/s␣/q␣localRep"
    BOD_PrintToDebug "␣␣␣␣␣Cleaned␣up␣temporary␣folders","",""


'********************************************************
BOD_PrintToDebug ">>>␣UnInitPackages␣section␣of␣BornOpenData␣package␣completed...","",""
'********************************************************

'********************************************************
' The above code is from the package BornOpenData.epk2
' this is the UnInitPackages section
'********************************************************
```

Listing 7: PreRun code for the ‘RunExperimentThenUpload’ section

```
'*******************************************************
' The following code is from the package BornOpenData.epk2
' This is the PreRun code of the sub BornOpenData_RunExperimentThenUpload
'*******************************************************

' It has to be in PreRun because the context c is a local variable

' Get base file name for the datafile
BOD_fBasename = c.GetAttrib("Datafile.BaseName")
BOD_eBasename = c.GetAttrib("Experiment")

' Get subject and session number
BOD_subj = c.GetAttrib("Subject")
BOD_sess = c.GetAttrib("Session")

'*******************************************************
' The above code is from the package BornOpenData.epk2
' This is the PreRun code of the sub BornOpenData_RunExperimentThenUpload
'*******************************************************
```

Listing 8: Code for the ‘RunExperimentThenUpload’ section

```
'*******************************************************
' The following code is from the package BornOpenData.epk2
' this is the script section of the sub RunExperimentThenUpload
'*******************************************************

'Store parameters from RunExperimentThenUpload into global variables
BOD_User = user
BOD_Password = password
BOD_Experiment = experiment
BOD_Owner = owner

' show some debug information if requested
BOD_PrintToDebug "User\t\t", "BOD_User", BOD_User
BOD_PrintToDebug "Password\t", "BOD_Password", BOD_Password
BOD_PrintToDebug "Experiment\t", "BOD_Experiment", BOD_Experiment
BOD_PrintToDebug "Owner\t\t", "BOD_Owner", BOD_Owner

' those were set from PreRun as they require an active context
BOD_PrintToDebug "Filename\t", "c.GetAttrib(""Datafile.BaseName"")", BOD_fBasename
BOD_PrintToDebug "Subject\t", "c.GetAttrib(""Subject"")", BOD_subj
BOD_PrintToDebug "Session\t", "c.GetAttrib(""Session"")", BOD_sess

BOD_PrintToDebug ">>>␣Script␣section␣of␣RunExperimentThenUpload␣completed...", "", ""


'*******************************************************
' The above code is from the package BornOpenData.epk2
' this is the script section of the sub RunExperimentThenUpload
'*******************************************************
```

Listing 9: PreRun code for the ‘RunTestsThenAbort’ section

```
'*****************************************************
' The following code is from the package BornOpenData.epk2
' This is the PreRun code of the sub BornOpenData_RunTestsThenAbort
'*****************************************************

' It has to be in PreRun because the context c is a local variable

' Get base file name for the datafile
BOD_fBasename = c.GetAttrib("Datafile.BaseName")
BOD_eBasename = c.GetAttrib("Experiment")

' Get subject and session number
BOD_subj = c.GetAttrib("Subject")
BOD_sess = c.GetAttrib("Session")

'*****************************************************
' The above code is from the package BornOpenData.epk2
' This is the PreRun code of the sub BornOpenData_RunTestsThenAbort
'*****************************************************
```

Listing 10: Code for the ‘RunTestsThenAbort’ section

```
'*****************************************************
' The following code is from the package BornOpenData.epk2
' this is the script section of the sub RunTestsThenAbort
'*****************************************************

Dim id As Variant
Dim result(5,5) As Integer
Dim compday As String, comptime As String, compname As String

Const cmdstr0 As String = "echo␣Is␣it␣running␣this?"
Const cmdstr1a As String = "date␣/T"
Const cmdstr1b As String = "time␣/T"
Const cmdstr1c As String = "set␣computername"
Const cmdstr2a As String = "git␣--version"
Const cmdstr2b As String = "ping␣www.github.com␣-n␣1"
Dim cmdstr3a As String
Dim cmdstr4a As String
Const cmdstr4b As String = "cd␣LocalTestRep␣&␣git␣add␣testBornOpenData.txt"
Dim cmdstr4c As String
Const cmdstr4d As String = "cd␣LocalTestRep␣&␣git␣push"
Dim cmdstr4e As String

''' PART 1 ''' Is the terminal accessible ? '''
result(1,1) = BornOpenData_ShellTest(cmdstr0)
BOD_PrintToDebug "cmdstr0", cmdstr0, result(0,0)

''' Get some identifiers '''
```

```
compday  = BornOpenData_ShellRead(cmdstr1a)
comptime = BornOpenData_ShellRead(cmdstr1b)
compname = BornOpenData_ShellRead(cmdstr1c)

''' PART 2 ''' Run basic tests '''
result(2,1) = BornOpenData_ShellTest(cmdstr2a)
BOD_PrintToDebug "cmdstr2a", cmdstr2a, result(2,1)
result(2,2) = BornOpenData_ShellTest(cmdstr2b)
BOD_PrintToDebug "cmdstr2b", cmdstr2b, result(2,2)

''' PART 3 ''' Test that the user can clone from the GitHub repository '''
If password = "" Then
    cmdstr3a = "git␣clone␣-q␣""file://" + trim(owner) + "//" + trim(experiment) + """␣␣LocalTestRep"
Else
    cmdstr3a = "git␣clone␣-q␣""https://" + trim(User) + ":" + trim(Password) + "@github.com/" + trim(owner) + "/" + trim(experiment) + """␣LocalTestRep"
End If
result(3,1) = BornOpenData_ShellTest(cmdstr3a)
BOD_PrintToDebug "cmdstr3a", cmdstr3a, result(3,1)

''' PART 4 ''' Test that the user can upload a dummy file to the GitHub repository '''
' echo Test of BornOpen on computer xx on date xx > testBornOpenData.txt
id = "Test␣of␣BornOpenData␣for␣" + trim(BOD_eBasename) + "␣on␣" + compname + "␣on␣DATE=" + compday + ":" + comptime + "␣RND=" + CStr(Random(0,1000))
cmdstr4a = "cd␣LocalTestRep␣&␣echo␣" + id + "␣>␣␣testBornOpenData.txt"
result(4,1) = BornOpenData_ShellTest(cmdstr4a)
BOD_PrintToDebug "cmdstr4a", cmdstr4a, result(4,1)

' git add testBornOpenData.txt"
result(4,2) = BornOpenData_ShellTest(cmdstr4b)
BOD_PrintToDebug "cmdstr4b", cmdstr4b, result(4,2)

' git commit --message="Test of BornOpen on computer xx on date xx"
cmdstr4c = "cd␣LocalTestRep␣&␣git␣commit␣--message=""" + id + """␣"
result(4,3) = BornOpenData_ShellTest(cmdstr4c)
BOD_PrintToDebug "cmdstr4c", cmdstr4c, result(4,3)

' git push
result(4,4) = BornOpenData_ShellTest(cmdstr4d)
BOD_PrintToDebug "cmdstr4d", cmdstr4d, result(4,4)

''' PART 5 ''' All tests completed '''
' Print the diagnostic results into the debug window
BOD_PrintDiagnosticToDebug result

msgbox "BornOpenData␣diagnostic␣completed.␣Consult␣the␣Debug␣tab␣of␣the␣Output␣in␣E-Studio.␣Experiment␣aborting..."

' skip upload then end this part of E-Prime
BOD_SkipUpload = 1
End
```

```
' this will nonetheless perform UninitPackages and UninitGlobals

'*****************************************************
' The above code is from the package BornOpenData.epk2
' this is the script section of the sub RunTestsThenAbort
'*****************************************************
```

Listing 11: Code for the ‘ShellRead’ section

```
' Function ShellRead(cmdstr As String) As String

' This function sends a command to the terminal (CMD.exe) then fetch the response

Dim id As Variant
Dim response As String
Dim cmd As String
Dim curfolder As String
Dim i As Integer
Const imax As Integer = 1000
Const tempfile As String = "_!_tempfile_!_.dat"

' get the current folder and remove the file containing the result
id = Shell("cmd␣/c␣cd␣>␣" + tempfile)
sleep 250
Open tempfile For Input As #1
Input #1, curfolder
Close #1
id = Shell("cmd␣/c␣del␣" + tempfile)
sleep 250

' Build the command string
cmd = "cmd␣/c␣" + cmdstr + "␣>␣" + curfolder + "\\" + tempfile

' run the command string and get the result
i = 0
id = Shell(cmd)
While (i < imax) And Not(fileExists(tempfile))
    i = i + 1
    sleep 25
Wend
If i = imax Then
    msgbox "Unexpected␣delay"
    End
End If
sleep 50

Open tempfile For Input As #1
Input #1, response
Close #1

' remove the file containing the result
id = Shell("cmd␣/c␣del␣" + tempfile)
```

```
sleep 250

BornOpenData_ShellRead= response

' End Function
```

Listing 12: Code for the ‘ShellRun’ section

```
' Sub ShellRun(cmdstr As String)

' This function sends a command to the terminal (CMD.exe)

Dim id As Variant
Dim cmd As String

' Build the command string
cmd = "cmd␣/c␣" + cmdstr

' run the command string
id = Shell(cmd)
sleep 500

' End sub
```

Listing 13: Code for the ‘ShellTest’ section

```
'Function BornOpenData_ShellTest(cmdstr As String) As Integer

' This function sends a command to the terminal (CMD.exe) to test that it worked

Dim id As Variant
Dim status As Integer
Dim curfolder As String
Dim cmd As String
Dim i As Integer
Const imax As Integer = 1000
Const tempfile As String = "_!_tempfile_!_.dat"

' get the current folder and remove the file containing the result
id = Shell("cmd␣/c␣cd␣>␣" + tempfile)
sleep 250
Open tempfile For Input As #1
Input #1, curfolder
Close #1
id = Shell("cmd␣/c␣del␣" + tempfile)
sleep 250

' Build the command string
cmd = "cmd␣/c␣" + cmdstr '+ ” 2> nul 1> nul”
cmd = cmd + "␣&&␣" ' if the first command worked
cmd = cmd + "echo␣0␣␣>␣" + curfolder + "\\" + tempfile
cmd = cmd + "␣||␣" ' if the first command did not work
cmd = cmd + "echo␣-1␣>␣" + curfolder + "\\" + tempfile

' run the command string and get the result
i = 0
id = Shell(cmd)
While (i < imax) And Not(fileExists(tempfile))
    i = i + 1
    sleep 25
Wend
If i = imax Then
    msgbox "Unexpected␣delay"
    End
End If
sleep 50

Open tempfile For Input As #1
Input #1, status
Close #1

' remove the file containing the result
id = Shell("cmd␣/c␣del␣" + tempfile)
sleep 250

BornOpenData_ShellTest = status

'End Function
```