File Clerk v0.3 Developer's Handbook

An Open Source Document Management System

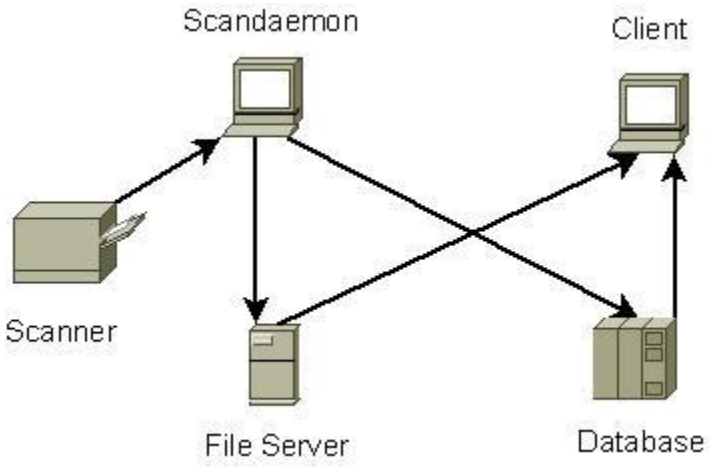
Purpose and Structure

File Clerk is a paperless document management system written by Eric Scott (http://www.SigmaX.org) for the Andrews University College of Arts and Sciences (AUCAS) Dean's Office from 2006-2010, and is released under the BSD license. It has the simple goal of organizing scanned copies of forms and documents in an electronic format and making them accessible to multiple users over the network. While this would seem like a fairly ubiquitous task in IT, commercial solutions to small office document management are limited (ergo the motivation for this project).

It is not an electronic forms or enterprise resource planning system, though depending on the office it may take on elements of both. It is not intended for more than a handful of users at a time, since it currently does not utilize transactions in its database layer – though it doubtless will be adapted in the future to serve more general (and high volume) purposes.

This document explains the inner workings of File Clerk in detail, and is intended for programmers who want to improve or modify the system. For user documentation, see the **User's Manual** or **Installation Manual**.

When a file is scanned, the document itself is stored in the file system (often on a server via a network drive), *and* it is indexed in the database along with miscellaneous user-defined *metadata* (such as keywords, name, department, etc). The files can then be viewed on a client's computer, and searches can be executed on the metadata. Drop-down menus and pre-specified document types are used to establish strict protocols for the metadata – keywords and such must be consistent in order to be searchable.



Code Layout

File Clerk is written in C#, and was developed in Microsoft Visual C# Express 2005. It is made up of the following namespaces:

- **AUCAS_Core**: A DLL containing most of application's functionality. Database methods, configuration forms, error handling, and anything that might be reused between projects is here.
- **AUCAS_Client3**: The primary client interface for searching the database, editing metadata, and viewing documents.
- **AUCAS_ScanDaemon**: A secondary interface for scanning and uploading new documents.
- **AUCAS_OCRDaemon**: Legacy. Intended to queue and process newly scanned files for optical character recognition.
- **SchemaConverter**: Used to convert data from the original (pre-0.3) database schema into the new, more dynamic layout.

Supported File Types

The system currently has full or limited support for the following file types. The code is written with new formats in mind, and can be easily extended (See the **FDocument** class).

- **MDI** (Microsoft Document Imaging): MDI is an extremely proprietary format, and the library is incapable of exporting to formats that are viewable by other applications (It's TIFFs are not even viewable in the Windows Picture and Fax Viewer). It was a mistake to use and we are abandoning it – support is included only for viewing files that have not yet been converted to a new format.
- **TIFF**: Multi-page TIFF documents are currently the primary format for storing and viewing documents.
- **PDF**: Files can currently be exported to PDF. Ideally, the entire system should support PDF in addition to TIFF, but I can't seem to find a free PDF viewer control.

Dependencies

File Clerk utilizes the following third-party libraries:

- **MODI** (Microsoft Office Document Imaging): Legacy/Optional. A library in Microsoft Office 2003 that provides a fixed document format, ActiveX control for viewing them, and an OCR (Optical Character Recognition) engine.
- **sharpPDF** (http://sharppdf.sourceforge.net): A PDF-creation library under the GNU LGPL.

Portability

File Clerk was written for Microsoft Windows XP. The core can hypothetically be ported to other platforms via third-party .NET framework implementations such as Mono (http://mono-project.com), but parts of **AUCAS_Client3** and **AUCAS_ScanDaemon** currently depend on P/Invoke calls to Win32 APIs.

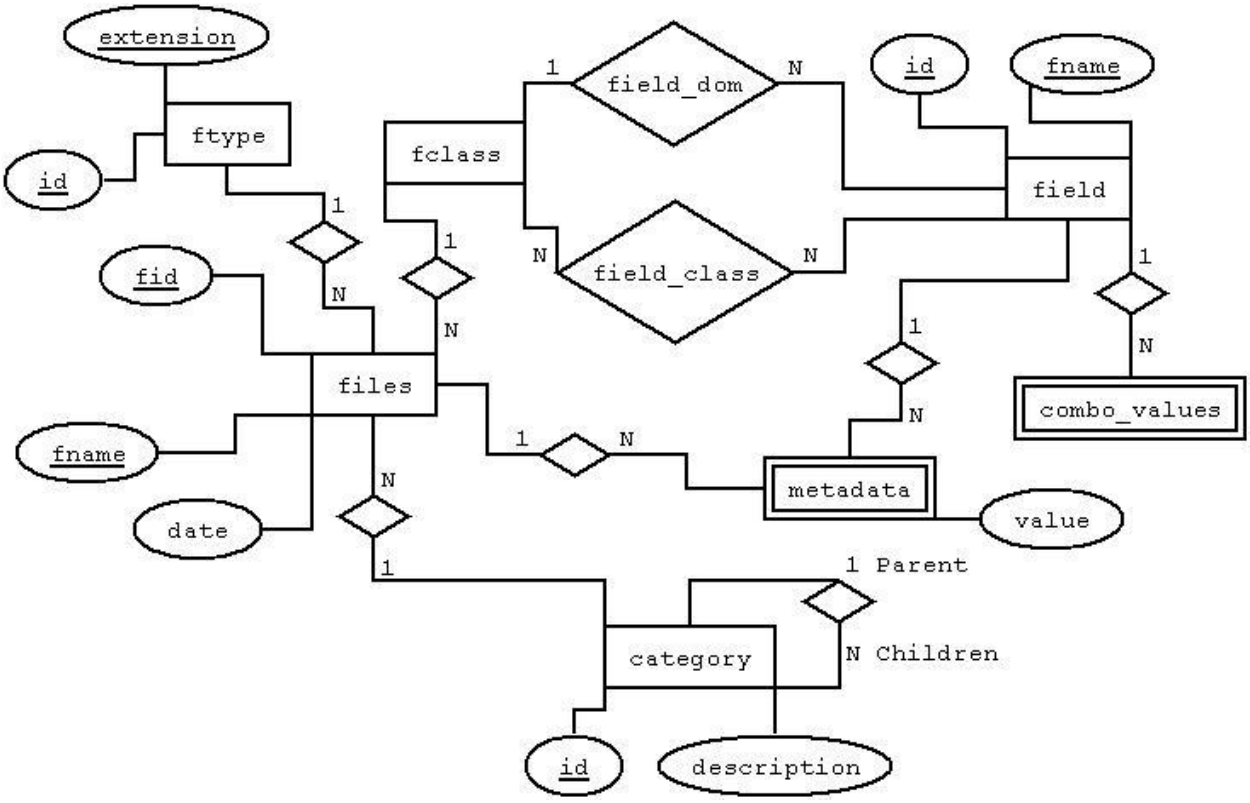
Critique

The use of a Windows Forms application for the database is acceptable for users who will be dealing with the system on a regular basis. What if, however, a department using File Clerk wants to make a collection of forms or documents readily available to a wider audience – such as the organization as a whole, or to users on the Internet? The current approach is useless.

A wonderful solution would be to write a replacement for **AUCAS_Client3** that uses an ASP .NET interface to make the system available via the web. All the core tools would then be available via **AUCAS_Core**, where most of the code for the project lies in a ready-to-use format (Except the parts of **AUCAS_Core.File_Index** etc which are currently designed to populate Windows Forms objects).

Database Schema

File Clerk's database layout is complex. The heart of the system is the **files** entity, which contains an entry for every document which has been indexed. Files are organized by **category**, and each category corresponds to a directory in the file system.



The purpose of each of these tables is as follows:

- **category:** Organizes the files according to the directory hierarchy. Categories contain both files and sub-categories. Each category can have up to one parent category.
- **combo_values:** Stores the allowed values for fields which are marked as drop-down menus.

- **fclass:** Indexes document classes. A document class is a predefined set of allowed fields. A given file belongs to one document class.
- **field:** Index of metadata fields (such as name, keyword, department, etc).
- **field_class:** The N to M relation between fields and classes. A field can belong to multiple classes, which in turn contain multiple fields.
- **field_dom:** 1 to 1 relation between fields and classes. This signifies which field is "dominant" in a class, meaning the field that represents the document's title to the user.
- **files:** File entries are stored with their name and date, along with various foreign keys to other tables.
- **ftype:** Supported file extensions.
- **metadata:** The data stored in the fields. This is usually the largest table. Note that the data must be linked to the document via **fclass** in order to be displayed. See **AUCAS_Core.Field_Interface** for how this is handled.

Attributes

category		
id	INTEGER	Primary Key
description	VARCHAR(128)	Category/Directory name
parent	INTEGER	Recursive Foreign Key, parent category

combo_values		
id	INTEGER	Primary Key
value	VARCHAR(255)	An allowed value for this ComboBox object
field	INTEGER	Foreign Key to field

fclass		
id	INTEGER	Primary Key
description	VARCHAR(128)	Class name

field		
id	INTEGER	Primary Key
fname	VARCHAR(64)	Field name
is_combo	INTEGER	Whether or not this field is a drop-down menu. 1 or 0.

field_class		
id	INTEGER	Primary Key

class	INTEGER	Foreign Key to `fclass`
field	INTEGER	Foreign Key to `field`

field_dom – *(field and class form a multicolumn primary key)*		
class	INTEGER	Foreign Key to `fclass`
field	INTEGER	Foreign Key to `field`

files		
id	INTEGER	Primary Key
ftype	INTEGER	Foreign Key to `ftype`
fclass	INTEGER	Foreign Key to `fclass`
fname	VARCHAR(64)	Name of file (without extension)
category	INTEGER	Foreign Key to `category`
date	DATETIME	Date file was added

ftype		
id	INTEGER	Primary Key
extension	VARCHAR(4)	File extension

metadata – *(field and parent form a multicolumn primary key)*		
field	INTEGER	Foreign Key to `field`
value	VARCHAR(255)	Data for this field
parent	INTEGER	Foreign Key to `files`

Constraints

The system was originally written for *MySQL*'s *MyISAM* database engine, which is very limited in terms of transaction and constraint support. This was a bad idea, but we're stuck with it for now. Ergo constraints are implemented on the application level, and some care must be taken by the programmer to ensure that they are properly enforced.

For example, the method **AUCAS_Core.Field_Interface.delete_field**() removes a row from the **field** entity. This would orphan countless entries in **combo_values**, **field_class**, and **metadata**, so the method must also check these tables for foreign key references to the deleted field's ID and remove them as well.

The trickiest constraints involve the relationship between **field_class** and **metadata**. See **AUCAS_Core.Field_Interface**.

Portability

File Clerk uses .NET's ODBC abstraction layer, but was written for and tested with *MySQL*. Some queries use elements of the *MySQL* dialect, so modifications will have to be made to the codebase before it will operate with other database platforms such as *Microsoft SQL Server* or *PostgreSQL*.

Critique

In a world of unlimited man hours, the following changes would be made to the database layer:

- Transactions, or at *least* table locking, are a must in a multi-user environment.
- Constraints should be controlled by the DBMS, not the code. The less bookkeeping we do on our end, the less chance for errors.
- Full ODBC portability is not possible for the above two points – not all DBMS's support it. But the system should be set up to work on more than MySQL. It'd be great to see SQLite, MS SQL Server, Postgres, and CSV available.
- The database interfaces contain a lot of redundant code. A light abstraction layer would make things much cleaner.
- Parameterized queries overused.

AUCAS_Core Namespace

AUCAS_Core is the backbone of the system, providing an abstract framework for the file and database operations the user may need. It compiles to a DLL that should be placed in the same directory as the GUI executables. It uses two XML files, one for storing configuration options (See **genConfig**), and another that indexes error messages (See **ErrorDesc**).