**Deduping Record Numbers in 001 Fields – Part 2**

***Overview***: In this cleanup task, you will find any cases where you have multiple bib records with the same record number in the 001 field.

***Importance***: The presence of multiple records with the same value in the 001 will most likely negatively impact the data migration process and the follow-on linking of records to the master WorldCat records in the Network Zone.

**Step #1 – Create a File of All 001 Field Values**

Using the largest review file you have, create a file of bib records starting at the first record in your catalog based on Millennium .b record numbers. Use the following search BIBLIOGRAPHIC MARC Tag 001 not equal to "" (001 not equal to nothing/empty box) to exclude records that do not contain an 001 field.

Export the 001 fields into a text file (“.txt” file extension). Use Test qualifier <none> and Repeated field delimiter semicolon for export parameters.

Repeat until you have a set of files with the 001 field values from all bib records in your catalog.

**Step #2 – Find and Fix Records with Multiple 001 Fields**

Using any standard word processor (such as WordPad or NotePad), open the files of 001 field values created in step up #1 above.

Find any lines in the file that contain a semicolon. This indicates a record with two 001 fields. If you completed the cleanup tasks described in “Deduping Record Numbers in 001 Fields – Part 1,” these pairs of 001 fields will always be non-identical.

Open the record in Millennium (using one of the 001 values in the .txt file), determine which 001 value is correct, and remove the incorrect value from both the record in Millennium and from the .txt file.

Where you’re done with this process, no record in Millennium should have more than one 001 field.

**Step #3 – Find 001 Values Present in Multiple Records**

Merge the files created in the steps above so that you have a single .txt file containing the 001 values from all bib records in your catalog. There are two methods for finding 001values that are present in multiple records.

The first method is to create a script that will run through the file of 001 values and find those values that are present more than once. We used a PERL script to do this. Our script is shown at the bottom of this document so you can use it or improve on it.

The second method is even easier. If you send your .txt file of 001 values to Lesley Lowery ([Lesley.Lowery@wwu.edu](mailto:Lesley.Lowery@wwu.edu)), she will run the field through her script and send back to you a single Excel spreadsheet listing the 001 values that are present multiple times in your file.

**Step #4 – Cleanup Records Multiple Records that Contain the Same Value in the 001 Field**

Now you have to do some trouble shooting. For each situation where you have multiple records with the same value in the 001 field, you’ll have to figure out what is appropriate. In some situations you may be able to move item records from one bib record to another and simply delete one of the item records. In other situation, you may need to overlay one record with a better bib record or even contribute a new record to WorldCat and overlay an old record with the new original record.

Whatever you need to do, the ultimate goal is to no longer have the same 001 value present in multiple bib records.

*Created 2/11/2013 by Bob Thomas*

***Sample PERL Script for finding values that occur multiple times in a file***

#!/usr/bin/perl -w

use strict;

=begin comment:

This program ingests a tab-delimited .txt file containing a single column of key values and outputs a tab-delimited .txt file containing one entry for each duplicated key value.

=end comment

=cut

################################################################################

########################### Declarations #####################################

my $Filename;

my $Discard;

my $First\_number;

my $Second\_number;

my @Controls;

my %Control\_Number;

my $Key;

my $Value;

################################################################################

############################ Main Program ####################################

#Open filehandles for reading and writing.

print "From which file would you like to read?\n"; #Prompt user for filename.

chomp ($Filename = <STDIN>);

open READING, "<C:/Documents and settings/loweryl/My Documents/$Filename"

or die "Cannot open file :$!";

open WRITING, ">C:/Documents and settings/loweryl/My Documents/Duplicates.txt"

or die "Cannot open file :$!";

#Assign each line of input file to an array element.

my @List\_Elements = <READING>;

#Increment through lines:

foreach my $Record (@List\_Elements) {

$Record =~ s/\s//; #Strip all whitespace.

@Controls = split /\;/ , $Record ; #Split each line into separate control numbers.

if ($Controls[1]) { #If the array has a second element,

$First\_number = shift @Controls;

$Second\_number = shift @Controls;

if ($First\_number == $Second\_number) { #and if first and second control numbers match,

unshift (@Controls, $Second\_number); #put only one number back into the array.

}

else { #Otherwise,

unshift (@Controls, $Second\_number); #put both numbers back in.

unshift (@Controls, $First\_number);

}

}

foreach my $Number(@Controls) { #Take each separate number

if ($Control\_Number{$Number}) { #If it's already a key in the hash,

$Control\_Number{$Number} += 1; #Increment it's value by one.

}

else { #Otherwise,

$Control\_Number{$Number} = 1; #Establish the new key with a value of one.

}

}

}

while (($Key,$Value) = each %Control\_Number) {

if ($Value>1) {

print WRITING "$Key\n"; #Print a list of all keys with value >1.

}

}