Introduction

The Acrobat Database Connectivity (ADBC) plug-in provides some basic JavaScript properties and methods for connecting to a database. These can be used to obtain information about the databases available on the system, the tables contained within each database and the data types used within any given table. More importantly, JavaScript can be used to execute SQL commands and retrieve data, which can, in turn, be used to populate a PDF file. Conversely, through JavaScript and SQL, field values of a PDF form can be saved to a database, data can be added or updated as needed.

All of this is possible without the use of a web server or CGI script. These queries and manipulations can take place on a desktop PC/workstation. When combined with the powerful batch processing capability of Acrobat 5.0, large scale database tasks involving PDF files can be performed in a automated manner.

Currently, ADBC is a Windows only feature and requires Open Database Connectivity (ODBC) provided by Microsoft.

The purpose of this article is to describe the basic steps for getting started; this article is not a tutorial in SQL. A good online reference to SQL is the article Introduction to Structured Query Language by James Hoffman.

Fundamentally, this paper covers the following topics.

- The ADBC Demo Files: A brief discussion of the ADBC demo files in this folder.
- Registering an ODBC Data Source
- Connect and Execute SQL
  - Connect to the Data Source
  - Executing a SQL statement
  - Retrieving Data and Populating a Form
  - Updating the Database

The ADBC Demo Files

Also contained in this folder are two PDF files *ADBCdemo.pdf* and *FormsBatch.pdf*, and one small Microsoft Access database *ADBCdemo.mdb*.

- **ADBCdemo.mdb:** Both PDF demo files use this database. Before using the demos, you must transfer *ADBCdemo.mdb* to your hard drive, then register it with the *ODBC Data Source Administrator*. Instructions for registering the database are contained in the *ADBCdemo.pdf* file or in the section [Registering an ODBC Data Source](#), below.
- **ADBCdemo.pdf:** This is a two page file. On the first page, it is demonstrated how to extract information from ODBC and list rows of data from your databases. This page also illustrates some of the new forms features, the dynamic creation (and removal) of form fields through JavaScript, for example. Complete instructions are contained on the page. The second page illustrates how to connect directly to a particular database (*ADBCdemo.mdb*). On this page, you can connect, page through the database, and even update the entries.
- **FormsBatch.pdf:** This demo illustrates how to populate a form (*FormsBatch.pdf)* and save the populated form to the hard drive all in batch mode. The batch sequence used is called "Populate and Save", located in the *Batch* folder of the Acrobat CD. Additional information on "Populate and Save" can be found in the file *BatchSequences.pdf*; you should read the discussion on "Populate and Save" before trying to use the *FormsBatch.pdf* demo file.

Registering an ODBC Data Source

Your database must be registered with ODBC as a data source in order to be accessible by ADBC. The following steps describes how to register a Microsoft Access file as a database. The steps may be slightly different for other databases.

- In the Windows control panel, click on the ODBC icon to gain access to the "ODBC Data Source Administrator".
- Click on either the "User DSN" or the "System DSN" tab.
    - A "User DSN" is only visible and available to the user currently logged on to the current computer.
    - A "System DSN" is available to any user logged on to the workstation.
- At this point you are in the "Create New Data Source" dialog box. Double click on the ODBC driver for the data source (database) you want to access. For example, you might click on "Microsoft Access Driver (*.mdb)".
- ODBC driver Setup Dialog
    - In the "Data Source Name" box, type in a name for the data source. This string does not have to be the actual name of your database or table.
    - In the "Description" box, enter some description of the data source.
    - In the "Database" box, click on the "Select..." button, and browse to find the database (or data source).
- After finding your data source, click on "OK", and back out of the "ODBC Data Source Administrator".

Having finished the above steps, the database can now be accessed through the ADBC interface.

Connect and Execute SQL

The premise of this section is that you have a data source registered with ODBC by the name of "my Data" and contained within this data source is a table called "my Table".

Refer to the *Acrobat JavaScript Object Specification* manual for details of the ADBC objects, properties and methods discussed below.

Connect to the Data Source

First, connect to your data source to obtain a *Connect Object,* which, in turn can be used to obtain a *Statement Object*:

```
try {
        // try to connect to the data source "my Data"
        connect = ADBC.newConnection("my Data");
        if (connect == null) throw "Could not connect";

        // get a statement object
        statement = connect.newStatement();
        if (statement == null) throw "Could not execute newStatement";
} catch(e) {
        app.alert(e);
}
```

This code assumes a fixed data source, "my Data". Alternatively, the script can get the data source's name from a text field of the PDF form you are trying to process; see, for example the file ADBCdemo.pdf on the Acrobat 5.0 CD.

Data sources, such as a Microsoft Access file, that have been password protected can be accessed by either providing the password at connect time, through a popup dialog that appears on the screen, or, by providing the password programmatically

```
connect = ADBC.newConnection("my Data", "Admin", "dps017");
```

Alternatively, back in the "ODBC Data Source Administrator", you can configure your Access file so that the password is supplied automatically. In the "ODBC Microsoft Access Setup" dialog, click on the "Advanced" button, then fill in the "Login name" and "Password" in the "Default Authorization" box.

Executing a SQL statement

An SQL command is executed through ADBC by using the *execute* method of the *statement object* just acquired:

```
connect = ADBC.newConnection("my Data");
statement = connect.newStatement();
```

```
        statement.execute(SQL)              // where SQL is some valid SQL statement
```

To retrieve data from the database, you must execute the SQL statement SELECT/FROM.

```
        SELECT <list of columns> FROM <table>
```

For example,

```
        SELECT "FirstName", "LastName" FROM "my Table"
```

To select all the columns from a table, you the following:

```
        SELECT * FROM <table>
```

Note, the above syntax is not presented in the most general form.

```
        try {
                connect = ADBC.newConnection("my Data");
                if (connect == null) throw "Could not connect";
                statement = connect.newStatement();
                if (statement == null) throw "Could not execute newStatement";

                // statement.execute returns true if successful
                // Note the use of the single quotes and double quotes here,
                // double quotes are needed if the name has white space in it.
                if(statement.execute( 'Select * from "my Table"' ))
                    throw "Could not execute the requested SQL";
        } catch(e) {
                app.alert(e);
        }
```

Retrieving Data and Populating a Form

Assuming now we are connected to our target data source and have executed a SELECT/FROM command to choose the data we wish to examine, we now want to retrieve a row of the data meeting the selection criteria and populate a PDF form.

The following script might be the mouse up action of a button (visible, but doesn't print) on the form.

```
        try {
            statement.nextRow();
            var row = statement.getRow(); // get a row object
            this.getField("id").value = row.ID.value;
            this.getField("firstname").value = row.FirstName.value;
            this.getField("lastname").value = row.LastName.value;
            ...... more fill in statements ..........
```

```
        this.getField("telephone").value = row.Telephone.value;
        this.getField("income").value = row.Income.value;
} catch(e) {
        app.alert("No more data.");
```

If the column names contain white space, using the syntax *row.FirstName.value* will not work, instead use a bracket notation:

```
try {
        statement.nextRow();
        var row = statement.getRow(); // get a row object
        this.getField("id").value = row["ID"].value;
        this.getField("firstname").value = row["First Name"].value;
        this.getField("lastname").value = row["Last Name"].value;
        ...... more fill in statements ..........
        this.getField("telephone").value = row["Telephone"].value;
        this.getField("income").value = row["Income"].value;
} catch(e) {
        app.alert("No more data.");
```

See the *Statement Object* in the *Acrobat JavaScript Object Specification* for more details.

Updating the Database

Given there is data in a PDF form, perhaps an FDF that has arrived by way of e-mail, the data can be stored in a database through an INSERT, for adding a new row of data, or by an UPDATE, for updating an existing row.

Example: The following script was taken from the "Update" button on the second page of the *ADBCdemo.pdf* file on the Acrobat CD

```
try {
    // construct the update string---update all fields
    var updateStr = "UPDATE ClientData"
      +" SET FirstName='"+this.getField("firstname").value+"',"
      +" LastName='"+this.getField("lastname").value+"',"
      +" Telephone='"+this.getField("telephone").value+"',"
      +" Income="+this.getField("income").value
      +" WHERE ID="+this.getField("id").value;

    // for debug, see what we have just constructed
    // console.println(updateStr);

    // and execute the update
    statement.execute(updateStr);
} catch(e) {}
```

Final Comments

This short article just touches only on the major steps needed for connecting to a database and for retrieving information. Study the demo files in the database folder of the Acrobat CD for additional examples, techniques and ideas.

Acrobat and ADBC now provides powerful tools for solving local database problems and it is hoped that developers will take these new tools and produce for the business community some useful and worthwhile applications.