AcroTEX Software Development Team

AcroTEX eDucation Bundle
The **eq2db** Package

D. P. Story

Directory

May 9, 2007

http://www.acrotex.net
dpstory@acrotex.net

Table of Contents

1. Introduction

This package is meant to be used with the exerquiz package, one of the components of the **AcroTEX eDucation Bundle**. This package is used to customize exerquiz to save results of the `quiz` environment to a database.

The eq2db Package has five options:

1. `eqRecord`: Save quiz data to a database, such as Microsoft Access.
2. `eqEmail`: Email quiz data to the instructor.
3. `eqText`: Save quiz data to a tab-delimited text file. Can then be opened and analyzed using Microsoft Excel.
4. `custom`: A hook for developers to use eq2db with their own script.
5. `tagged`: Write quiz data in an XML-like data stream.

The package itself redefines the 'End Quiz' button appropriately so that when the user clicks on it, the results will be sent to server-side script. The package also provides some commands for easily creating hidden text fields that are used to send information to the server-side script.

Accompanying each of the first three options listed above is server-side script. These are ASP pages, Microsoft IIS is required.

2. What's New

In version 1.3, I've added the `eqEmail` and `eqText` server options and added a `tagged` option that can be sued in combination with any of the server options.

3. The Distribution and Requirements

The eq2db Package is distributed with the following files:

1. `eq2dbman.pdf`: This document, the documentation for the LaTeX package eq2db and its related files.
2. `eq2db.dtx` and `eq2db.ins`: The LaTeX package with its installation file.
3. eqRecord.asp: An ASP script for saving exerquiz quizzes to a database.
4. eqEmail.asp: An ASP script for forwarding exerquiz quizzes via e-mail.
5. eqText.asp: An ASP script for saving exerquiz quizzes to a tab-delimited file.
6. `quiz1.tex`, `quiz2.tex`, `quiz3.tex`, `quiz4.tex` (quizzes 1 and 2 demo the `eqRecord` option; `quiz3` demos `eqEmail`, and `quiz4`

demos `eqText`.)

7. `eqQuiz.mdb`: A Microsoft Access 2000 database that is used with the demo files `quiz1.tex`, `quiz2.tex`.

The eq2db Package requires the exerquiz package to create online quizzes and to create supporting buttons and text fields. The exerquiz package, in turn, assumes a number of packages; a listing of these can be found in the documentation for the AcroTEX eDucation Bundle.

4. The `eqRecord` option

In this section we describe how to create an online quiz that is to be submitted to the server-side script eqRecord.asp[1]. For this option, quiz results are saved to a database.

▶ `quiz1.tex` and `quiz2.tex` are the demo files for this option.

The steps to create a quiz to be submitted to eqRecord are simple enough. First, the preamble of your document should look something

[1] The script eqRecord.asp comes with absolutely *no guarantees*. Extensive testing should be made on your own system to assure yourself script is reliable enough to use in practice. Feel free to modify the script to suite your needs. If your improvements are noteworthy, please allow me to incorporate them into the basic eqRecord.asp for others to use.

like this:

```
\documentclass{article}
\usepackage[designi]{web}
\usepackage{exerquiz}
\usepackage[eqRecord]{eq2db}
```

You may have other packages loaded, as well as other options for web and exerquiz. The eq2db package was designed to be seamless in the following sense: If the eq2db package is not loaded, then an exerquiz quiz is self-contained, that is, it is not submitted to a server-side script; if eq2db package is loaded, then the quizzes are submitted to a server-side script.

Next, you write your exerquiz quiz:

```
\eqSubmit{http://localhost/scripts/eqRecord.asp\#FDF}{QuizIt}{Math101}
\begin{quiz}*{Quiz1} Answer each of the following. Passing is 100\%.
\begin{questions}
\item ...
...
...
\item
\end{questions}
\end{quiz}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

Preceding the quiz is the \eqSubmit command, defined in the exerquiz.

```
\eqSubmit{<URL to script>}{<dbName>}{<dbTable>}
```

The first parameter is the URL to the server-side script, in the example above, we have `http://localhost/scripts/eqRecord.asp#FDF` (note the suffix of `#FDF`); the second parameter is the database, my test examples use ODBC, with `QuizIt` as the DSN (data source name); the third parameter is the name of the table into which the data record is to be inserted (this is `Math101` in my example).

There is another parameter of importance. The first argument of the `quiz` environment is the `quizName` of the quiz, this is `Quiz1` in the above example.

Assuming eqRecord.asp is installed on your web server, and the *FDFToolkit* has also bee installed (see Section 4.4), we are ready to submit the quiz.

4.1. Field Values Processed by eqRecord.asp

The eqRecord script processes two classifications of field data: certain "hard-wired" field data; and field data having a hierarchal name with a root of either `<dbName>` or `IdInfo`, e.g., `Quiz1.numQuestions` or `IdInfo.Name.Last`. Details of each of these follow.

- **"Hard-wired" Fields**

When you compile your source document using the package eq2db with the eqRecord option, the package creates three hidden text fields, with field titles of dbName, dbTable and quizName. These hidden fields are created under the 'End Quiz' button. At submit time, these fields are populated and submitted. Below is an enumerated list of these three fields with a brief description of each:

1. dbName: The name of the database to which the data is to be saved. (I've been using ODBC to reference the database.)
2. dbTable: The name of the table within dbName where the data is to be stored.
3. quizName: The name of the quiz or test. Name would uniquely characterize the quiz/test the student has taken.

Exerquiz has the \eqSubmit command to acquire two of the three above mentioned items. \eqSubmit takes three parameters:

```
\eqSubmit{<URL to script>}{<dbName>}{<dbTable>}
```

For example,

```
\eqSubmit{http://localhost/scripts/eqRecord.asp\#FDF}{QuizIt}{Math101}
```

The first is the URL to the server-side script, the second parameter is dbName (QuizIt, in this case), and dbTable (Math101). The three

arguments of \eqSubmit are used to define the values of \db@Name, \db@Table and \eq@CGI, respectively, the values of which are used to populate the hidden fields.

The third piece of data, quizName, is obtained from the first required argument of the quiz environment. The following is a skeleton example of a quiz:

```
\eqSubmit{http://localhost/scripts/eqRecord.asp\#FDF}{QuizIt}{Math101}
\begin{quiz}*{Quiz1}
  \begin{questions}
      \item ...
      \item ...
      . . .
      . . .
      \item ...
  \end{questions}
\end{quiz}
```

This argument is assigned as the value of the text macro \curr@quiz, also seen below.

- **Fields with Hierarchal Names**

Other than the fields described in "Hard-wired" Fields, eqRecord.asp processes only fields with hierarchal names that have a root name of <quizName> (see "Hard-wired" Fields) or a root name of IdInfo.

quizName: quizName is the field title of one of the hidden fields, its value is picked up as the first argument of the quiz environment; the value is stored in the text macro \curr@quiz.

In addition to the "hard-wired" hidden fields described earlier, there are actually three more hidden fields (under the 'End Quiz' button) with root name <quizName>. These are

- \curr@quiz.numQuestions: The number of questions in the quiz
- \curr@quiz.numCorrect: The number of correct questions
- \curr@quiz.Responses: a list of all the responses of the user

The values of these fields are also sent to eqRecord.asp. There is a mechanism for creating more hidden fields the values of which are sent to the script. The technique for doing this will be discussed later.

IdInfo: As mentioned earlier, eqRecord.asp processes fields that use a hierarchal naming convention, with root name of <quizName> or IdInfo. Fields whose root name is IdInfo are meant to hold information about the person taking the quiz: first name, last name, student number, etc.

For example, you can create fields the user fills in for self-identifi-

cation. In the preamble, you can define new commands:

```
% User's First Name
\newcommand\FirstName[2]{\textField{IdInfo.Name.First}{#1}{#2}}

% User's Last Name
\newcommand\LastName[2]{\textField{IdInfo.Name.Last}{#1}{#2}}

% User's SSN
\newcommand\SSN[2]{\textField[\MaxLen{11}
    \AA{\AAKeystroke{AFSpecial_Keystroke(3);}
        \AAFormat{AFSpecial_Format(3);}
    }]{IdInfo.SSN}{#1}{#2}}
```

Note that I've defined these fields so that their names follow a hierarchal name structure, with a root of `IdInfo`. The two required arguments are the dimensions of the text field being constructed. For example `\FirstName{100pt}{10pt}` creates a text field with a title of `IdInfo.Name.First` which is `100bp` wide and `10bp` high.

▶ See the demo file `quiz1.tex` for examples.

Additional `IdInfo` fields can be constructed.

4.2. Mapping PDF Field Names onto DB Field Names

eqRecord.asp is a quasi-general script for mapping the values of PDF fields into corresponding fields in a database. The way eqRecord is

set up, the DB field name is derived from the PDF field name. For example, the value of the PDF field `Quiz1.numQuestions` is stored in the DB under the DB field name of `numQuestions`.

For field names such as `IdInfo.Name.Last`, we can't map this into the DB field name `Name.Last` because for some databases (notably, Microsoft Access) database field names containing a 'dot' are not legal. As a work around, eqRecord.asp strips all dots from the field name, and replaces them with the value of the VB Script variable `dotReplace`. The definition of `dotReplace` in eqRecord.asp is

```
Dim dotReplace : dotReplace = "_"
```

That is, a 'dot'(`.`) is replaced by an 'underscore' (_).

The table below illustrates the mapping of PDF field name onto database field names. Suppose the `quizName` is `Quiz1`:

PDF Field Name	**DB Field Name**	**Required**
`quizName (e.g. Quiz1)`	`quizName`	Yes
`Quiz1.numQuestions`	`numQuestions`	Yes
`Quiz1.numCorrect`	`numCorrect`	Yes
`Quiz1.Responses`	`Responses`	Yes
`IdInfo.Name.Last`	`Name_Last`	No

PDF Field Name	DB Field Name	Required
IdInfo.Name.First	Name_First	No
IdInfo.SSN	SSN	No
N/A	TimeOfQuiz	Yes

The last entry needs comment: eqRecord.asp generates a time stamp when a quiz is processed. This time stamp is stored in a field with a name of TimeOfQuiz. This is a required field in your database.

4.3. Adding more Hidden Fields

When you add fields with a hierarchal name with a root of <quizName> or IdInfo—whether hidden or not—the values of these fields will be submitted and processed by eqRecord.asp. When you add fields in your document, there should be a corresponding DB field to receive this data (see Section 4.2 for naming conventions). You can also add hidden fields to transmit information about the quiz that the user does not need to see.

Example. Suppose you want to a point value to each question, and report the point score (rather than the number missed). In this case, you would use the \PointsField field instead of the \ScoreField (though you could use both).

The field \PointsField has a hierarchal name, but its root does not begin with <quizName>; consequently, the value of this field is not submitted to eqRecord.asp. (The value of this field is a string what is meant to be read by the user; it reads, for example, "Score: 16 out of 20"; we don't want this string submitted anyway, we would want the point score (20) and the total points (20) submitted.) We want to transmit the points scored and the total number of points to the server-side script.

The eq2db package defines two helper commands for creating hidden fields (these hidden fields are hidden under the 'End Quiz' button); these are \addHiddenField and \populateHiddenField.

Suppose we had a quiz, Quiz1, in which we wanted to report points scored and total points. First, we need to add two hidden fields, we'll call them Quiz1.ptScore and Quiz1.nPointTotal; we create the hidden fields using \addHiddenField:

```
\addHiddenTextField{Quiz1.ptScore}{}
\addHiddenTextField{Quiz1.nPointTotal}{\theeqpointvalue}
```

The first parameter is the title (name) of the field, the second parameter is the default value. For the first line, no default value is given; for the second line, a default value of \theeqpointvalue is given. The counter \eqpointvalue contains the total number of points for the

quiz so we can insert that value at latex compile-time.

The value of the first field added, Quiz1.ptScore, is not known until a user takes a quiz and submits results. The eq2db command \populateHiddenField inserts the necessary JavaScript that would populate the specified field. For example,

```
\populateHiddenField{Quiz1.ptScore}{ptScore}
```

The first parameter is the field name, the second is the value the field is to hold when the user clicks on 'End Quiz'. The command \populateHiddenField{fieldname}{fieldvalue} expands to

```
this.getField("fieldname").value = fieldvalue;
```

which is the JavaScript for populating the field, is inserted into the code just prior to the submission of the data.

▶ See second quiz in the sample file quiz1.tex for a complete example of creating a quiz with weight set for each question, and for reporting the point score and total points. ◀

The table below lists some variables (both LaTeX and JavaScript) that might be useful in extracting desired information from quiz for submittal.

Variable	Description
`Score`	Score of the user with one point per problem. This value is always reported in the hidden text field `quizName.numCorrect`. This is a JavaScript variable known when the user finishes the quiz. This value is always reported in `quizName.numQuestions`, the hidden text field.
`\thequestionno`	The number of questions in the current quiz. A LaTeX macro; known at compile-time.
`Responses`	A comma-delimited list of all responses of the user. A JavaScript variable known when the user finishes the quiz. Always reported in the hidden field `quizName.Responses`
`ptScore`	The point score of the user's quiz. A JavaScript variable, known when the user finishes quiz.

Variable	Description
\theeqpointvalue	The total number of points in the current quiz. A LaTeX macro, known at compile-time.
pcScore	The score expressed as a percent. A JavaScript variable known when the user finishes the quiz.
QuizGrade	The letter grade of the user's quiz. A JavaScript variable, known at "run-time".
\eqGradeScale	The grade scale for the quiz. A LaTeX macro, known at "compile-time".

▶ Other useful variables—perhaps at developers request—will be added in the future. ◀

4.4. Setting Up

In this section, we briefly discuss configuring your server and setting up the demo files that accompany this distribution.

- **Configuring the Server**

On the server side, in order for eqRecord.asp to run correctly, Microsoft Internet Information Server (IIS), version 4.0 or greater, is needed. The script eqRecord.asp needs to be placed where ASP scripts have execute permissions.

The eqRecord.asp uses the *Acrobat FDF Toolkit*[2], version 6.0. Follow the directions for installation contained in the accompanying documentation.

Install eqRecord.asp in a folder (perhaps called `Scripts`) designated to execute scripts. If you don't have such a folder, then the following steps explain how to create a virtual directory through IIS that points to this folder.

1. Create a new folder on the system (`Scripts`, for example). Its recommended location is inside the `Inetpub` folder.
2. Place eqRecord.asp in this newly created folder.
3. In the MMC snap-in for IIS, create a virtual directory by right-clicking on the Default Web Site and selecting `New > Virtual Directory`.

[2]Currently located at the Acrobat Family Developer Center.

4. Type "Scripts" (or whatever the name of the folder you created in Step 1) as the alias for the virtual directory, and then link it to the physical directory you created in Step 1.
5. Make sure that "Script execution" privileges are enabled. If not, enable them.

- **Set Up the Demo Files**

Place the Access 2000 database, `QuizIt.mdb`, in a folder that is accessible by the server, such as the root level of your web server, or perhaps, a folder dedicated to database files. Register the database with the ODBC Data Source Administrator as a System DSN under the System Data Source Name of "QuizIt".

Modify the first parameter of `\eqSubmit` in `quiz1.tex`, compile (`latex`. Create `quiz1.pdf` using Acrobat Distiller, pdftex or dvipdfm, and place it on your server. Test by opening `quiz1.pdf` in your web browser and taking a sample test. Good luck, I hope it works!

- **Comments on Demo Files**

`quiz1.tex`: This file has two quizzes, `Quiz1` and `Quiz2`. `Quiz1` has no frills, the user enters his/her name and SSN, takes the quiz, and results are saved to the `eqQuiz.mdb` Access database. `Quiz2` is a

bit more interesting as it demonstrates how to assign points to each question, and how to report the results. The techniques used in this quiz were discussed in Section 4.3.

`quiz2.tex`: The quizzes of `quiz1.tex` did not check whether the user is enrolled in the class, and as such, is allowed to take the quiz. The demo file `quiz2.tex` demonstrates how to...

1. check whether the user has entered a valid SSN, that is, the SSN of someone enrolled in the class, or is permitted to take the quiz
2. set a deadline date to take the quiz
3. set a time limit to take the quiz

Other variations are possible.

4.5. Accessing Results in the DB

The database that is to hold quiz results is placed on a web-server and consequently, is not easily accessible by the instructor. The ASP script GenericDB (http://www.genericdb.com/), developed by Eli Robillard, is a tool that can be used to access the database through your web browser. Using GenericDB, you can view, edit, update, add new and delete database records—and it's free! Check it out!

5. The `eqEmail` Option

In this section we describe how to create an online quiz that is to be submitted to the server-side script eqEmail.asp[3]. For this option, quiz results are e-mailed to the instructor.

- `quiz3.tex` is the demo file for this option.

The steps to create a quiz to be submitted to eqEmail are simple enough. First, the preamble of your document should look something like this:

```
\documentclass{article}
\usepackage[designi]{web}
\usepackage{exerquiz}
\usepackage[eqEmail]{eq2db}
```

You may have other packages loaded, as well as other options for web and exerquiz. The eq2db package was designed to be seamless in the following sense: If the eq2db package is not loaded, then an exerquiz `quiz` is self-contained, that is, it is not submitted to a server-side

[3]The script eqEmail.asp comes with absolutely *no guarantees*. Extensive testing should be made on your own system to assure yourself script is reliable enough to use in practice. Feel free to modify the script to suite your needs. If your improvements are noteworthy, please allow me to incorporate them into the basic eqEmail.asp for others to use.

script; if eq2db package is loaded, then the quizzes are submitted to a server-side script.

Next, you write your exerquiz quiz:

```
\eqSubmit{http://localhost/scripts/eqEmail.asp\#FDF}
    {dpstory@uakron.edu}{Math101}
\begin{quiz}*{Quiz1} Answer each of the following. Passing is 100\%.
\begin{questions}
\item ...
...
...
\item
\end{questions}
\end{quiz}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

Preceding the quiz is the \eqSubmit command, defined in the exerquiz.

```
\eqSubmit{<URL to script>}{<instr@someedu.edu>}{<courseName>}
```

The first parameter is the URL to the server-side script, in the example above, we have http://localhost/scripts/eqEmail.asp#FDF (note the suffix of #FDF); the second parameter is the e-mail address of the person to receive the quiz results; the third parameter is the course name. In the above example, the course name is Math101.

▶ If instr@someedu.edu is a comma delimited list of e-mail addresses, then eqEmail.asp will parse the list, and use the first e-mail in the list in the From field of the e-mail. ◀

There is another parameter of importance. The first argument of the `quiz` environment is the `quizName` of the quiz, this is `Quiz1` in the above example.

Assuming eqEmail.asp is installed on your web server, and the *FDFToolkit* has also bee installed (see Section 4.4), we are ready to submit the quiz. Below is the body of one of my test e-mails I made using `quiz3.pdf`:

```
Course Information
    Course Name: Math101
    Quiz: Quiz1
    TimeOfQuiz: 8/23/2003 8:12:40 PM

Student Results
    Name_First: Don
    Name_Last: Story
    SSN: 121212121
    Responses: Leibniz,2x e^(x^2),-1,-cos(x)
    numCorrect: 4
    numQuestions: 4
```

If you add more fields that processed, as described in Section 4.3, this information should be appended to the body of the e-mail.

5.1. Field Values processed by eqEmail.asp

Actually, eqEmail.asp is a variation of eqRecord.asp and many of the details of this section are the same Section 4.1.

The "hard-wired" fields are `mailTo`, `courseName` and `quizName`. These fields are hidden under the "End Quiz" button, and are populated at submit time. As in the `eqRecord` option, the value of the `quizName` field comes from the argument of the `quiz` environment:

```
\eqSubmit{http://localhost/scripts/eqEmail.asp\#FDF}
    {dpstory@uakron.edu}{Math101}
\begin{quiz}*{Quiz1} Answer each of the following. Passing is 100\%.
\begin{questions}
\item ...
...
...
\item
\end{questions}
\end{quiz}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

In this quiz, the field with name `quizName` will have a value of `Quiz1`.

See paragraph "Fields with Hierarchal Names", as well as Section 4.3 on "Adding more Hidden Fields" for the `eqRecord` option, the details are the same.

5.2. Setting up and Modifying the Script

On the server side, in order for eqEmail.asp to run correctly, Microsoft Internet Information Server (IIS), version 4.0 or greater, is needed. The script eqEmail.asp should be placed where ASP scripts have execute permissions. There are two methods of sending e-mail:

1. CDONTS: This method (which is commented out by default) can be used on an NT server. Uncomment if you want to use CDONTS, and comment out the CDOSYS code lines that follow.
2. CDOSYS: This can be run on a Win2000 or WinXP server.

The script needs to be modified appropriate to your server, in particular, search down in eqEmail.asp for the configuration line

```
eqMail.Configuration.Fields.Item
    ("http://schemas.microsoft.com/cdo/configuration/smtpserver")
        = "mySMTP"
```

replace mySMTP with your SMTP server.

The subject heading of the returning e-mail has the following format: Quiz Results: Quiz1 of Math101. If you want another subject format, you are free to modify eqMail.Subject as desired.

- **References**

The following links were used as a reference in the development of the `Email.asp` script.

- CDOSYS:
  - Invision Portal Tutorial: CDOSYS email tutorial
  - MSDN: CDO for Windows 2000. The IMessage Interface. (Use MIE to view this page.)
  - ASP 101 Sending Email Via an External SMTP Server Using CDO
- CDONTS
  - Juicy Studio The ASP CDONTS Component
  - DevASP Sending Mail from ASP with CDONTS.NewMail Object

6. The `eqText` Option

In this section we describe how to create an online quiz that is to be submitted to the server-side script eqText.asp[4]. With this option, you

[4]The script eqText.asp comes with absolutely *no guarantees*. Extensive testing should be made on your own system to assure yourself script is reliable enough to use in practice. Feel free to modify the script to suite your needs. If your improvements are noteworthy, please allow me to incorporate them into the basic eqText.asp for others to use.

save your quiz data to a tab-delimited file.

- The demo file for this option is `quiz4.tex`.

Begin the document with the usual preamble:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[dvipsone,designi]{web} % dvips, pdftex, dvipsone, dvipdfm
\usepackage{exerquiz}
\usepackage[eqText]{eq2db}
```

Write you quiz, and include an `\eqSubmit` command. The first argument of `\eqSubmit` is the URL of the server-side script, the second is the path to the text file on the server the data should be written to, the third is the class name.

```
\eqSubmit{http://localhost/scripts/eqText.asp\#FDF}
    {c:/Inetpub/Data/math101.txt}{Math101}

\begin{quiz}*{Quiz1} Answer each of the following. Passing
is 100\%.

\begin{questions}

\item ...
...
...
```

```
\item ...
\end{questions}
\end{quiz}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

6.1. Setting up the Script

On the server side, in order for eqText.asp to run correctly, Microsoft Internet Information Server (IIS), version 4.0 or greater, is needed. The script eqText.asp should be placed where ASP scripts have execute permissions.

7. The tagged Option

You can take the `tagged` option in combination with any of the other three "server" options: `eqRecord`, `eqEmail`, or `eqText`. When you take this option, the exerquiz data is written as a "XML-like" string. Here's an example:

```
<results id="Quiz1" file="quiz1.pdf" n=4>
    <question n=1 type="text" points=3 correct=0>
        <value>D. P. Story</value>
    </question>
    <question n=2 type="math" points=4 correct=0>
        <value>2x</value>
    </question>
    <question n=3 type="math" points=3 correct=1>
```

```
        <value>-1</value>
    </question>
    <question n=4 type="math" points=5 correct=1>
        <value>-cos(x)</value>
    </question>
</results>
```

This string packs more information in it than the default string. If you inspect this string, you'll see the name of the quiz is `"Quiz1"`, from the file `"quiz1.pdf"` and there are four questions. The first question was a text fill-in type, worth 3 points, the student missed that one when s/he entered an answer of D. P. Story. The second question was a math fill-in worth 4 points, this person got this one wrong as well when s/he enter `2x` as the answer. And so on.

8. The `custom` Option

This option allows a script developer to utilize the macros of the eq2db Package to prepare an AcroTEX document to submit to a custom script. Simple create a file named `eq2dbcus.def` and include any custom creation of fields you may need for your script. When you then take the `custom` option,

```
\documentclass{article}
\usepackage[designi]{web}
```

```
\usepackage{exerquiz}
\usepackage[custom]{eq2db}
```

the command definitions of eq2db are read, followed by the inputting of eq2dbcus.def.

► That's all for now! Hope you find the package useful. Now I really must get back to work. 𝔇𝔓