Ingesting and retrieving documents in Adlib with a repository with sub-Folders

Introduction

Adlib can have application fields that point to document files. These fields are underlined in the Adlib User Interface and once a user clicks on them the appropriate viewer for the file type (e.g. Adobe Reader for .pdf, Word for .doc and .docx files, Excel for .xls or a web browser for .html files) is launched and the user can view the contents of the file.

Under standard conditions all the files are held in a single folder. This is not so bad for small collections of files or when using a fast file system (NTFS), but if the collection becomes larger or when the files reside on a share with a not so fast file system (e.g. Samba on a Linux machine) then performance might become an issue.

Under these circumstances it would be better to subdivide the repository in several sub folders, each holding a maximum number of files. In this article we use the example of 500 files per sub folder, but the solution works for any arbitrary number.

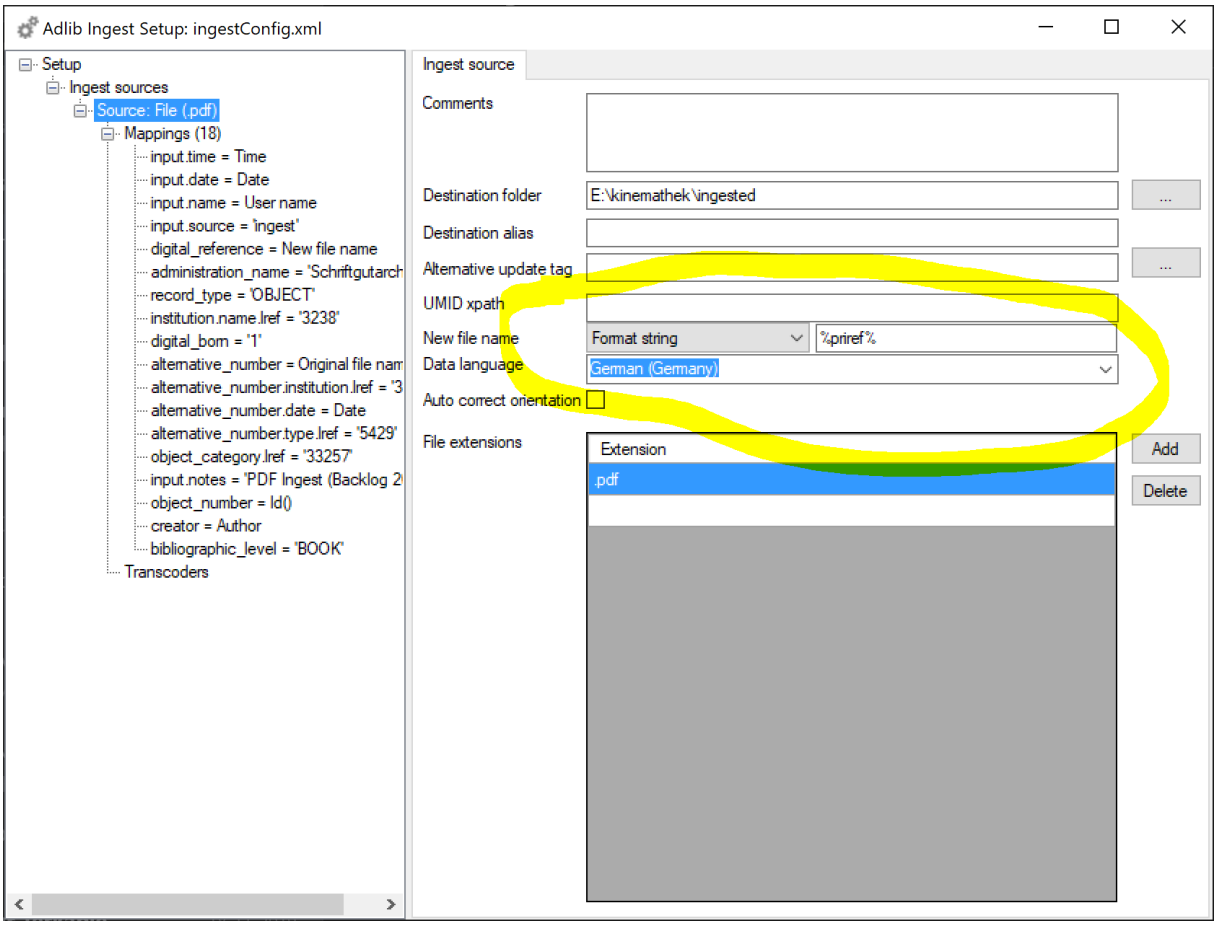
The solution consists of several components:

The Axiell Ingest module

This module reads the files to ingested from a *hot* folder. For each file type there is a specific mapping to fields in the Adlib database. In this article I assume that you already have set up this, if not read the manual the Ingest module.

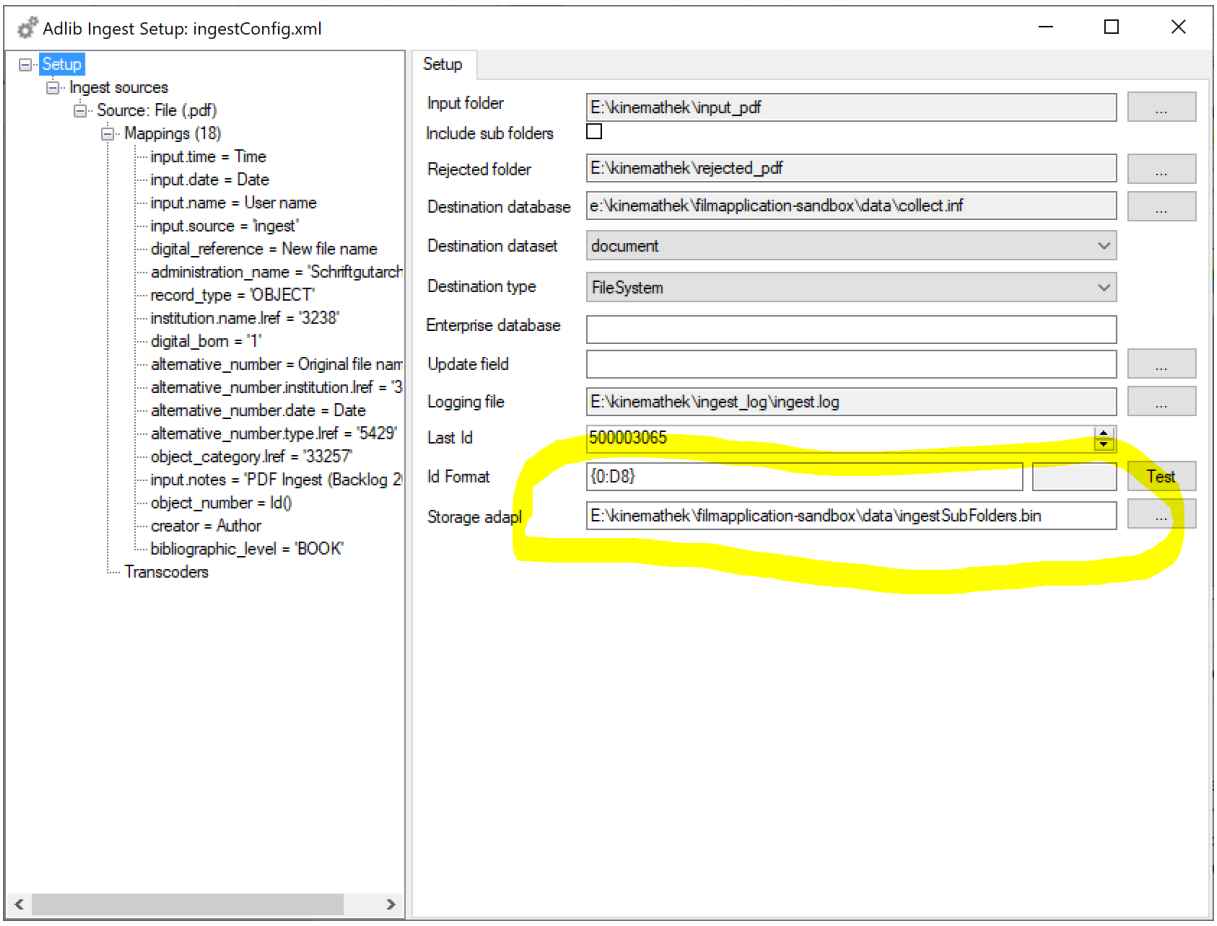
What we need is two things: the generated file names need to be sequential and numeric and we need an adapl to modify the content of the reference to the document to contain a sub folder number.

Creating sequential unique numbers for the files is easy: in the latest version of the ingest module we can use the record number of the created metadata record as the unique file name. To enable this, we need to use the following setting in the ingest source:



The ingested files will then be renamed in correspondence with the generated meta data records, e.g. 200000197.pdf for the metadata record with priref 200000197. Note that you can still retain the old file name too. In the above mapping this is in the line with field ‘alternative_number’

But instead of writing just the newly generated file in the application field we need to precede it by its folder number. To automate this, we need a small adapl. I have named this adapl ingestSubfolders.ada. The adapl can be placed anywhere and needs to be compiled and linked to the ingest module. You link it in the setup screen of the ingest module:



Note that the new adapl 'replaces' the original storage adapl, so if there is any specific processing that you want to retain from the 'real' storage adapl then this code has to be replicated in the ingest storage adapl. In our example this was not needed

The adapl code is not overly complicated:

```
integer folder

folder = int(val(before$(1, RF, '.'))) / 500   /* 500 files per sub folder
if (folder <> 0)
{
  RF = (folder - 399999) + '/' + RF            /* subtract 399.999 to set the first folder to 1.
}
```

The first line declares an integer to hold the folder number. Then the folder number is calculated by taking everything in front of the '.', converting it to an integer value and dividing the outcome by 500 (the number of files that we want in the sub folders). For the prirefs in our example with a dataset range of 200.000.000 to 399.999.999 this results in a number between 400.000 and 599.999. To scale this down we subtract 399.999 so the first folder maps to 1. Note the folder <> 0 check… this is needed because the adapl is actually executed twice. In order to get the record number of the newly created metadata record an empty record is written first and then the record is updated with the file name. If we did not have the folder <> 0 check then an extra 0 would be prepended in the new value in RF

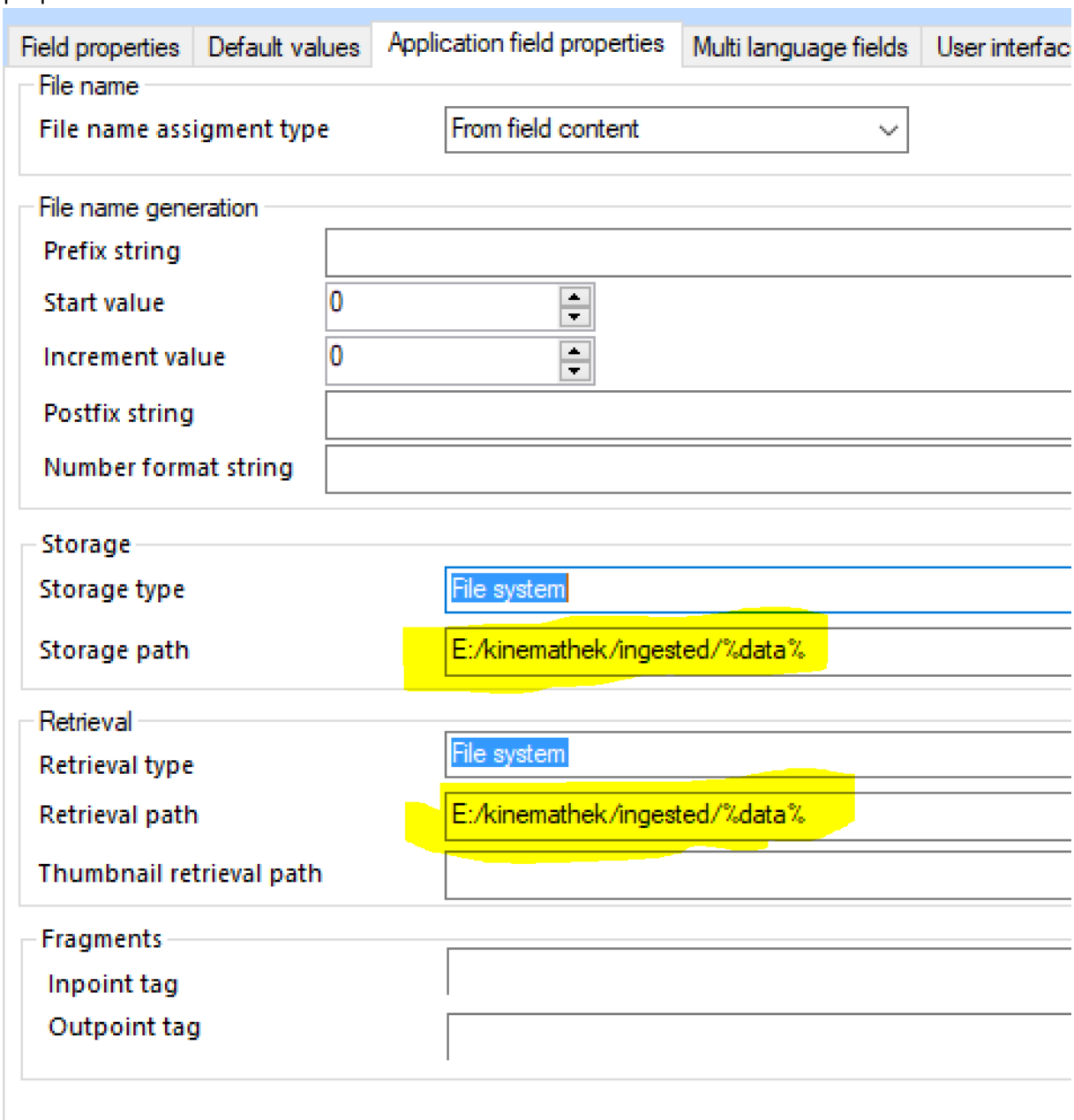
The outcome for our 200000197.pdf will be '1/200000197.pdf', meaning it will be placed in folder 1. This value is now written in the Adlib record, like this example:

link	1/200000205.pdf

This is all good, but not enough to allow the user to actually see the file when clicking on it. To get to this goal we have two more components.

The setup in ADLIB for windows

We saw in the Ingest setup that the files are copied to ‘e:\kinemathek\ingested’. In the setup of our application field RF in the database we need to define this path as well. To do this we change the properties of the RF fields as follows:



This means that instead of %data% our newly generated ‘path’ will be placed, like this:
E:/Kinemathek/ingested/1/200000197.pdf
For some reason it is necessary that the path is defined as Storage AND Retrieval path, even though

we do not deal with the storage here. Not setting the storage path has as result that the file cannot be retrieved.

Moving the file to the right sub folder

As you might have guessed all ingested files now end up in the master folder, but still have to be moved to the appropriate sub folder.
For this purpose, I have written a small command line program that checks the ingested folder. Once it finds files there it takes the file name, extracts the priref and makes the same calculation as the above described adapl. The number of files per sub folder and the fileOffset are optional command line arguments, but default to 500 and 399.999, the same as the adapl.
The first command line argument is mandatory and points to the Adlib Ingest folder (e:\kinemathek\ingested in pour example.

The program runs in an infinite loop and suspends itself after checking for files for a period of 10 seconds. The program can be run on a server or on a workstation, it must have write access in the ingest folder.

Usage: IngestFileMover parentFolder [filesPerFolder] [fileOffset]

The source code can be found on

https://github.com/bertdd/moveIngestedFile

Summary of this article:

To have your files in multiple subdirectories in a document repository you need

- Setup the ingest with prirefs as field names and compile and add a storage adapl.
- Setup the retrieval path for the application field with the right **storage AND retrieval** paths
- Run the moveingestedfile program to get the files into the right subfolder.\

Version 1. February 2020, Bert Degenhart Drenth