Detecting Academic Dishonesty in Office 2007 Assignments

Don Gable, PhD
Computer Information Systems Department
University of Dubuque
Dubuque, IA 52001
DGable@dbq.edu

Abstract

This paper describes the design and execution of software developed at our institution that is capable of deconstructing Office 2007 Word documents in such a way that inappropriate collaboration can be detected.

Using the various Rsid codes (Revision Save Identifier codes) that are imbedded in these documents, the instructor can determine if two or more students are submitting images of the same work or descendent images of other students' work.

Since all assignments for the affected courses are submitted online using Moodle, the examination of work can be performed across multiple sections and even multiple semesters: If a student submits work completed by a student in a prior semester, the software will catch it.

Future effort on this project will include automatic grading of Word, Excel, and PowerPoint assignments.

1 Introduction

Classes that teach the use of Microsoft Office applications are faced with at least two problems: One, since a large volume of assignments are produced, there is a large grading load; two, since the goal of many assignments is to produce work that looks like the examples in the text, how can the instructor be sure that students are turning in their own work and not the identical work produced by some other student? This paper addresses this second point.

2 Detecting Academic Dishonesty

With earlier versions of Microsoft Office such as Office 2003, the Creation date imbedded in the application file was a trustworthy method for detecting academic dishonest events where a student would turn in an assignment that was produced by a student in an earlier school year. Beyond that, however, there were few tools at the instructor's disposal to identify work that was not original for each student.

Since the introduction of Office 2007, the creation date that is imbedded in the document file is of little value. When a Save-As operation is performed, the creation date is updated to the current date. The only visual way to check student work is to move the insertion point into each paragraph and check the ribbon to see what features are being used. For example, we had a situation where 4 students turned in an assignment where the 'Quote' quick style was used although this style was never addressed in any assignment and was never mentioned in class. Still, an exhaustive comparison of works using this approach is prohibitively time consuming.

A better way to detect academic dishonesty is needed. The Open Office XML document format used by default in Office 2007 provides this better way.

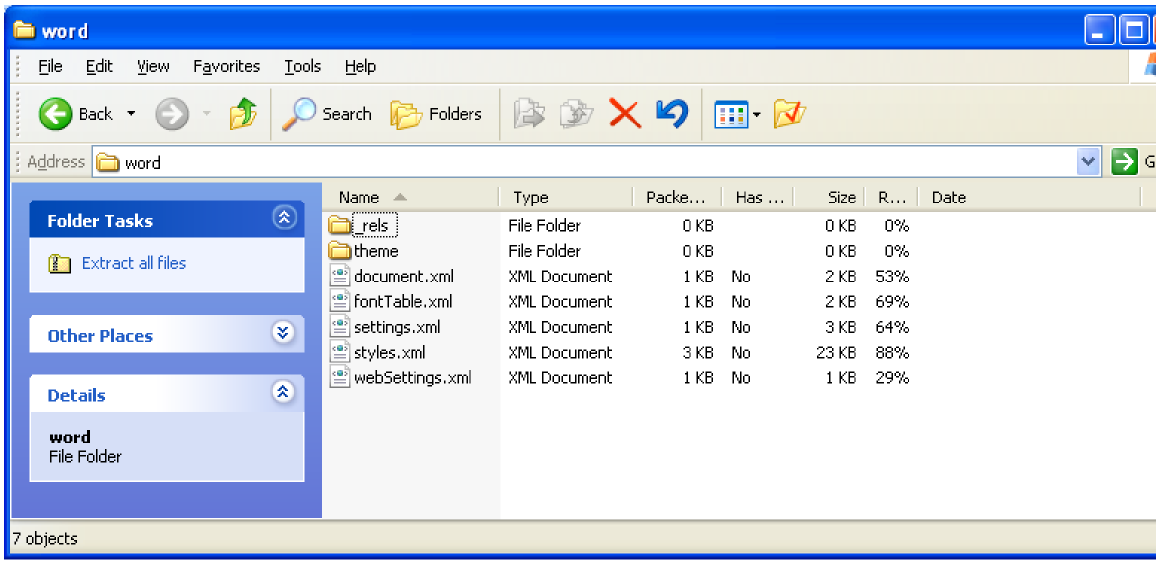
3 The Open Office XML (OOX) Document Format

The files produced by Office 2007—files with .docx, .xlsx, and .pptx file extensions—are stored as ZIP compressed 'packages' that are comprised of a number of XML files. Because XML files are pure text, significant size reduction is realized when the files are compressed.

A feature of the OOX format that is particularly useful for the task at hand is the embedding of unique codes into the document for use in merging documents that have a common document ancestor. These codes, called Rsid (Revision Save Identification) codes, are unique 32-bit values represented in text as hexadecimal values.

3.1 Office 2007 Approach

Microsoft Office 2007 uses a revised default format when storing Word, Excel, and PowerPoint documents. The information is stored, to the extent possible, as text encoded into XML document. Each top level document, called a 'package', is comprised of a number of these XML documents as shown in Figure 1. The set of these XML documents is then compressed into a ZIP file resulting in a smaller size when stored.



Name	Type	Packe...	Has ...	Size	R...	Date
_rels	File Folder	0 KB		0 KB	0%	
theme	File Folder	0 KB		0 KB	0%	
document.xml	XML Document	1 KB	No	2 KB	53%	
fontTable.xml	XML Document	1 KB	No	2 KB	69%	
settings.xml	XML Document	1 KB	No	3 KB	64%	
styles.xml	XML Document	3 KB	No	23 KB	88%	
webSettings.xml	XML Document	1 KB	No	1 KB	29%	

Internal File Structure of a Microsoft Word Document

Figure 1

By causing a document to be stored in a text format, Microsoft permits other applications to access and manipulate the contents of the package.

3.1.1 Document Representation

The principal document content is found in the /word/document.xml file in the package. A typical document is shown in Figures 2-1 and 2-2.

This is a new Word document typed by a single user. There are no edits to in at the time of initial creation. We will want to have a look at the RSID codes that show up in this original document. Just for fun, we will add this frivolous sentence that must be removed on a later revision. there is only this paragraph. Once <enter> is pressed, there will be 2 paragraphs with the last one empty. ¶

¶

Typical Word Document

Figure 2-1

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
w:document xmlns:ve="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:o="urn:schemas-
  microsoft-com:office:office" xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
  xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:v="urn:schemas-microsoft-com:vml"
  xmlns:wp="http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing" xmlns:w10="urn:schemas-
  microsoft-com:office:word" xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main"
  xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml">
w:body>
w:p w:rsidR="0071385E" w:rsidRPr="00E40B19" w:rsidRDefault="00E40B19">
w:r w:rsidRPr="00E40B19">
w:rPr>
w:rStyle w:val="tx1" />
</w:rPr>
w:t xml:space="preserve">This is a new Word document typed by a single user. There are no edits to in at the time of
  initial creation. We will want to have a look at the RSID codes that show up in this original document. Just for fun, we
  will add this frivolous sentence that must be removed on a later revision.</w:t>
</w:r>
w:proofErr w:type="gramStart" />
w:r w:rsidRPr="00E40B19">
w:rPr>
w:rStyle w:val="tx1" />
</w:rPr>
w:t>there</w:t>
</w:r>
w:proofErr w:type="gramEnd" />
w:r w:rsidRPr="00E40B19">
w:rPr>
w:rStyle w:val="tx1" />
</w:rPr>
w:t xml:space="preserve">is only this paragraph. Once <enter> is pressed, there will be 2 paragraphs with the last one
  empty.</w:t>
</w:r>
</w:p>
w:sectPr w:rsidR="0071385E" w:rsidRPr="00E40B19" w:rsidSect="006C16DB">
w:pgSz w:w="12240" w:h="15840" />
w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440" w:header="720" w:footer="720" w:gutter="0" />
w:cols w:space="720" />
w:docGrid w:linePitch="360" />
</w:sectPr>
</w:body>
</w:document>
```

XML Representation of a Word Document

Figure 2-2

There are two primary entities in the word document: Paragraphs and runs. A paragraph begins with the start of the document and ends when the user presses <enter> and inserts a ‘hard return’ denoted by the ‘¶’ formatting mark. A run consists of the actual text of the document, often broken into separate sections depending on editing steps or spelling and grammar errors that may be found. Typically, there will be multiple paragraphs per document and multiple runs per paragraph. The text in the actual document is found only in a run.

There are other sections supporting these primary entities. These include ‘run properties’ and ‘paragraph properties’. These sections can hold various attributes that determine the appearance of the document: <b /> for bold, <i /> for italics, for example.

The internal representation of the document is comprised of a hierarchy of primary XML tags:

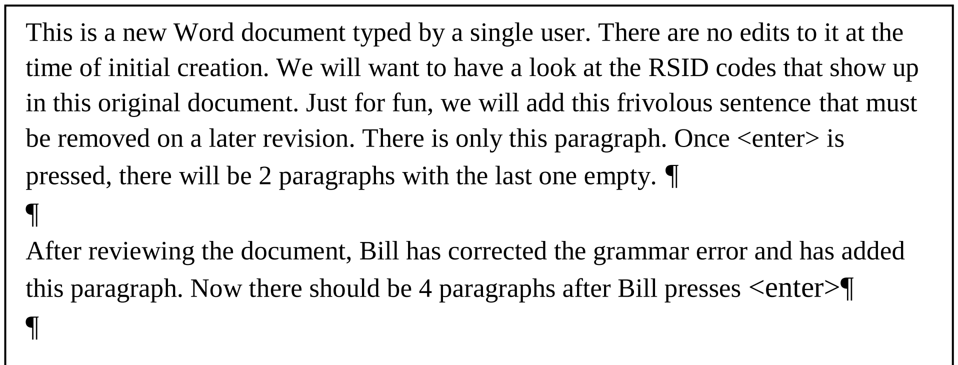
```
<document>
        <body>
                <paragraph>
                        <run>
                                <text>Actual document text</text>
                        </run>
                </paragraph>
        </body>
</document>
```

Considering the documents above in Figure 2, it will be observed that the grammatical error (non-capitalized ‘there’) is flagged in a special way in the XML document representation. The run with the single word ‘there’ is bracketed by <proofErr> tags. When the image of the document is rendered by Word the word(s) in the run will be displayed with the typical wavy green line indicating a grammatical error.

3.1.2 Tracking Revisions

Each paragraph and each run are prefaced with Rsid Revision-Save Identifier codes. During a single editing session, the same codes will be used to track changes throughout the document. In subsequent editing sessions, new codes will be written to the package. A new session is started each time the document is opened and edited or saved.

As is shown in Figure 3, when a copy of the document is modified by another user, new codes are also stored in the package. It is because of these codes that Word, in its Combine function (Review Tab/Compare Group/Compare Command/Combine Choice) can take these multiple descendent copies of a document and combine them back into a single document.



This is a new Word document typed by a single user. There are no edits to it at the time of initial creation. We will want to have a look at the RSID codes that show up in this original document. Just for fun, we will add this frivolous sentence that must be removed on a later revision. There is only this paragraph. Once <enter> is pressed, there will be 2 paragraphs with the last one empty. ¶

¶

After reviewing the document, Bill has corrected the grammar error and has added this paragraph. Now there should be 4 paragraphs after Bill presses <enter>¶

¶

Modified Word Document

Figure 3-1

```
   <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

- <w:document xmlns:ve="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:o="urn:schemas-
     microsoft-com:office:office"
     xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
     xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:v="urn:schemas-microsoft-
     com:vml" xmlns:wp="http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing"
     xmlns:w10="urn:schemas-microsoft-com:office:word"
     xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main"
     xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml">
- <w:body>
- <w:p w:rsidR="0071385E" w:rsidRDefault="00A05879">
- <w:r>
  <w:t xml:space="preserve">This is a new Word document typed by a single user. There are no edits to it at the
     time of initial creation. We will want to have a look at the RSID codes that show up in this original document.
     Just for fun, we will add this frivolous sentence that must be removed on a later revision.</w:t>
   </w:r>
- <w:r w:rsidR="000540A8">
  <w:t>There</w:t>
   </w:r>
- <w:r>
  <w:t xml:space="preserve">is only this paragraph. Once <enter> is pressed, there will be 2 paragraphs with the
     last one empty.</w:t>
   </w:r>
   </w:p>
  <w:p w:rsidR="00A05879" w:rsidRDefault="00A05879" />
- <w:p w:rsidR="000540A8" w:rsidRDefault="000540A8">
- <w:r>
  <w:t>After reviewing the document, Bill has corrected the grammar error and has added this paragraph. Now
     there should be 4 paragraphs after Bill presses <enter></w:t>
   </w:r>
   </w:p>
  <w:p w:rsidR="000540A8" w:rsidRDefault="000540A8" />
- <w:sectPr w:rsidR="000540A8" w:rsidSect="006C16DB">
  <w:pgSz w:w="12240" w:h="15840" />
  <w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="1440" w:header="720" w:footer="720"
     w:gutter="0" />
  <w:cols w:space="720" />
  <w:docGrid w:linePitch="360" />
   </w:sectPr>
   </w:body>
   </w:document>
```

XML Representation of the Modified Document
Figure 3-2

3.3 Using Rsid Codes To Identify Academic Dishonesty In Students' Work

When a student modifies a document that was originated by someone else, both the originator's codes and the modifier's codes will appear in the document. When a person modifies a document they, in effect, are putting their 'finger-prints' on the document. Furthermore, if a student pastes text from another document, the codes from that document are copied as well.

In Figure 3, we can see that after the document has been modified that the originator's Rsid of 0071385E is still in the document but that now the modifier's Rsids of 00A05879 and 000540A8 have been added.

By reading through a set of documents representing a single particular assignment, it is possible to assert with high certainty that duplicate codes in different documents are an indication that the work was shared between students.

3.3.1 False Positives

Since the Rsid codes are 32-bit values, there are only 0xFFFFFFFF possible codes. Thus, there is a possibility that the same codes will show up in unrelated documents, that is, documents that are actually unique works.

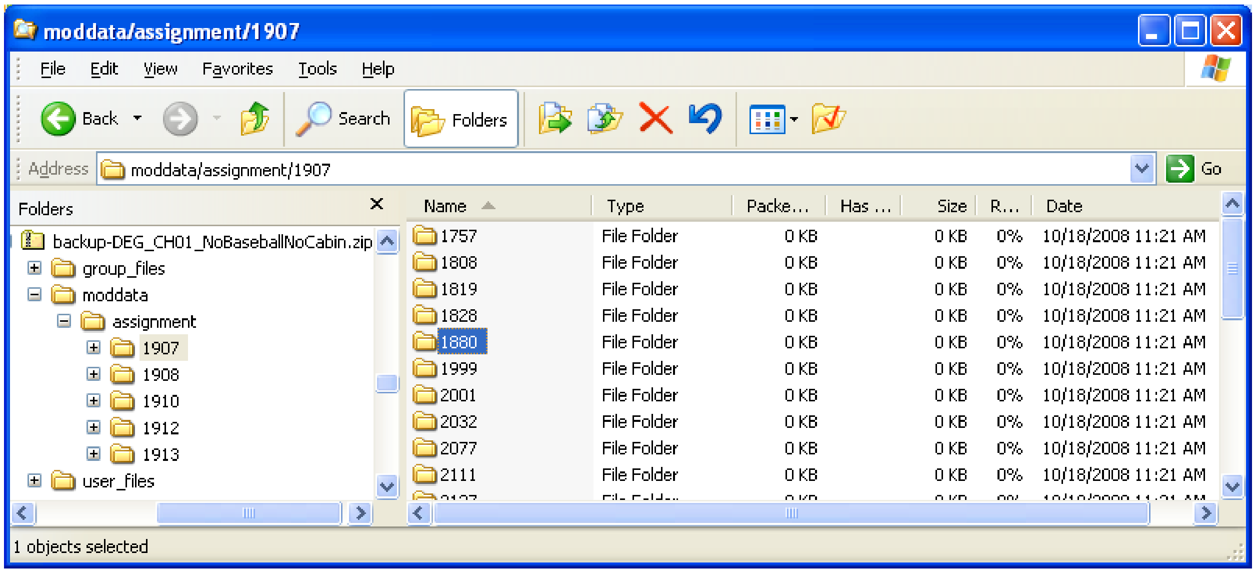
There have been a small number of such false positives in our experience at the University of Dubuque. These have been resolved by one or more of several different strategies:

- The works, when visually inspected, bear no resemblance to each other.
- Questioning of the students involved produces convincing evidence of a lack of collaboration.
- The codes show up in unrelated assignments.

4 Organization of Assignments

Because of the large number of documents to be checked—approximately 18 word assignments for approximately 200 students—we have chosen to compartmentalize the assignments using the Moodle courseware that we use on UDOnline.dbq.edu, our on-line teaching portal.

Moodle permits the export/backup of assignments into large containers. See Figure 4.



moddata/assignment/1907

Address: moddata/assignment/1907

Folders

- backup-DEG_CH01_NoBaseballNoCabin.zip
  - group_files
  - moddata
    - assignment
      - 1907
      - 1908
      - 1910
      - 1912
      - 1913
  - user_files

Name	Type	Packe...	Has ...	Size	R...	Date
1757	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
1808	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
1819	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
1828	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
1880	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
1999	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
2001	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
2032	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
2077	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM
2111	File Folder	0 KB		0 KB	0%	10/18/2008 11:21 AM

1 objects selected

Moodle Backup Container
Figure 4

Each assignment backup consists of a ZIP Compressed Folder with each individual student's work at the bottom of an assignment/student nesting.

The example in Figure 4 highlights student #1880, folder in the 1907 assignment folder. The actual Word document (.docx file) is found in the student folder. There are 4 other assignment folders shown in the left-hand pane, all representing a particular chapter's assignments.

4.1 Processing the Assignment Files

The software developed in our Computer Information Systems Department is used to read through these backup ZIP Compressed Folders. The process involves identifying the particular assignment—1907 in the highlighted example in Figure 4—then identifying each particular student—such as 1757, 1808, 1819, etc.—as shown in Figure 4 and then reading the actual assignment file as a ZIP file from the student folder.

Each assignment file is scanned, and the Rsid codes are duplicate-checked within the document and then exported for further processing. See Figure 5.

In the figure we see instructor name (leigh), year, term, assignment number, student number, file creator's name, last modified by name, and the unique Rsid codes encountered in the document. (In this example, it should be noted that 'Mary Ann Beaumont' as creator is the publisher's agent who produced the initial assignment file that was downloaded from the publisher's website. There will be a set of Rsid codes from the publisher's agent. These 'starter file' codes are separately removed downstream in the processing since those codes would appear in all assignment files.)

```
leigh,2008,fall,1907,1880,Mary Ann Beaumont,Hannah Montana,00374C06
leigh,2008,fall,1907,1880,Mary Ann Beaumont,Hannah Montana,00A2241E
leigh,2008,fall,1907,1880,Mary Ann Beaumont,Hannah Montana,0056039C
leigh,2008,fall,1907,1880,Mary Ann Beaumont,Hannah Montana,00AB738D
leigh,2008,fall,1907,1999,Mary Ann Beaumont,TyCobb,003C2ECC
leigh,2008,fall,1907,1999,Mary Ann Beaumont,TyCobb,001F4369
leigh,2008,fall,1907,1999,Mary Ann Beaumont,TyCobb,0056039C
leigh,2008,fall,1907,1999,Mary Ann Beaumont,TyCobb,008E0BBA
```

Extracted Rsid and Other Identifying Information
Figure 5

4.1.1 Processing the Assignment Rsid Codes

Once the Rsid codes, together with the appropriate assignment and student identification information, are isolated, they are then inserted into a Microsoft Access database for comparison.

A series of SQL queries and supporting tables have been developed that search the database to identify duplicated codes across student files within assignments.

copy1_instructor	copy1_user_id	assignment_name	copy1_created_by	copy1_modified_by	copy2_instructor	copy2_user_id	copy2_created_by	copy2_modified_by
Willcox	2087	WordCh1 Airplane	SueNaumie	SueNaumie	Willcox	1981	SueNaumie	EJSmith
Willcox	2087	WordCh1 Baseball	Mary Ann Beaumont	SueNaumie	Willcox	1981	Mary Ann Beaumont	EJSmith
Willcox	2087	WordCh1 Festival	Shelly Cashman	SueNaumie	Willcox	1981	Shelly Cashman	EJSmith
Willcox	2087	WordCh1 LabCabin	SueNaumie	SueNaumie	Willcox	1981	SueNaumie	EJSmith
Willcox	2087	WordCh1 Tour	Mary Ann Beaumont	SueNaumie	Willcox	1981	Mary Ann Beaumont	EJSmith

Final Evidence of Academic Dishonesty
Figure 6

In Figure 6 we see the final resulting evidence of Academic Dishonesty: Student 2087 (SueNaumie) has shared these 5 assignments with student 1981 (EJSmith).

5 Postlog

This software was developed during the Fall 2008 school term. During that term 53 students were found to be cheating on Word assignments and were generally disciplined in accordance with institutional guidelines.

At the start of the Spring 2009 term, the CIS Departmental chair made presentations on the first session of class section where Word assignments were given. In this presentation, the students were informed of this tool and implored not to share their work.

At the time of this writing, only 15 students have been found to be cheating in this way this term. Some of those were absent when the presentations were made.

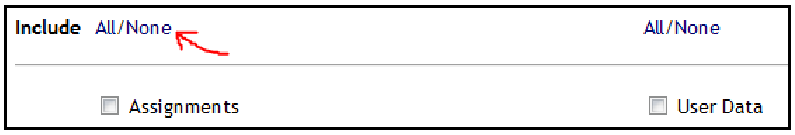
We believe this tool to be a successful deterrent to cheating in these classes.

Appendix User Instructions

Obtaining Assignment Data for Import into Centralized Database

Step 1: Export Zip File Containing Assignment Files from Moodle

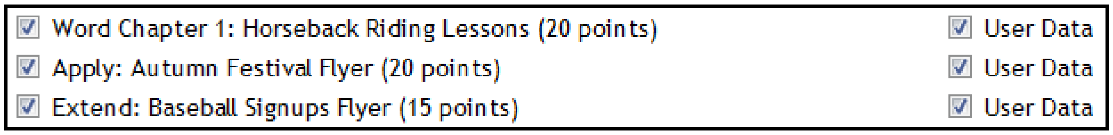
1. Click on the Backup link in the Administration block in Moodle.
2. Click on the "None" link at the top where it says "Include All/None" to clear selections.



Include All/None All/None

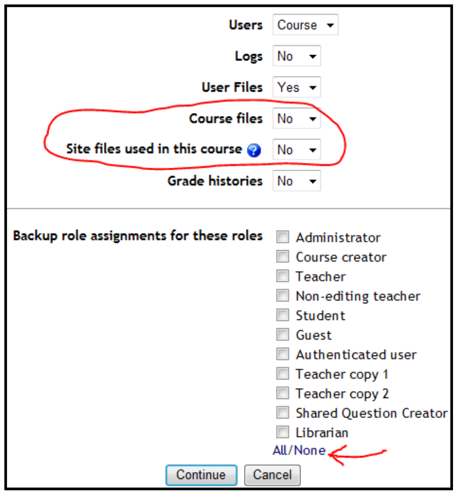
☐ Assignments ☐ User Data

3. Select the assignments you want to export, checking both the Assignment and User Data checkboxes.



☑ Word Chapter 1: Horseback Riding Lessons (20 points) ☑ User Data
☑ Apply: Autumn Festival Flyer (20 points) ☑ User Data
☑ Extend: Baseball Signups Flyer (15 points) ☑ User Data

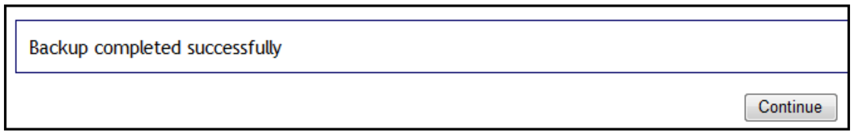
4. Scroll down to the next sections.
   - Change "Course files" to "No".
   - Change "Site files used in this course" to "No".
   - Click the None link to uncheck the roles.
5. Click the Continue button.



Users: Course
Logs: No
User Files: Yes
Course files: No
Site files used in this course: No
Grade histories: No

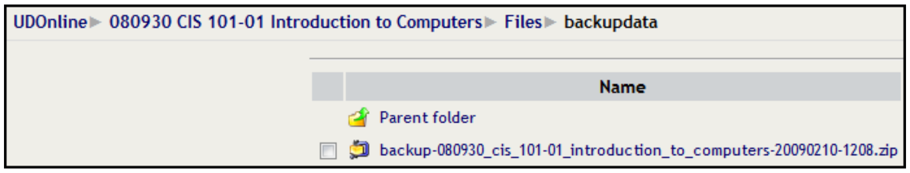
Backup role assignments for these roles:
☐ Administrator
☐ Course creator
☐ Teacher
☐ Non-editing teacher
☐ Student
☐ Guest
☐ Authenticated user
☐ Teacher copy 1
☐ Teacher copy 2
☐ Shared Question Creator
☐ Librarian
All/None

6. Keep the default zip file name (changing it may cause the backup to fail), and click the Continue button.
7. Scroll down to check for the successful completion message.



Backup completed successfully

8. Click the Continue button to proceed to the backupdata folder on the Files page.

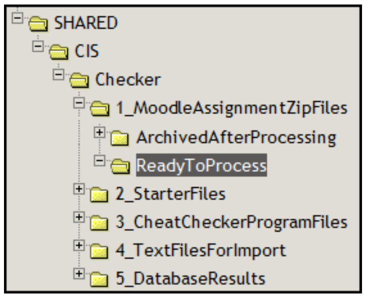


UDOnline ▸ 080930 CIS 101-01 Introduction to Computers ▸ Files ▸ backupdata

Name
Parent folder
☐ backup-080930_cis_101-01_introduction_to_computers-20090210-1208.zip

\

9. Right-click on the newly created zip file, and choose Save Target As to download the file.



Save it in the G:\Shared\CIS\Checker\1_MoodleAssignmentZipFiles\ReadyToProcess folder.

Step 2: Provide Any Starter Files

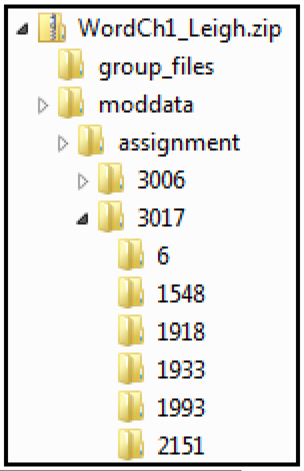
If you provided starter documents for any of the assignments you exported from Moodle (as opposed to having students create documents from scratch), then you need to provide those starter files. (Otherwise the database reports listing possible cases of cheating will incorrectly include students who only shared data from a common starter file.)

Save any starter files in the G:\Shared\CIS\Checker\2_StarterFiles\ReadyToProcess folder.

Step 3: Provide Translation of Moodle Assignment IDs found in Exported Zip File

The zip file exported from Moodle will contain a subfolder for each assignment, where the folder name is a numeric assignment ID. For example, the Word Chapter 1 horseback riding lesson flyer might have an ID of 3006 on one instructor's web site but an ID of 2056 on another instructor's web site. In the Access database, where assignments will be compared across sections, these assignment IDs need to be mapped to the same "assignment name."

Continuing the previous example, the first instructor needs to specify that assignment ID 3006 goes with assignment name WordCh1Horse, and the second instructor would map assignment ID 2056 to WordCh1Horse. A list of assignment name values is found on the next page (and can be added to as needed).



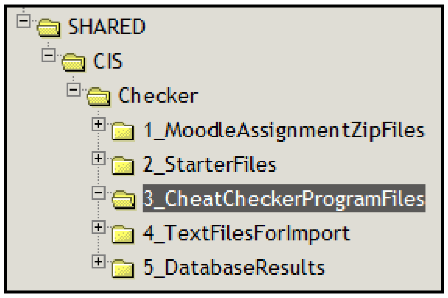
1. Examine the assignment zip file to identify the assignment IDs in the file.

   Each assignment ID has its own folder under moddata\assignment. Under each assignment ID folder there is a folder for each student ID. These student folders contain the actual assignment files.

   For example, the image to the right shows two assignment ID folders, 3006 and 3017, in a zip file. The 3017 folder has been expanded to show several student ID folders, 6, 1548, etc.

2. Examine several assignments under each assignment ID folder to determine what assignment the folder contains.

Step 4: Run Job to Create Text File of Assignment Data from Assignment Zip File



1. Move the assignment zip file to process from 1_MoodleAssignmentZipFiles to 3_CheatCheckerProgramFiles.
2. Edit the appropriate _EditAndRun_<instructor>.bat file to pass the appropriate parameters to MoodleZipAnalyzer.bat. The job has comments at the top explaining the parameters. Normally the only change needed will be to the name of the assignment zip file to process.
3. Run the edited job file. The resulting text file should have the same name as the assignment zip file but with a TXT extension. (The job reads all the assignments in the zip file and writes the appropriate information to a text file, which will later be imported into an Access database for analysis.) The job will open a command prompt window and state the name of the output text file if it is working correctly. It will close automatically when the job is complete. While the window remains open, you can verify progress by seeing the output text file increasing in size.
4. Move the assignment zip file to 1_MoodleAssignmentZipFiles\ArchivedAfterProcessing. If appropriate, rename the zip file to be consistent with the other archived files. (The zip file is kept for future reference, so that assignments from any course section can be examined as needed when evaluating cases reported in the final results.)
5. Move the resulting text file to 4_TextFilesForImport\TextFilesForImport. (The person responsible for loading the assignment text file into the Access database will find it there, load it, and archive the text file in the ArchivedAfterLoading subfolder when done.)

Step 5: Send Email

Email the person responsible for loading the database that you have data ready to be loaded. Include the following in your email:

1. A statement of whether you provided any starter files in step 2
2. Your mapping of assignment IDs to assignment names from step 3
3. The name of the text file you created in step 4, or a request to have the step 4 processing done for you which includes the name of the assignment zip file to be processed