The eq-pin2corr Package

D. P. Story
Email: dpstory@uakron.edu

processed June 5, 2021

Contents

1 ⟨∗package⟩

1 Introduction

This package is an add-on to the quiz environment of the exerquiz package. It uses the eq-save package. To correct a quiz, the document consumer must press the Correct button of a quiz and successfully enter the correct PIN number.

Purpose. This package adds PIN security to a quiz created by the quiz environment. This package is designed for the educational sector, for instructors who use the quizzes of exerquiz to assess their students understanding of the course material.

PDF Viewers. Discussion of PDF viewers for document author and document consumer.

Instructor Any PDF viewer may be used as a PDF previewer, sumatraPDF, for instance, can be used, but it has not functionality. To test the newly created document to see if it is functioning correctly, must use Adobe Reader DC or Acrobat DC.[1]

Document consumers (students) The exerquiz and eq-pin2tocorr extensively use JavaScript to perform many background tasks. For the student to have any success in this workflow, he/she must use Adobe Reader.

Workflow. The package is designed for the following workflow:

1. The instructor creates the quiz using the exerquiz and eq-pin2corr packages.
2. The instructor delivers the "PDF quiz" to each student. (System drive or email)
3. The student takes the quiz. The student can press the Correct but, unless he/she knows the PIN, the quiz is not marked up.
4. The student saves the PDF quiz in Adobe Reader DC.
5. The student returns the PDF to the instructor. (System drive or email)
6. The instructor presses the Correct button to mark up the quiz and record the grade of the student. The instructor saves the quiz.
7. The instructor returns the PDF, at some point, to the student.
8. Both instructor and student happily go on with their lives.

2 Options and package requirements

```
\newif\ifPINshowScore \PINshowScorefalse
\DeclareOption{showscore}{\PINshowScoretrue}
\DeclareOption{!showscore}{\PINshowScorefalse}
\ProcessOptions\relax
\RequirePackage{exerquiz}[2021/05/21]
\RequirePackage{eq-save}[2021/04/27]
```

3 Package commands

\showScoreOn
\showScoreOff

Implement local versions of the package options `showscore` and `!showscore`, these are `\showScoreOn` and `\showScoreOff`.

```
\def\showScoreOn{\PINshowScoretrue}
\def\showScoreOff{\PINshowScorefalse}
```

[1] We use DC here to refer to actually any Adobe AA/AR application. Earlier versions of these applications will work.

3.1 PIN security on the Correct control

\SaveAndSendMsg Define a message that appears on the console when the PIN entered is not correct.

```
\flJSStr[noquotes]{\SaveAndSendMsg}{Success! %
Now save and send to the instructor}
```

\postSubmitQuizPIN Begin by modifying the \postSubmitQuiz command, which is a hook within the executing code of the End Quiz control.

```
\begin{defineJS*}[\makeesc\@\makecmt\%]{\postSubmitQuizPIN}
// Begin post submit quiz code%
@ifPINSecurity%
@ifPINshowScore@else
      var f = this.getField("ScoreField.@oField");
      if ( f!=null ) {
        f.textSize=0;
        f.value = "@SaveAndSendMsg";
      } else {
        var f = this.getField("PointsField.@oField");
        if (f!=null) {
          f.textSize=0;
          f.value = "@SaveAndSendMsg";
        }
      }@fi@fi
      oRecordOfQuizData["ScoreData.@oField"]=%
[1*Score,1*NQuestions,1*ptScore,1*NPointTotal];
      oRecordOfQuizData["RightWrong.@oField"]=%
eval(RightWrong.toSource());
      oRecordOfQuizData["ProbDist.@oField"]=%
eval(ProbDist.toSource());
      cntCorrectResponses();
\end{defineJS*}
```

\eQzBtnActns The command name for the action of the End Quiz control is \eQzBtnActns. We save this and pre-pend a single code line, as needed.

```
\let\eQzBtnActnsSave\eQzBtnActns
\def\makeEndQuizPIN{%
  \let\eQzBtnActns\eQzBtnActnsPIN
  \let\postSubmitQuiz\postSubmitQuizPIN
}
\def\eQzBtnActnsPIN{\ifPINshowScore\else
  var bDisplaySilent=true;\r\fi
  \eQzBtnActnsSave
}
%\makeEndQuizPIN
```

The command name for the action of the Correct control is \CorrBtnActionsJS we save this and later modify it.

```
\let\CorrBtnActionsJSSave\CorrBtnActionsJS
```

\usePINCorrBtn \restoreCorrBtn We can turn on and off the PIN feature by expanding \usePINCorrBtn and \restoreCorrBtn.

```
\newif\ifPINSecurity \PINSecurityfalse
\def\usePINCorrBtn{\PINSecuritytrue
  \makeEndQuizPIN % dps5-25
  \let\CorrBtnActionsJS\CorrBtnActionsPwdJS}
\def\restoreCorrBtn{\PINshowScoretrue\PINSecurityfalse
  \restoreEndQuiz % dps5-25
  \let\CorrBtnActionsJS\CorrBtnActionsJSSave}
```

The instructor can tediously press the Correction button, or place an entry,

```
var _PinCode1 = "02JRVZdRgYgCA-Rtje8VkD";
```

in the file `config.js`. If such a variable exists and its value matches the PIN hash string, the instructor clicks the Correction button to get the quiz markup. Conceivably, the instructor might want different PIN names and string hash values. The `\classPINVar` is a convenient way of declaring the PIN variable name; eg, if `\classPINVar{_PinCode1}` is declared prior to the `quiz` environment, the instructor need not manually enter the PIN.

`\classPINVar`

```
\def\classPINVar#1{\def\PINclassPV{#1}}
\let\PINclassPV\@empty
```

`\FreezeThisQuiz` Causes the interactive parts of a quiz to be readonly. This can be passed into the End Quiz control so that when the student presses End Quiz the quiz will be frozen (after a warning); or through the Correct button, then the returned quiz will be readonly.

```
\newif\ifFreezeQuiz\FreezeQuizfalse
\def\FreezeThisQuiz{\FreezeQuiztrue}
\def\FreezeThisQuizNot{\FreezeQuizfalse}
```

`\CorrBtnActionsPwdJS` The modified action for the Correct button. we save this and later modify it.

```
\begin{defineJS*}[\makeesc\|\makecmt\%]{\CorrBtnActionsPwdJS}
|ifx|PINclassPV|@empty%
var userPIN = "";|else%
var userPIN = "|PINclassPV";|fi
if (userPIN == "") userPIN = undefined;
try {
  if ( typeof eval(userPIN) == "undefined") userPIN = undefined;
} catch(e) { userPIN = undefined; }
if (typeof userPIN == "undefined") {
  var resp=app.response({
    cQuestion: "Enter the PIN number",
    cTitle: "View Answers",
    bPassword: true
  });
    var _resp=Collab.hashString(resp);
    if (resp != null) var _bQzResults = ( _resp ==_PinCode );
} else var _bQzResults = ( eval(userPIN) ==_PinCode );
if (_bQzResults) {
```

(2021/05/24) Do nothing if no quiz data

```
    if (typeof oRecordOfQuizData["ScoreData.|currQuiz"]!="undefined") {
```

```
    RightWrong=eval("RightWrong.|currQuiz");
    ProbDist=eval("ProbDist.|currQuiz");
    correctQuiz("|currQuiz",3);
    DisplayQuizResults("|currQuiz",3,3);
    if (typeof correctSumryTbl == "function")
      correctSumryTbl("|currQuiz",3);%
```

(2021/05/22) If `\ifFreezeQuiz` is true, we add some code to freeze all operational components of this quiz.

```
|ifFreezeQuiz
    var aFrzExt=new Array("obj.","grpobj.","essay.",%
"beginQuiz.","endQuiz.");
    for (var i=0; i<aFrzExt.length; i++) {
      var f=this.getField(aFrzExt[i]+"|currQuiz");
      if (f!=null) f.readonly=true;
    }|fi
  } else {
    var retn=app.alert({%
cMsg: "No quiz data, do you want to mark it anyway?",%
cTitle: "AcroTeX", nIcon: 2, nType: 2});
    if (retn==4) {
      var bDisplaySilent=false;
        InitializeQuiz("|currQuiz",1,0);
        var f=this.getField("ScoreData.|currQuiz");
        f.value="0;0;0;0";
        cntCorrectResponses();
      correctQuiz("|currQuiz",3);
      DisplayQuizResults("|currQuiz",3,3,false);
      if (typeof correctSumryTbl == "function")
        correctSumryTbl("|currQuiz",3);
    }
  }
} else {
  if (resp != null) {
    console.clear()
    console.println("Something went wrong, %
you entered an incorrect PIN Id, %
or the class PIN Id (\\\\classPINVar) was incorrect or undefined");
    app.beep(0);
    console.show();
  }
}
\end{defineJS*}
```

3.2 Declare PIN for **Correction** and **Begin Quiz** controls

Two PIN-related commands are defined: (1) `\declPINId` is for security on the Correct control; (2) `\declRePINId` is for security on the Begin Quiz control.

`\declPINId{`⟨*PIN-Id*⟩`}{`⟨*hash-str*⟩`}` Set the basic parameters of this PIN security scheme for

the Correct control: the pin-id and its corresponding hash-string.

```
\def\declPINId#1#2{\def\numPINId{#1}\def\hashPINId{#2}}
\@onlypreamble\declPINId
\declPINId{5243}{02JRVZdRgYgCA-Rtje8VkD} % PIN Id, hash-str
```

`\declRePINId{`⟨*PIN-Id*⟩`}{`⟨*hash-str*⟩`}` Set the basic parameters of this PIN security scheme on the Begin Quiz control: the pin-id and its corresponding hash-string.

```
\def\declRePINId#1#2{\def\numRePINId{#1}\def\hashRePINId{#2}}
\@onlypreamble\declRePINId
\declRePINId{1234}{By9mbLFONJMA2sN2x4DOVB}
\begin{insDLJS}{pin}{Pin Code}
var _PinCode = "\hashPINId";
var _rePinCode = "\hashRePINId";
\end{insDLJS}
```

3.3 Warn and Freeze on End Quiz

`\useWarnEndQuiz` (2021/05/22) Optional modifications to the End Quiz control.

```
\def\useWarnEndQuiz{\let\EndQuizG@te\EndQuizG@tePIN}
\flJSStr{\EndQuizG@te@Msg}{Warning:
Are you sure you want to end this quiz?\r\r
The quiz will be frozen and no more changes will be allowed.
Click 'Yes' to end the quiz or 'No' to continue working on the quiz.}
\def\PINgobii#1#2{}
\begin{defineJS}[\makeesc\|\makecmt\%]{\EndQuizG@tePIN}
// Begin insertion of new code
    var retn=app.alert({cMsg: |EndQuizG@te@Msg,%
cTitle: "AcroTeX", nIcon: 2, nType: 2});
    if (retn==4) {
```

Freeze this quiz; we make a relevant fields readonly.

```
      var aFrzExt=new Array("mc.","obj.","grpobj.",%
"beginQuiz.","endQuiz.");
      for (var i=0; i<aFrzExt.length; i++) {
        var f=this.getField(aFrzExt[i]+"|currQuiz");
        if (f!=null) f.readonly=true;
      }%
|PINgobii
\end{defineJS}
```

The `\PINglobii` vacuums up '␣{', which appears in `\eQzBtnActns` of exerquiz, in the line `*EndQuizG@te␣{`.

3.4 PIN Security on Begin Quiz

`\useBeginQuizPIN` (2021/05/29) This command redefines the Begin Quiz control to require PIN security to retake the quiz.

```
\def\useBeginQuizPIN{%
  \ifx\BeginQuizG@te\BeginQuizG@tePIN\else
```

```
148   \let\BeginQuizG@te\BeginQuizG@tePIN\fi}
```

`\restoreBeginQuiz` is defined in exerquiz, but because of a typo, it was originally defined as `\restorBeginQuiz` (no 'e'); until the changes trickle through the system, we cover ourselves by `\let`ing `\restoreBeginQuiz` to `\restorBeginQuiz`, if the correct spelling of the command does not already exist.

```
149 \@ifundefined{restoreBeginQuiz}{\let\restoreBeginQuiz\restorBeginQuiz}{}
```

```
150 \flJSStr{\BeginQuizG@te@Msgi}{Enter the PIN number
151 to retake this quiz}
152 \flJSStr{\BeginQuizG@te@Msgii}{Press the Begin Quiz control to begin
153 the quiz again}
154 \def\PINgobiii#1#2#3{}
```

```
155 \begin{defineJS}[\makeesc\|\makecmt\%]{\BeginQuizG@tePIN}
156 if (typeof oRecordOfQuizData["ScoreData.|currQuiz"] != "undefined" ) {
157   var resp=app.response({
158     cQuestion: |BeginQuizG@te@Msgi,
159     cTitle: "AcroTeX",
160     bPassword: true
161   });
162   var _respHash=_resp=Collab.hashString(resp);
163   if (_resp == _rePinCode) {
164       oRecordOfQuizData["ScoreData.|currQuiz"]=undefined;
165       app.alert(|BeginQuizG@te@Msgii);
166       var f=this.getField("tallyresets.|currQuiz");
167       if (f!=null) f.value=1+f.value;
168   }
169 } else {
170 |PINgobiii
171 \end{defineJS}
```

`\PINglobiii` gobbles the three tokens '␣{\jsR', which appear in the definition of `\@initQuiz` in exerquiz, in the line `\BeginQuizG@te\space\jsLB\jsR\jsT`.

4 Tracking the number of **Begin Quiz** events

A command `\qzResetTally` is defined to hold the number of times a user has taken the same quiz. This field is readonly, and does not reset, unless you have know how to do it.

`\qzResetTally[`⟨*options*⟩`]` The `\qzResetTally` text field, similar to `\sqTallyTotal`, is used to count the number of times a student needs to retake the same quiz.

The default appearance of `\qzResetTally`

```
172 \def\qzTallyTotalDefaults{%
173   \rawPDF{}\W{1}\BC{0 0 0}\S{I}\textColor{1 0 0 rg}
174   \Q{2}}
```

The definition of `\qzResetTally`

```
175 \newcommand\qzResetTally[1][]{%
```

```
\textField[\presets{\qzTallyTotalDefaults}#1\DV{0}\V{0}
\Ff{\FfReadOnly}
]{tallyresets.\currQuiz}{\TBW}{\DefaultHeightOfWidget}}
```

4.1 Placing a maximum on the number of resets

The next level of complication is to add a restriction onto the number of times the user can retake the quiz.

`\setMaxRetakes{`⟨*qz-name*⟩`}{`⟨*num*⟩`}` A command that sets the maximum number of retakes ⟨*num*⟩ (nonneg integer) by declaring `\setMaxRetakes` prior to the quiz ⟨*qz-name*⟩.

`\nMaxRetakes` `\nMaxRetakes{`⟨*qz-name*⟩`}` is a public macro that expands to the max number of retakes (⟨*num*⟩).

```
\newcommand{\setMaxRetakes}[2]{\expandafter
  \def\csname1MaxReset#1\endcsname{#2}}
\def\nMaxRetakes#1{\@nameuse{1MaxReset#1}}
```

4.2 JavaScript support for tracking

`\useBeginQuizCnt` Modifies `\BeginQuizG@te` (exerquiz, 2021/05/21 or later). `\useBeginQuizPIN` also modifies `\BeginQuizG@te`. These two are incompatible; one will overwrite the other. `\restoreBeginQuiz` retores Begin Quiz to its default definition.

```
\def\useBeginQuizCnt{%
  \ifx\BeginQuizG@te\BeginQuizG@teCNT\else
  \let\BeginQuizG@te\BeginQuizG@teCNT\fi}
```

`\maxRetake@Msg` The message that appears in an alert dialog box after the student has started the quiz more than the max number, as set by `\setMaxRetakes`.

```
\flJSStr[noquotes]{\maxRetake@Msg}{You have taken the quiz the maximum
number of times permitted ("+nMaxReset+"). You will not be allowed to
continue to re-take this quiz. Move on with your life.}
```

The action added to Begin Quiz, it determines if `\qzResetTally` is present for the current quiz and if so, increments the current value.

```
\begin{defineJS}[\makeesc\|\makecmt\%]{\BeginQuizG@teCNT}
if (typeof oRecordOfQuizData["ScoreData.|currQuiz"] != "undefined" )
{
  var _bOKReset=true;
  var f=this.getField("tallyresets.|currQuiz");
  if (f!=null) var value=1*f.value;%
|expandafter|ifx|csname1MaxReset|currQuiz|endcsname|relax|else
  var nMaxReset=|csname1MaxReset|currQuiz|endcsname;
  if (f!=null && value==nMaxReset) {
    app.alert("|maxRetake@Msg");
    _bOKReset=false;
  }|fi
  if (_bOKReset) {
    oRecordOfQuizData["ScoreData.|currQuiz"]=undefined;
```

```
    app.alert(|BeginQuizG@te@Msgii);
    if (f!=null) f.value=1+value;
  }
} else {
|PINgobiii
\end{defineJS}
```

`\PINglobiii` gobbles the three tokens '␣{`\jsR`', which appear in the definition of `\@initQuiz` in exerquiz, in the line `\BeginQuizG@te\space\jsLB\jsR\jsT`.

⟨/package⟩

5 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

6 Change History