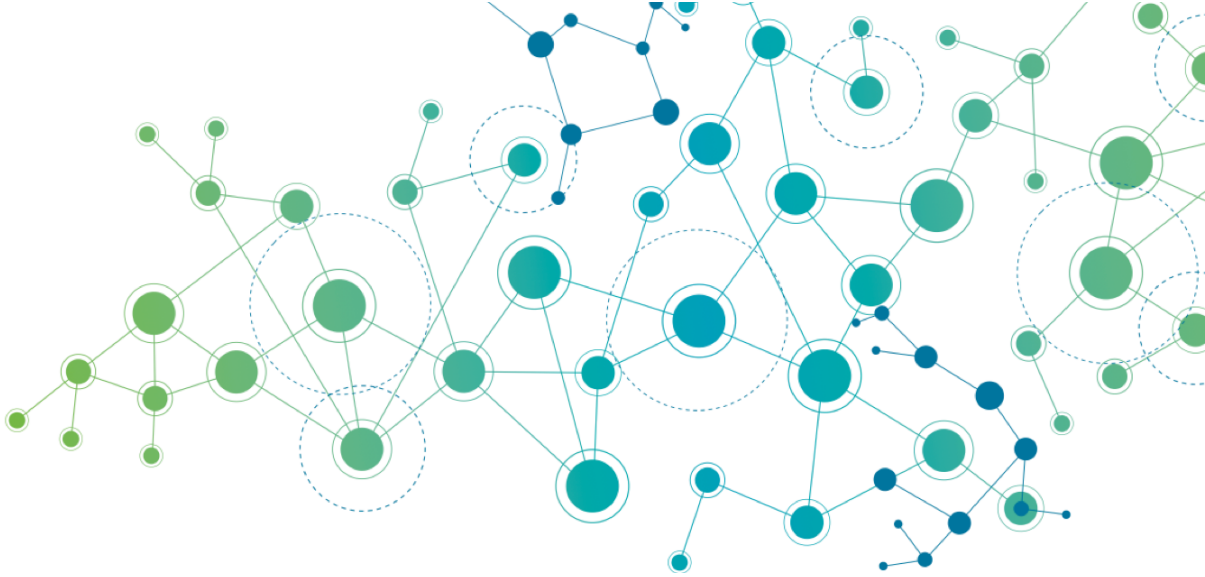


ETD Lifecycle Management Tools Manual

ETD Lifecycle Management Project

Authors: **Matt Schultz, Stephen Eisenhauer, and Nick Krabbenhoeft**

29 September 2014

Version 1.0

Publication Notes

Title: ETD Lifecycle Management Tools Manual

Authors: Matt Schultz, Stephen Eisenhauer, and Nick Krabbenhoeft

Publisher: Educopia Institute, 1230 Peachtree Street, Suite 1900, Atlanta, GA 30309.

Introduction

The IMLS-funded Lifecycle Management of ETDs project (http://www.metaarchive.org/imls) has researched, developed, and documented a suite of modular Lifecycle Management Tools for curating electronic theses and dissertations (ETDs). The project targeted the following curation activities: *Virus Checking*, *Format Recognition*, *Simple ETD & Metadata Submission*, *Preservation Event Record-Keeping*, and *Reference Link Archiving*. This manual describes how to implement tools for those activities.

The manual is written for ETD Program Managers and other Librarians responsible for accepting and archiving ETDs and related content. It describes a general rationale and use case for each curation activity mentioned above in the context of an ETD program. While the technical and administrative implementations of ETD programs are diverse, this manual includes generalized recommendations for where and when to deploy the tools in an ETD submission workflow. ETD Program Managers are encouraged to coordinate with the full range of stakeholders (including the graduate schools, libraries, campus IT, and vendors) to adapt these tools to their implementation.

In most instances these Lifecycle Management Tools are existing, standalone technologies with either *command-line interfaces (CLI)* and/or *graphical user interfaces (GUI)*. The tools have been developed, tested, and documented in the context of curating ETDs. All of the tools are open-source and freely available for download.

All technical terms are italicized and defined in the Glossary of Technical Terms. Examples of commands for *command-line interfaces* are prefaced with a '$' and rendered in `monotype`. The project has also published more information about the broader context of ETD curation in the *Guidance Documents for Lifecycle Management of ETDs* (http://www.educopia.org/publishing/gdlmetd).

Special Note

The project researched the feasibility of incorporating the Lifecycle Management Tools into several freely downloadable open-source submission and/or institutional repository software systems, e.g., DSpace, E-Prints, ETD-db, Open-ETD, and Vireo. In general, the systems mentioned above either:

1. Already have existing modular *APIs* or *plugin architectures* to incorporate standalone curation technologies within their workflows (e.g., DSpace, E-Prints, Vireo); or
2. Do not currently have the necessary *APIs* or *plugin architectures* to reliably support such integrations (e.g., ETD-db, Open-ETD).

The standalone curation technologies investigated and tested in the project do not require any additional *APIs* or *scripts* to make them functional for any of the above-mentioned systems. Those systems without *APIs* or *plugin architectures* (e.g., ETD-db, Open-ETD) would need to have those features developed in order to interoperate with the tools documented below in a secure fashion. The Lifecycle Management of ETDs project encourages all ETD-related software systems to consider development (where not yet undertaken) to support curation activities such as those described here.

1 Virus Checking for ETDs

1.1 Rationale

Virus scanning of ETDs is an important step in the curation lifecycle. ETDs are generated in workspaces with unknown protections and they pass through a number of production workspaces during the submission process. Infected files can damage the originating ETD, other ETDs in the collection, and the computing systems used to store, maintain, or disseminate ETDs. Identifying viruses and other malware is an essential step in protecting the stability and integrity of an ETD repository over time.

Technical services and computing environments responsible for hosting submission systems are the first line of defense when it comes to preventing infected files from entering an ETD repository. The steps taken after detection depend on the workflow but require either quarantining and treating the files in question or asking the students to resubmit clean versions of the file.

1.2 Virus Checking Use Cases for ETDs

ClamAV is an open and widely used virus-scanning tool. It is best used as a standalone utility at or near the point of the initial submission. Where possible, ClamAV should be incorporated directly into the system being used to handle author submissions.[1] ClamAV has an *API* (LibClamAV) that can be used to add ClamAV programmatically into an open source ETD submission system; see 1.2.2 for more details. The ETD Drop application described below in *section 3* has support for virus detection built-in.

1.2.1 1.2.1. Manual Use of ClamAV (CLI application)

For best usage, ClamAV should be installed by IT staff on the server where ETDs are submitted. Any graduate school or library staffs that are typically responsible for approving an ETD can use a *command-line console* to check the submitted ETDs via *ssh accounts* with appropriate permissions and brief training and instructions from IT staff. Users of the tool can direct ClamAV to output the tool's results as a text file to a specified directory for review. Library/campus IT staff may also perform virus checks on the graduate school's behalf. Depending upon the *platforms* being used there are several graphical user interface (GUI) applications that make use of ClamAV. However, in many cases graduate school or library staff will be accessing ETD submissions remotely, making it difficult for these GUI applications to access a submission across a network.

1.2.1.1 Obtaining & Installing ClamAV

ETD Program Managers or the relevant Librarian(s) should share this manual with the graduate school liaison for their ETD program. In conjunction with the graduate school liaison, the relevant library/campus IT staff should be contacted to discuss the proper implementation and training as described above.

[1] ETD Program Managers are encouraged to check with their library/campus technical services to determine whether campus-wide support for virus detection may already be in force. This could help to diminish the risk level that would necessitate a special implementation for ETD submissions.

To download ClamAV for Linux, BSD, and Windows *platforms*, see:
http://www.clamav.net/lang/en/download/packages/.

To read documentation on ClamAV, see:
http://www.clamav.net/lang/en/doc/.

1.2.1.2 Starter Instructions for Using ClamAV

Once ClamAV is installed there are several included *clients* that can be used. The *client* known as clamscan is sufficient for diagnosing whether an ETD submission contains a virus. Below are some sample commands to use (lines beginning with '$') and examples of their outputs:

Ex. 1: Scanning an ETD ZIP submission

```
$ clamscan ETD_Submission.zip
```

The command above should return a line of output like this:

ETD_Submission.zip: Worm.Mydoom.U FOUND

Ex. 2: Scanning a Single ETD File

```
$ clamscan /data/ETD_Submission_File/ETD_Submission_File.pdf
```

The command above should return a line of output like this:

/data/ETD_Submission_File/ETD_Submission_File.pdf: Worm.Sober FOUND

Ex. 3: Scanning Multiple ETD Files

```
$ clamscan /data/ETD_Submission_File/*
```

The command above should return a line of output like this:

/data/ETD_Submission_File/ETD_Submission_File.pdf: Worm.Sober FOUND
/data/ETD_Submission_File/metadata/ETD_Submission_Metadata.xml: OK
/data/ETD_Submission_File/supplementals/ETD_Submission_Supplemental_File_1: OK
/data/ETD_Submission_File/supplementals/ETD_Submission_Supplemental_File_2: OK

1.2.1.3 What to Do With the Results

You can tell clamscan to output the scan results (which includes a summary) to a text file in a location that you specify by appending the following *command-line option* to your clamscan command (`>> /path/to/save/report/file.txt`).

Ex. 1: Outputting a Full Summary Report

```
$ clamscan ETD_Submission_File.pdf >> /path/to/save/report/file.txt
```

Example 1 above will produce a text file report that reads as follows:

> ETD_Submission_File.pdf: OK
> ---------- SCAN SUMMARY ----------
> Known viruses: 3543913
> Engine version: 0.98.1
> Scanned directories: 0
> Scanned files: 1
> Infected files: 0
> Data scanned: 0.35 MB
> Data read: 0.20 MB (ratio 1.78:1)
> Time: 16.024 sec (0 m 16 s)

In addition, you can tell clamscan to output only the FOUND results (i.e., detected viruses) to a text file in a location that you specify. Just append the following *command-line option* to your clamscan command (`| grep FOUND >> /path/to/save/report/file.txt`). This works well in cases where you are scanning multiple files and only want to see a list of offending files. See below for examples:

Ex. 2: Outputting a FOUND Viruses Report

```
$ clamscan /data/ETD_Submission_File/* | grep FOUND >>
/path/to/save/report/file.txt
```

Example 2 above will produce a text file report that reads as follows:

> ETD_Submission_File.pdf: Worm.Sober FOUND

If any infected files are found they should be deleted on the submission server or quarantined until IT staff can review the results and recommend a virus/malware removal tool. The student submitter should be notified of the infected file(s), and in the case of PDFs, encouraged to recreate the final submission on a clean computer and then re-submit. Supplemental files, if infected, may prove more problematic – they may need to be excluded from the final submission. Consult your library/campus IT specialists for policy and/or troubleshooting advice in these circumstances.

1.2.2 Automated Use of ClamAV (CLI application)

Automating the use of ClamAV upon submission is possible but only recommended for systems and infrastructures that employ the use of *APIs* and *plugin architectures*. The reason being that any "rough-and-dirty" scripted integrations would be highly susceptible to errors and malfunction after routine software updates or updates to the overall code base. Systems such as Archivematica and DSpace are excellent examples of submission and repository systems that have incorporated ClamAV as an automated curation task. These systems make use of the aforementioned LibClamAV *API*.

2 Format Recognition for ETDs

2.1 Rationale

Format recognition is increasingly important in the ETD curation lifecycle. ETD submissions are expanding from just PDFs to encompass supplemental digital works that include multimedia, datasets, and other digital objects. Identifying and recording the file types of these (occasionally proprietary) digital formats is essential to ensuring that they can be preserved and accessed both immediately and in the future. Formats may require specific software or even specific software versions in order to be rendered accurately. Additionally, knowing the file formats included in an ETD submission can be extremely helpful for ETD program curators and managers to support preservation and access.

Similar to virus checking, format recognition is a task that likely falls first and foremost to the graduate schools or other unit, such as the library hosting the primary submission system. Checking a submission's compliance with the institution's ETD guidelines is the most appropriate time to record the submission's file format. If a file format does not meet valid requirements, the student author should be contacted to correct the issue. As an additional benefit, collecting this information early in both human and machine-readable formats systematizes follow-on record-keeping processes.

2.2 Format Recognition Use Cases for ETDs

The Digital Record Object Identification (DROID) application is a user-friendly format recognition application for a graduate school or other front-line submission unit such as the library to deploy. It is the only format recognition tool with a *graphical user interface (GUI)*. There are also *command-line interface (CLI)* applications such as Unix `file`, the File Information Tool Set (FITS), and JHOVE2 that can be run in a *command-line console* and have their outputs printed to the screen or placed in a simple text or spreadsheet usable format (e.g., tsv, csv).[2]

2.2.1 Use of DROID (GUI application)

Once DROID has been downloaded and installed (see 2.2.1.1) be sure to consult the Running DROID.txt file for setting permissions properly. DROID can be activated on Windows platforms by double-clicking on the droid.bat file, or on Unix *platforms* (Mac or Linux) by navigating to the directory where you extracted the files and typing `./droid.sh`. The application interface will open. As long as the ETD submission files are accessible to the application, a user can follow the steps visualized in 2.2.1.2 to produce a format recognition report. See 2.2.1.3 for best practices for using and managing the outputs. DROID is highly recommended for institutions looking to get started quickly with identifying and managing file formats included with ETD submissions.

[2] While these *CLIs* have a number of *command-line options* to tailor the tools to a specific need, the output of even the simplest command is useful. These tools report an extensive amount of data in XML that will be of interest to curators later in the ETD workflow, but the reports require some careful attention to extract the most relevant elements. See sections 2.2.2 – 2.2.4

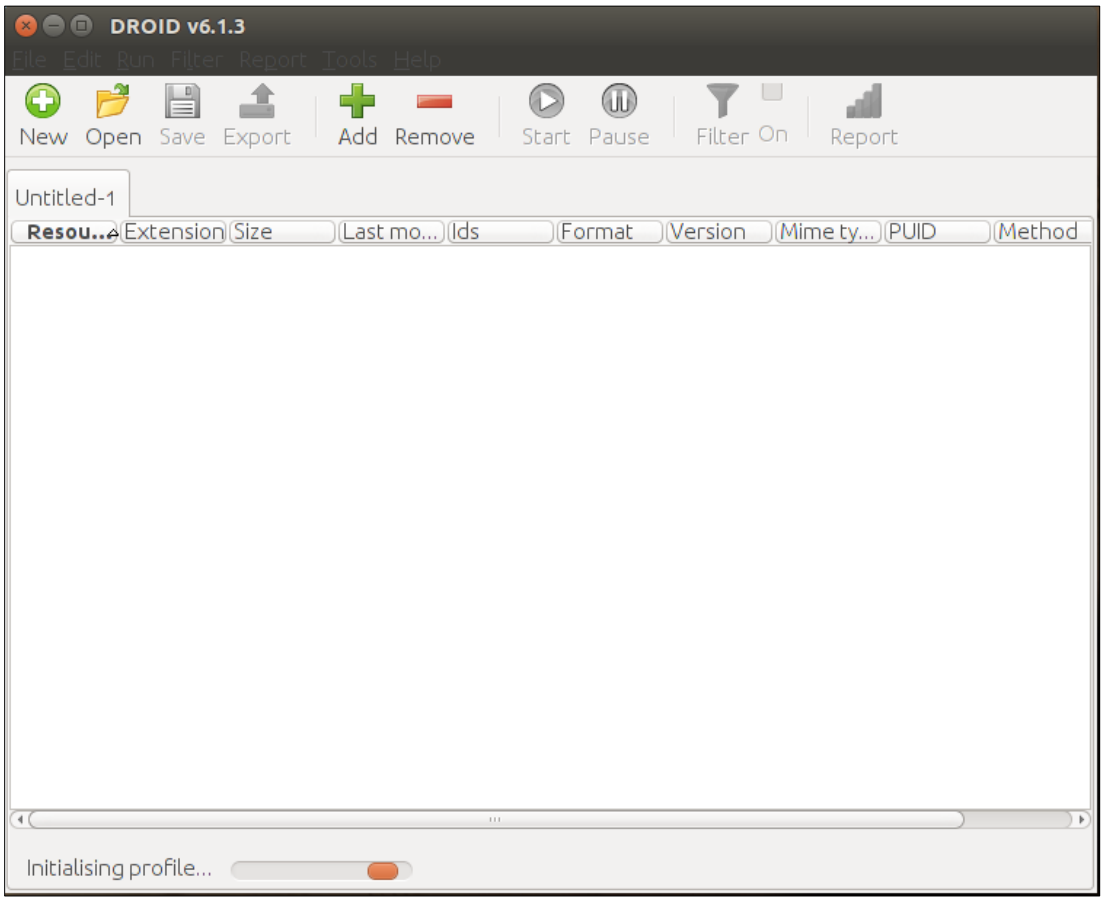
2.2.1.1 Obtaining & Installing DROID

To download DROID for Windows, Mac, and Linux *platforms* (requires Java 6), see: http://www.nationalarchives.gov.uk/information-management/projects-and-work/droid.htm.

To start the graphical interface for DROID:

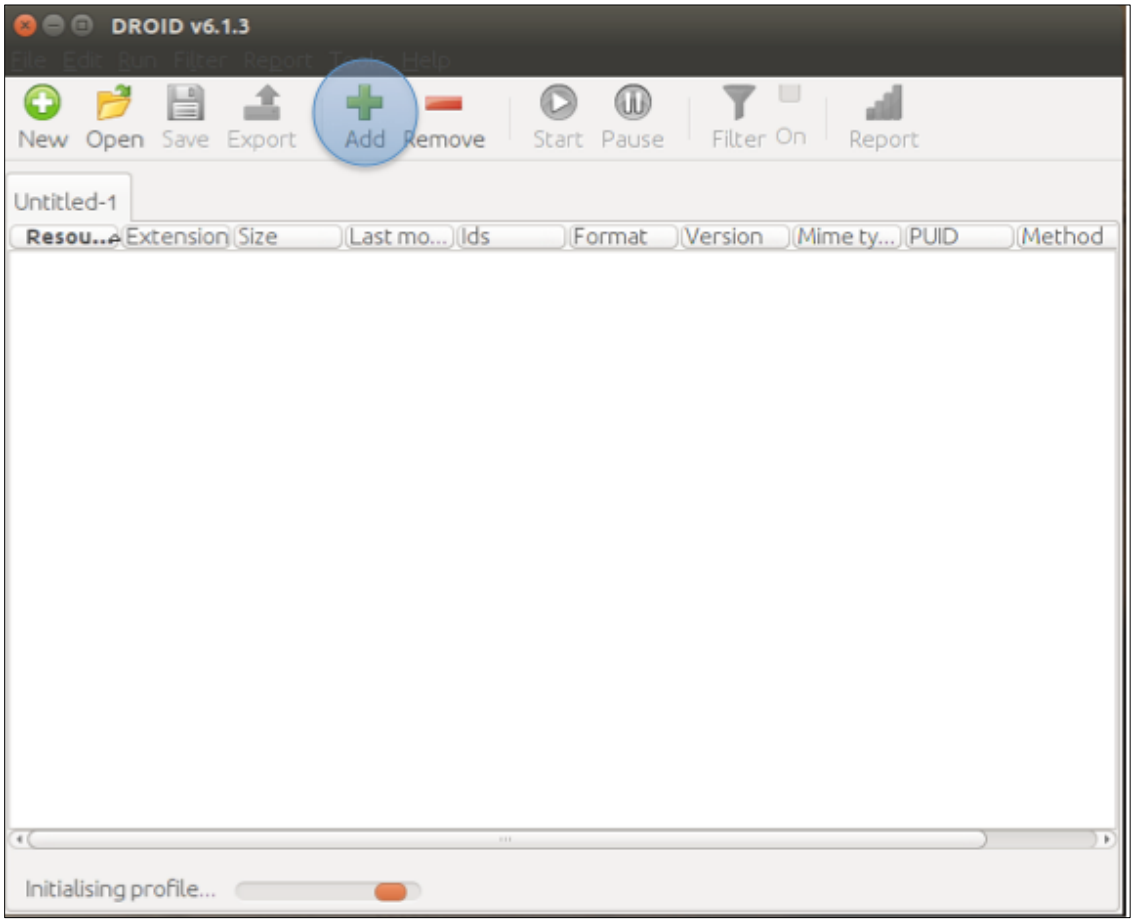
1. If using Windows, navigate to the folder where you extracted DROID and double-click the icon for "droid.bat".
2. If using Linux or OSX:
   a. Open a *command-line console* and navigate to the directory where you extracted the files.
   b. Type `./droid.sh` and press enter.
3. At this point, the interface should appear. It may offer to download new updates.

2.2.1.2 Starter Instructions for Using DROID

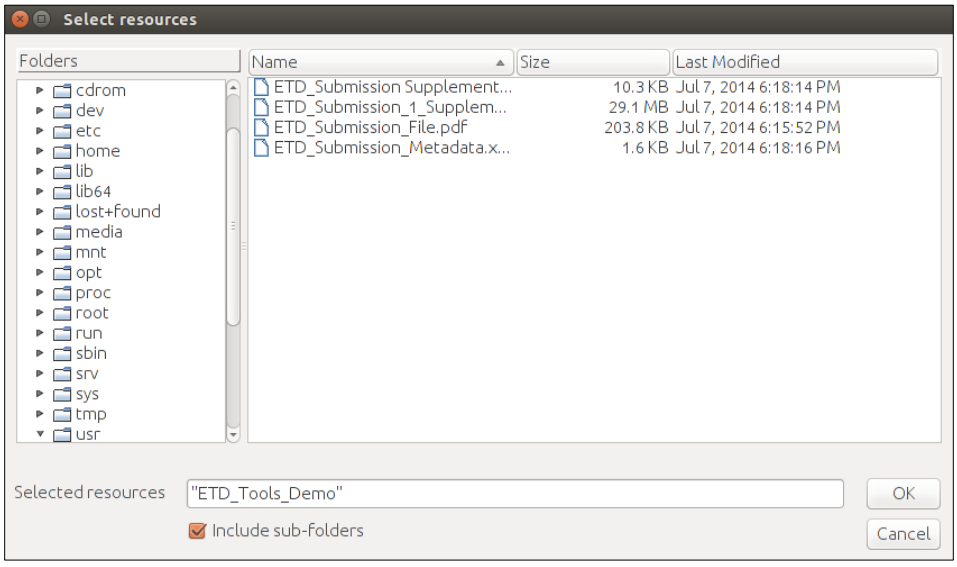


Step 1: Open DROID.[3]

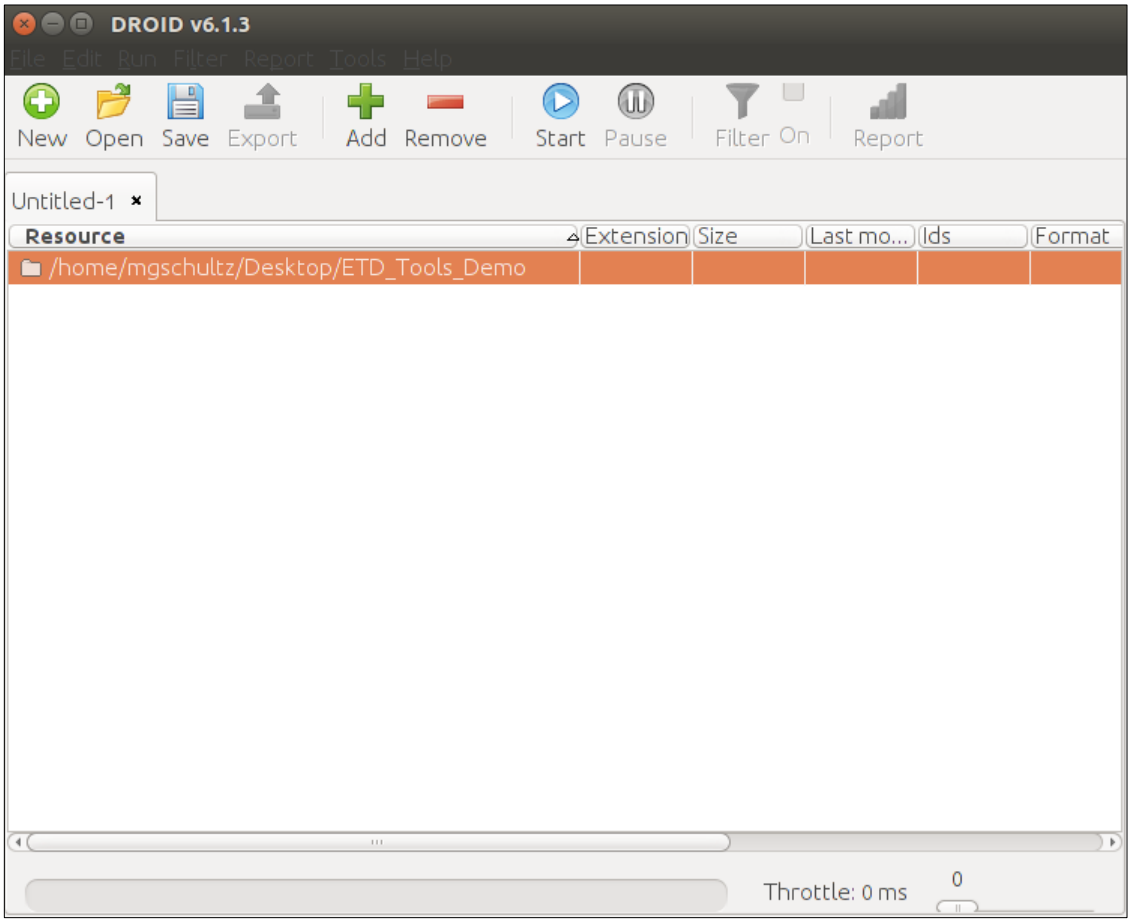
[3] All provided screenshots are from DROID running on a Linux Ubuntu computer platform, but the application looks and behaves exactly the same on Mac and Windows platforms.



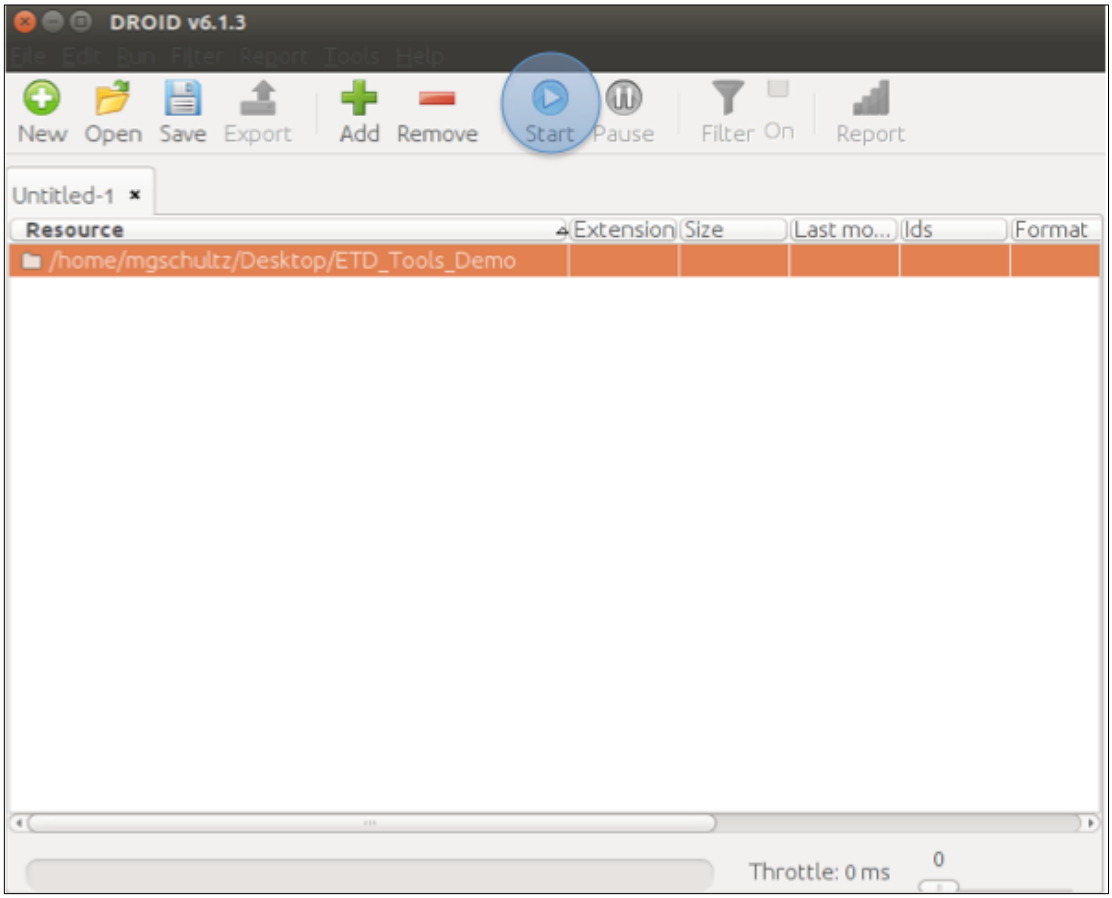
Step 2: Click the Add button to select a file or directory.



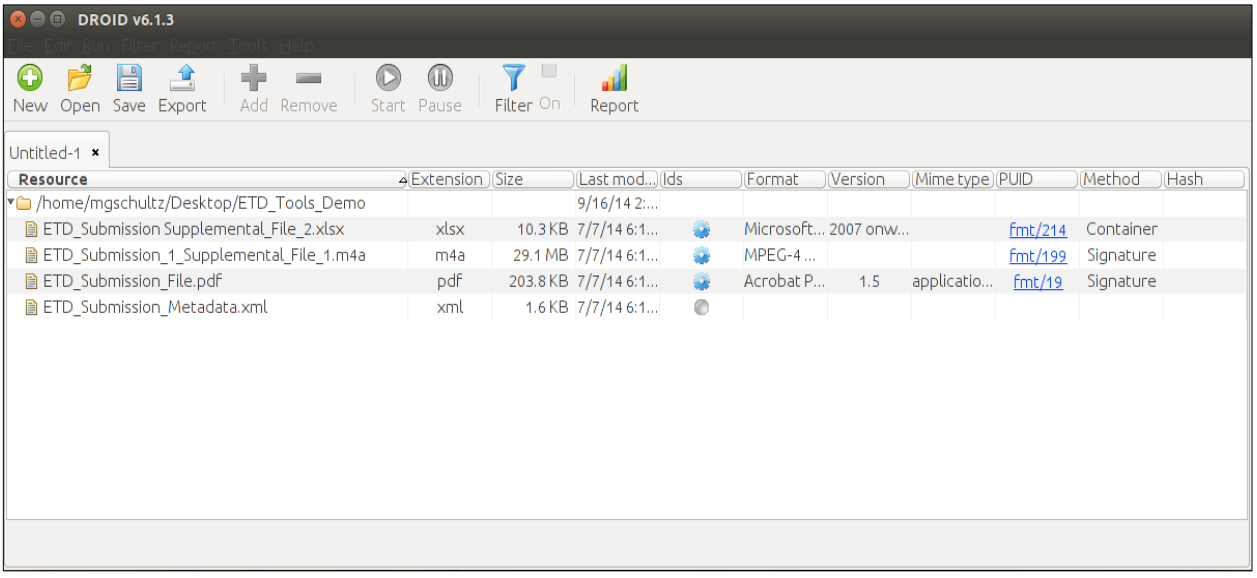
Step 3: Choose a folder or set of files.



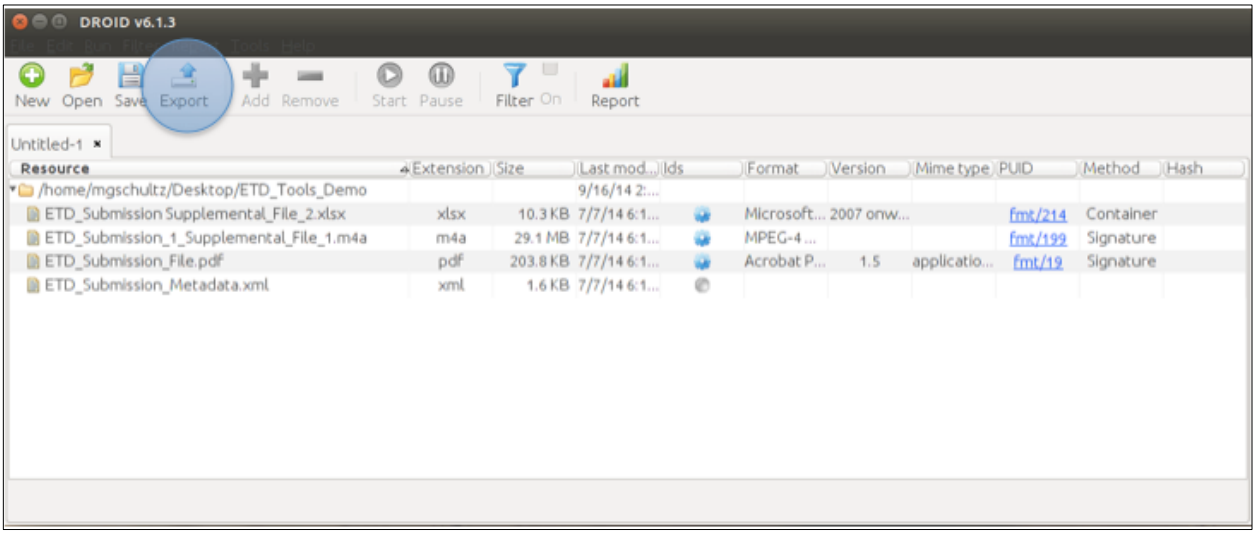
Step 4: Verify that DROID has queued up your selection.



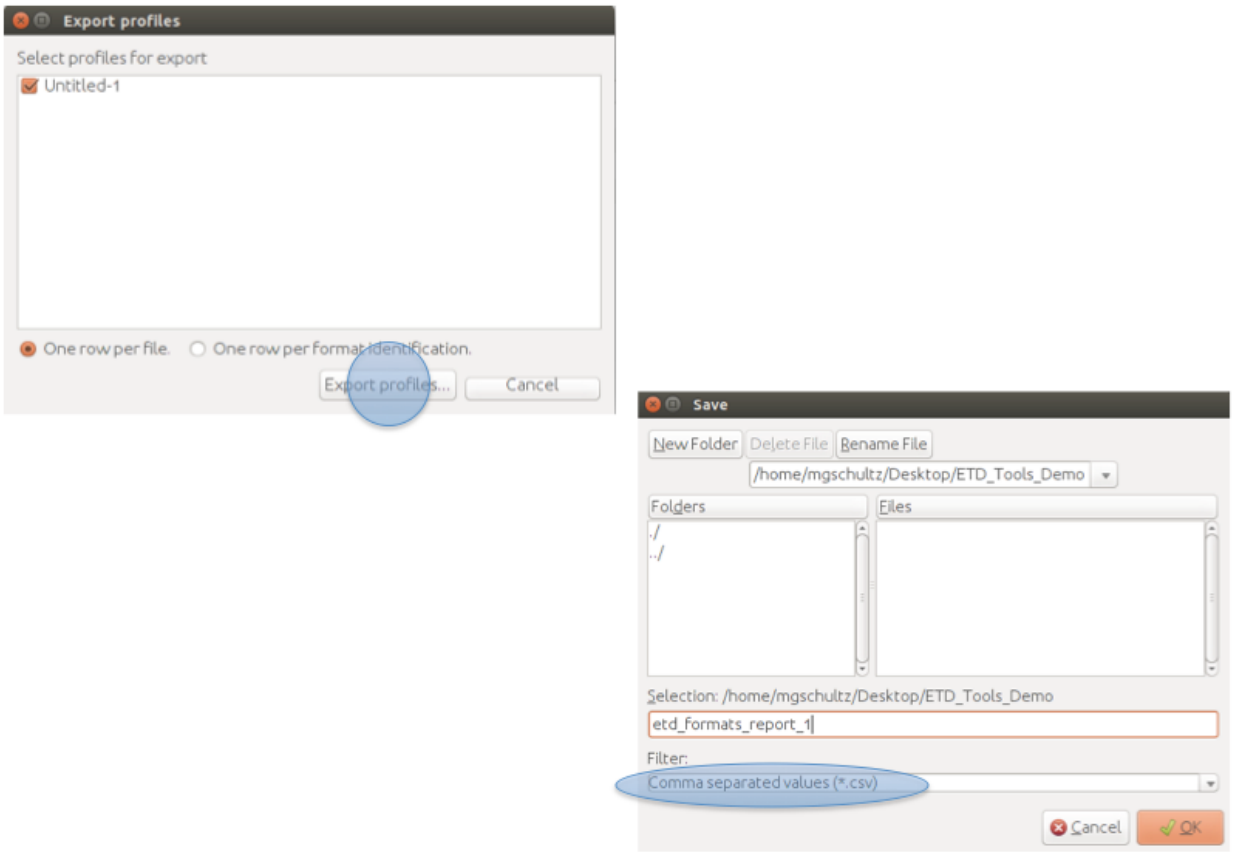
Step 5: Click “Start” to start the file recognition process.



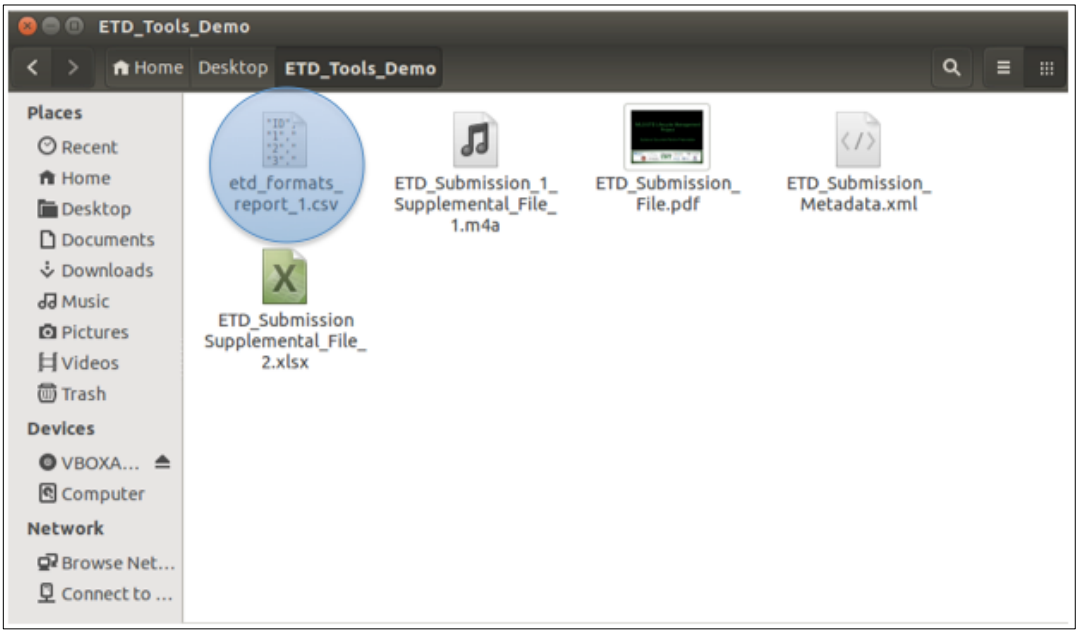
Step 6: Review the results in DROID.



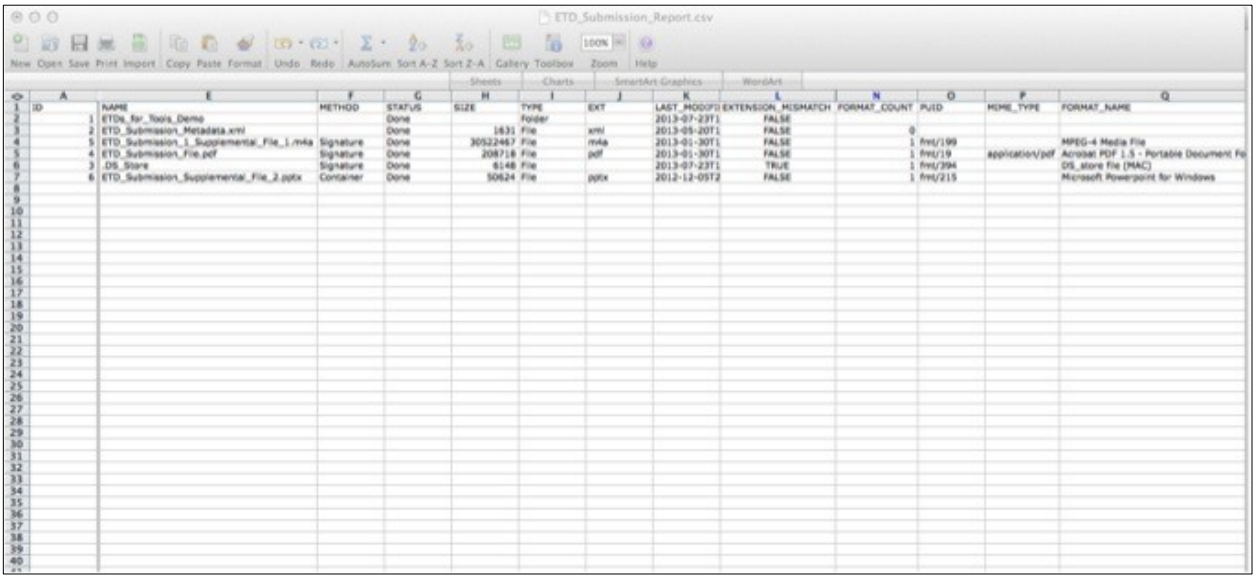
Step 7: Click “Export” to produce a report file.



Step 8: Customize your export report by choosing the name, format (CSV is recommended), and destination directory.



Step 9: Find and open your exported report. CSV files can be opened with Microsoft Excel.



Step 10: Review Your report.

2.2.1.3 What to Do With the Results

The most helpful categories in a DROID report are the extension (EXT), the PRONOM Unique ID (PUID), the MIME type (MIME_TYPE), and the format name (FORMAT_NAME) columns. The PRONOM Unique ID and the format name contain the most extensive details about the file format. The PRONOM Unique ID can be used to perform a search for format details in the PRONOM Format Registry hosted by the UK National Archives.

To search the PRONOM Format Registry, see:
http://www.nationalarchives.gov.uk/PRONOM/PUID/proPUIDSearch.aspx?status=new

Where possible PRONOM will provide details on format risks, rights (open or proprietary), and related file types that may serve as migration pathways.

DROID's reports can be stored with an initial ETD submission and passed along with the content for reference by other stakeholders later in the ETD submission workflow cycle.

2.2.2 Use of Unix `file` (CLI application)

Unix `file` is extremely helpful in those cases where an ETD submission staff person has been given setup, proper permissions, and very light training in the use of a *command-line console*. If you are already up and running with using a *command-line console* to run a service like ClamAV (see 1.2.1) then Unix `file` will be easy to add to your toolset. It can be used to analyze files one-by-one by printing results directly to the screen, or it can be scripted to report on multiple files in a directory and/or set of sub-directories. In those cases the *script* should be set up to produce outputs in a simple text format (e.g., tsv, csv). See your library/campus IT staff for the development of these *scripts* – they will know how to best engineer these in standardized ways for your *platform* and setup. Ultimately such *scripts* should be no more difficult to run than the standalone `file` command itself.

2.2.2.1 Obtaining & Installing Unix `file`

The Unix `file` command is pre-installed on most OSX and Linux *platforms*. If the command isn't already available on yours, you will need to install it using your operating system's package manager (the package should be named just "file"). Consult with your IT staff to have `file` installed.

Unix commands can be used on Windows *platforms* with the use of Cygwin, a Unix console emulator. For information about Cygwin, see: http://www.cygwin.com/.

2.2.2.2 Starter Instructions for Using Unix `file`

Once you have `file` installed on your system, getting started with it is very easy. Open a *command-line console* and navigate to the data you want to analyze. For instance, to analyze a file named ETD_Submission_Supplemental_File_1 (note that this file is not showing its file type extension), type:

Ex: Basic file Output

```
$ file ETD_Submission_Supplemental_File_1
```

The command above should return a line of output like this:

ETD_Submission_Supplemental_File_1: ISO Media, MPEG v4 system, iTunes AAC-LC

The `file` command is reporting that this is an audio and/or video file using iTunes AAC-LC encoding and compression. The Unix `file` command offers a handful of *arguments* that can change the format of the output string to perhaps offer a few more details. The following are some useful examples:

Ex. 1: Brief file Output

```
$ file --brief ETD_Submission_Supplemental_File_1
```

The command above should return a line of output like this:

ISO Media, MPEG v4 system, iTunes AAC-LC

Ex. 2: Type file Output

```
$ file ETD_Submission_Supplemental_File_1 -i
```

The command above should return a line of output like this:

ETD_Submission_Supplemental_File_1: regular file

Ex. 3: Brief Unix Type file Output

```
$ file ETD_Submission_Supplemental_File_1 --brief -i
```

The command above should return a line of output like this:

regular file

Ex. 4: Brief MIME Type file Output

```
$ file ETD_Submission_Supplemental_File_1 --brief --mime-type
```

The command above should return a line of output like this:

audio/mp4

You can use the command `file --help` or `man file` to learn more about the *command-line options* and *arguments* for the version of `file` you have installed.

2.2.2.3 What to Do With the Results

To output the results of `file` to a text file in a location you specify, append the following to your command (`>> /path/to/save/report/file.txt`). For example:

```
$ file ETD_Submission_Supplemental_File_1.pdf >>
/path/to/save/report/file.txt
```

You can then open the text file and review the results.

2.2.3 Use of FITS (CLI application)

The File Information Tool Set (FITS) is a robust tool for identifying a file format and determining its adherence to its format specification. FITS bundles several format recognition tools and compares their results. Conflicts (i.e., disagreement about whether the file format is what it purports to be) are also recorded in the reports. By default, the output from FITS defaults to a custom FITS XML representation

that prints to the console, but it can also output to a simple text file. Much like the Unix `file` command, FITS requires setup, proper permissions, and training in the use of a *command-line console*. Below are a set of basic FITS commands to produce a report. See 2.2.3.3 for an explanation of how to interpret and use the results.

2.2.3.1 Obtaining & Installing FITS

To download FITS for Linux and Mac *platforms* and to read its documentation, see:
http://projects.iq.harvard.edu/fits

2.2.3.2 Starter Instructions for Using FITS

Instructions:

- Open a *command-line console* and navigate to the directory where you extracted FITS.
- Type `./fits.sh -h`. You should now see a list of *command-line options* and their explanations.
- The basic usage for a single input file is `./fits.sh -i path/to/file`.

Ex: Basic FITS Usage for ETD Submissions

```
./fits.sh -i /data/ETD_Submission/ETD_Submission_1_Supplemental_File_1
```

The above command will output a series of XML encoded information that approximates the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<fits xmlns="http://hul.harvard.edu/ois/xml/ns/fits/fits_output"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://hul.harvard.edu/ois/xml/ns/fits/fits_output
http://hul.harvard.edu/ois/xml/xsd/fits/fits_output.xsd" version="0.8"
timestamp="9/17/14 1:47 PM">
  <identification>
    <identity format="MPEG-4 Audio" mimetype="audio/mp4"
toolname="FITS" toolversion="0.8">
      <tool toolname="file utility" toolversion="5.14" />
      <tool toolname="Exiftool" toolversion="9.13" />
    </identity>
  </identification>
  <fileinfo>
    <lastmodified toolname="Exiftool" toolversion="9.13"
status="SINGLE_RESULT">2014:07:07 18:18:14-04:00</lastmodified>
    <created toolname="Exiftool" toolversion="9.13"
status="SINGLE_RESULT">2013:01:30 20:27:43</created>
    <filepath toolname="OIS File Information" toolversion="0.2"
status="SINGLE_RESULT">/home/mgschultz/Desktop/ETD_Tools_Demo/
ETD_Submission_1_Supplemental_File_1.m4a</filepath>
    <filename toolname="OIS File Information" toolversion="0.2"
status="SINGLE_RESULT">ETD_Submission_1_Supplemental_File_1.m4a</
filename>
    <size toolname="OIS File Information" toolversion="0.2"
status="SINGLE_RESULT">30522467</size>
    <md5checksum toolname="OIS File Information" toolversion="0.2"
status="SINGLE_RESULT">7f19265880c299ad8c01abdbe3fcc3ba</md5checksum>
    <fslastmodified toolname="OIS File Information" toolversion="0.2"
status="SINGLE_RESULT">1404771494000</fslastmodified>
  </fileinfo>
```

2.2.3.3 What to Do With the Results

To read the documentation on how to interpret FITS output, see:
http://projects.iq.harvard.edu/fits/understanding-output

The first and most important place to look in the FITS output is the `<identification> </identification>` section. If the opening `<identification>` tag element does not report `<identification status="CONFLICT">`, the relevant format analysis tools that are supported by FITS did not encounter an identification issue. Depending on the file type being analyzed FITS will have more or less to report. The Lifecycle Management of ETDs project encourages ETD programs that decide to use FITS to spend some time reading through the FITS User Manual to become better acquainted with the full range of report details. The File Info and Metadata Sections can often provide a great deal of helpful technical information about the file(s) being analyzed.

The outputs of FITS can be directed by using:
`./fits.sh -o /path/to/save/report/file.txt.`

This saves the program's output to a file instead of writing it to the *command-line console* display.

2.2.4 Use of JHOVE2 (CLI application)

JHOVE2 is a successor tool to JHOVE (no longer supported). JHOVE2 is another robust format recognition tool that requires use of a *command-line console*. Of the three *command-line interface (CLI)* tools (Unix `file`, FITS, JHOVE2) described in this manual, JHOVE2 is probably the most difficult to work with from the standpoint of its report outputs, which are geared toward being further machine processed into preservation metadata schemas such as METS and/or PREMIS. That being said, JHOVE2's output in its default output form can be useful if you know what you are looking for. At the time of the release of this manual (Fall 2014) the Lifecycle Management of ETDs project recommends that most ETD programs start with DROID (see 2.2.1) or FITS (see 2.2.3) rather than with JHOVE2. Like those tools, JHOVE2's output might be useful to later stages of ETD curation (e.g., use cases for METS and/or PREMIS).

2.2.4.1 Obtaining & Installing JHOVE2

To download JHOVE 2 for Linux and Mac *platforms* and to read documentation for JHOVE2, see:
https://bitbucket.org/jhove2/main/wiki/Home

All you have to do is download the current version from the Project homepage and extract the contents.

Technical Note (consult your IT services as needed): JHOVE2 works with both Java 6 and Java 7, but produces many warning messages under Java 7. If you have a Java 6 Java Runtime Environment (JRE) available on your system, you may want to use it for running JHOVE2. To run JHOVE2 with Java 6, use the following command in your *command-line console* before using JHOVE2:
`export JAVA_HOME=/path/to/jre1.6.0_xx/`

The shown path should be replaced with the appropriate 1.6.0_ version. The effects of this command are not permanent, so you will need to run it each time you open a new terminal session (or in each shell *script* you wish to use with JHOVE2).

2.2.4.2 Starter Instructions for Using JHOVE2

Instructions:

1. Open a *command-line console* and navigate to the directory where you extracted JHOVE2
2. Run JHOVE2 and provide the file system path to the files of interest:
   a. `./jhove2.sh /data/ETD_Submission/ETD_Submission_File.pdf`

Beyond that, there are not many other *command-line options*. JHOVE2 produces a lot of information by default, and it's up to you to sift through the output to get the parts you're interested in. See below.

2.2.4.3 What to Do With the Results

The outputs of JHOVE2 can be directed by using:

1. `./jhove2.sh -d JSON|Text|XML|CMD|CDX` : This selects the priority of the format of JHOVE2's output, which is printed to your command-line console display. "Text" is used by default, but others can be helpful depending on your needs.
2. `./jhove2.sh -o /path/to/save/report/file.txt`: Saves the program's output to a simple text file instead of writing it to your command-line console display.

From a human-readable standpoint, the most important elements to pay attention to are those at the very beginning of the output. Below is an example:

```
FileSource:
 StartingOffset (byte): 0
 EndingOffset (byte): 1562309
 Size (byte): 1562310
 FileSystemProperties:
  Path:
/Users/username/data/ETD_Submission_File/ETD_Submission_File.pdf
  LastModified: 2014-03-20T00:45:54-04:00
 PresumptiveFormats:
  PresumptiveFormat {FormatIdentification}:
   NativeIdentifier {I8R}:
    Value: fmt/95
    Namespace: PUID
   JHOVE2Identifier {I8R}:
    Value: http://jhove2.org/terms/format/pdf
    Namespace: JHOVE2
   IdentificationProduct {I8R}:
    Value:
http://jhove2.org/terms/reportable/org/jhove2/module/identify/DROIDIde
ntifier
    Namespace: JHOVE2
   Confidence: Tentative
```

The NativeIdentifier Value (e.g., fmt/95) can be queried through PRONOM using the link below, just as with DROID (see 2.2.1.3). In the above case PRONOM explains that fmt/95 is a PDF/A file.

To search the PRONOM Format Registry, see:
http://www.nationalarchives.gov.uk/PRONOM/PUID/proPUIDSearch.aspx?status=new

The full output can be saved in any preferred output along with the ETD submission contents, and referenced as needed by ETD stakeholders responsible for depositing the ETD submission in the institution's repository.

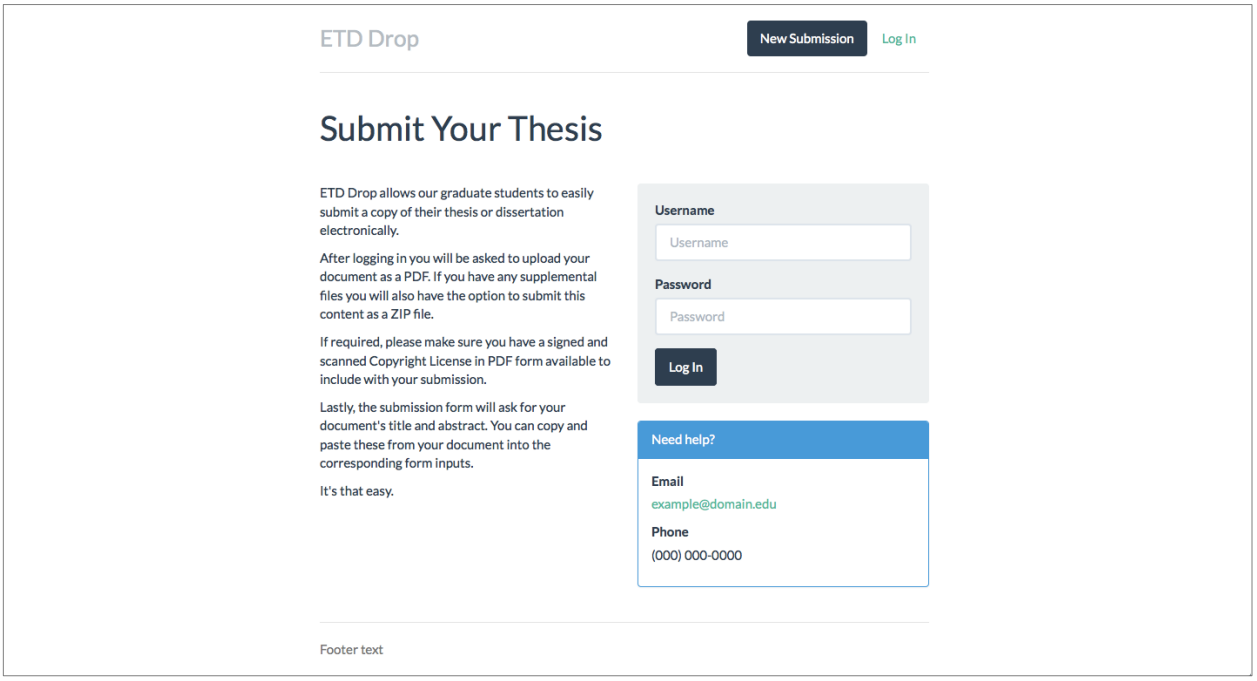
3 Simple ETD Submission

3.1 Rationale

Submission systems for ETDs can span a diversity of implementations depending on the relationship of the ETD program stakeholders and their available technical and administrative resources. At some institutions, the first point of deposit is with a vendor service such as ProQuest. At other institutions, it may be through a submission service provided through the graduate school or library in coordination with library/campus IT. The submission system configuration can run the gamut from the use of a simple mail send and an *FTP* transfer to the use of Vireo or another homegrown application (e.g., ETD-db). To get started, ETD authors and ETD programs need a simple capability for users to deposit ETDs into a remote location via a webform that gathers submission information required by the ETD program. The application should facilitate quality control and approval of an ETD submission by the appropriate ETD program stakeholders.

3.2 Simple Submission Use Case for ETDs

For those institutions that may be in early stages of accepting ETDs and would like a professional application interface without the responsibility of having to manage a database, the Lifecycle Management of ETDs project has developed a lightweight web service known as ETD Drop (https://github.com/MetaArchive/etd-drop). ETD Drop was designed for ETD programs that have limited resources, want to quickly begin facilitating ETD submissions, and can acquire the assistance of library/campus IT services without demanding a large amount of on-going support and hosting.



3.2.1 Use of ETD Drop for ETDs

ETD Drop (https://github.com/MetaArchive/etd-drop) provides a simple submission interface for an ETD author's files (pdf and supplemental files). The application gathers a set of helpful submission metadata, written to XML and JSON, that can then be used to complete a fuller set of ETD-specific metadata (e.g., ETD-MS) for later cataloging purposes. It places an ETD submission on the server's file system rather than in a database, and thereby facilitates direct access to the submission files for the purposes of review and approval (supports the use of a web browser, file browser, *FTP*, *ssh*, etc.).

Most importantly, the ETD submission is normalized and packaged with BagIt (http://www.digitalpreservation.gov/documents/bagitspec.pdf) to facilitate later processing by the ETD program's library/campus IT to transfer the files from the receiving server to the institutional repository. BagIt produces an inventory and set of per-file checksums that can be used for end-to-end validation in such transfers. BagIt does not render the data itself inaccessible prior to any such transfers (as opposed to use of ZIP or TAR).

ETD Drop includes support for a scripted implementation of the DAITSS Description Service (known as bag-describe.py, see: https://github.com/MetaArchive/bag-describe), which can be activated to create per-file technical metadata for the Bag's contents if a program so chooses. Support for use of ClamAv is also provided. See *3.2.1.2* for links to the configuration documentation to enable these services.

ETD Drop can also be programmatically configured to work with other separately scripted programs and installed services. Such implementations will be dependent upon other environment-specific configurations that cannot be predicted up-front, but will be straightforward for library/campus IT units to accomplish if the use case is desired and brought to their attention.

3.2.1.1 Obtaining & Installing ETD Drop

Installing ETD Drop will require the assistance and hosting support services of your ETD program's library/campus IT services. Review the use case documentation and talk with your technical support about installation.

To download ETD Drop from the MetaArchive Cooperative's GitHub page, see:
https://github.com/MetaArchive/etd-drop.

To read the documentation available on Read the Docs, see:
http://etd-drop.readthedocs.org/en/latest/.

3.2.1.2 Starter Instructions for Using ETD Drop

To read the documentation for Configuration, User Management, Suggested Workflow, and Submitting an ETD on Read the Docs, see:
http://etd-drop.readthedocs.org/en/latest/configuration.html,
http://etd-drop.readthedocs.org/en/latest/user_management.html,
http://etd-drop.readthedocs.org/en/latest/suggested_workflow.html,
http://etd-drop.readthedocs.org/en/latest/submitting_an_etd.html.

3.2.1.3 What to Do With a Submission

To read the documentation for Managing Submissions on Read the Docs, see: http://etd-drop.readthedocs.org/en/latest/managing_submissions.html.

4 Event Record-Keeping for ETDs

4.1 Rationale

There can be several preservation events that can take place in the life of an ETD. Preservation events can include *Virus Checking for ETDs* and *Format Recognition for ETDs* as described above. Other preservation events can also include ingest into an institutional repository, proper replication archiving according to institutional policies. The Library of Congress maintains a helpful controlled vocabulary of numerous preservation event types that can apply to ETDs, as well as any other type of digital content, see: http://id.loc.gov/vocabulary/preservation/eventType.html).

ETD programs, where possible, should begin consistently tracking the success or failure outcomes of as many preservation events as apply. Having an easily retrievable record of any failure outcomes, for example, can help ETD programs target and optimize their responses and work toward ensuring preservation events are ultimately successful.

4.2 Event Record-Keeping Use Case for ETDs

ETD programs can record the success or failure of various preservation events using PREMIS (see: http://www.loc.gov/standards/premis/). Creating and recording PREMIS event records for ETDs should be carried out through a careful planning effort involving the full range of ETD program stakeholders, but particularly the library and its technical services. PREMIS is designed to associate an Event (virus checking, format recognition, ingest, replication, etc.) with a corresponding Agent (software, human action, etc.) and to tie the Agent and the Event outcome (success/failure) to an individual digital object (e.g., ETD pdf). The Agents that are applied to an object can have their outputs directly parsed with *scripts* or manually entered into files or databases that can be accessed programmatically. The outputs are then recorded as PREMIS Event records.

The Lifecycle Management of ETDs project has endeavored to make the creation and management of PREMIS Event records easy by providing a PREMIS Event Service. The PREMIS Event Service provides a straightforward way to send PREMIS-formatted events to a central location to be stored and retrieved. In this fashion, it can serve as an event logger for any number of services that use it. PREMIS was chosen as the underlying format for events due to its widespread use in digital libraries.

4.2.1 Use of PREMIS Event Service

The PREMIS Event Service is a Python Django application for managing PREMIS Events in a structured, centralized, and searchable manner. For example, PREMIS Event Service can record events in an ETD workflow where a program transfers ETDs from a submission system such as Vireo or ETD Drop (see 3.2.1) to archival storage. The program could generate PREMIS Event XML based upon the success or failure of the transfer along with any error details that the program was designed to understand (e.g., ingest failed due to lost network connection, etc.). The PREMIS Event XML would then be wrapped in AtomPub and posted to the PREMIS Event Service, where the graduate school or library can easily query the Service for failed transfers.

This is just one of many examples. A similar series of events could be logged for ETD objects that subsequently failed to achieve a proper number of backup replications.

To read the Library of Congress controlled vocabulary for a more complete list of event types, see: http://id.loc.gov/vocabulary/preservation/eventType.html.

To read an additional PREMIS Event Service usage example on Read the Docs, see: http://premis-event-service.readthedocs.org/en/latest/api.html#example.

4.2.1.1 Obtaining & Installing the PREMIS Event Service

Installing the PREMIS Event Service will require the assistance and hosting support services of your ETD program's IT staff. Review the use case documentation and talk with your technical support about an installation. Creation and use of PREMIS metadata generally should also involve close consultations with the ETD program's library contact. They may be using PREMIS for various preservation efforts and can advise on conformance issues and use case implementations. They may also have a broader use case interest for the PREMIS Event Service.

To download the PREMIS Event Service from UNT Libraries' GitHub page, see: https://github.com/unt-libraries/django-premis-event-service.

To read the full documentation for the PREMIS Event Service on Read the Docs, see: http://premis-event-service.readthedocs.org/en/latest.

4.2.1.2 Starter Instructions for Using the PREMIS Event Service

To read the usage documentation on Read the Docs, see: http://premis-event-service.readthedocs.org/en/latest/usage.html.

4.2.1.3 What to Do With the Event Records

Every institution will respond differently to various event outcomes. It is not within the parameters of the PREMIS Event Service to handle the response or resolution of any events, regardless of their success or failure. That being said, the PREMIS Event Service is designed to be helpful in calling attention to the objects or processes that may need further inspection or troubleshooting. Once Events are logged, the PREMIS Event Service highlights specific Events, Agents, and Objects needing action (according to institutional policies and procedures). Each of the records logged with the PREMIS Event Service are intended to be unique and canonical for that Event.

To read more information on usage see the Read the Docs: http://premis-event-service.readthedocs.org/en/latest/usage.html.

5 Reference Link Archiving

5.1 Rationale

The Lifecycle Management of ETDs project is proud to collaborate with the Hiberlink project (http://hiberlink.org/) to promote adoption of that initiative's solutions to solve the 'reference rot' problem. Reference Rot occurs whenever the original version of a linked resource is not available any more – a growing problem given that today's web-based scholarly communication includes links to an increasingly vast range of online materials, including software, datasets, websites, presentations, blogs, videos, scientific workflows and ontologies. These resources pose a significant challenge. Unlike traditional scholarly articles, the content referenced by any given http:// link is liable to change over time. The issue is two-fold: a link may no longer work or the content referenced has dramatically changed from what it was originally. As a result, what is online at the time of citation is less likely to be there when scholars wish to look up the citation.

An important part of the Hiberlink project has been to draw reference rot to the attention of the scholarly community, having carried out large-scale analysis of both journal articles and ETDs and demonstrated significant loss due to reference rot. Hiberlink also aims to identify practical solutions. These solutions are aimed at pro-active archiving. The first, for authors of scholarly works, including theses and dissertations, is geared towards helping avoid reference rot. This is achieved by triggering the archiving of what has (just) been consulted. There is also work-in-progress, code-named HiberActive, to support pro-active archiving by repository managers as part of the repository ingest workflow, with the aim of helping 'stop the rot' for references made in the (later) submissions into a repository.

These solutions dovetail nicely with an enhanced approach to reference resources, described as the Missing Link proposal. That describes how to reference a URI with increased persistence: it keeps the original link to the online resource and adds attributes that define the temporal context, such as time/date of referencing and the URI of the version of the resource that was archived. Inclusion of versiondate and versionurl attributes in the anchor (<a>) element can be readily leveraged by the Memento protocol, which enables search of archives for earlier versions of web resources. Further information on this is found on the Memento website, especially http://www.mementoweb.org/missing-link/. All other progress and technical information can be found in links on Hiberlink.org including presentations made about HiberActive (e.g., http://www.slideshare.net/martinklein0815/hiberactive).

5.2 Hiberlink Use Case for ETDs

As of this publication, the Hiberlink project has prototyped a Zotero plugin (http://hiberlink.org/developments.html) that will assist authors in proactively archiving at-risk digital content. The Hiberlink plugin creates a backup of the reference with an archival service and keeps a record of this backup. Should the original resource change or disappear the plugin allows you to view the content as it was when you referenced it. The archived references can be exported and used to refer to the original resource in other publications. This Hiberlink plugin is a resource that ETD programs can

encourage their campus researchers and student authors to make use of during the course of their ETD production cycle.

5.2.1 Use of Hiberlink Zotero Plugin

A demonstration of the prototype Hiberlink plugin is available on YouTube (https://youtube.googleapis.com/v/ZYmi_Ydr65M&vq=hd1080). The Hiberlink project is currently encouraging ETD programs to contact them to assist with an implementation. See the contact info below:

Email: edina@ed.ac.uk

Subject: Hiberlink ETD

6 Glossary of Technical Terms

API (Application Programming Interface): The mechanism provided by software to allow communication with other software.

Arguments: Data expected by a command-line program (options, file names, etc.). Provided after the program name before executing a command.

Command-Line Console: A computer's mechanism for running text commands. Sometimes called a "Terminal" or "Command Prompt".

Command-Line Interface (CLI): Refers to the use of software through text-based (non-graphical) input and output.

Command-Line Options: Aspects of a software's operation that the user is able to change.

FTP (File Transfer Protocol): A standard for exchanging files between computers. Programs known as "FTP clients" use this protocol to make file transfers with "FTP servers". A newer version of the protocol called SFTP adds encryption for security.

Graphical User Interface (GUI): Refers to the use of software through graphical means instead of purely text. All software typically used on personal computers, tablets, and mobile phones use Graphical User Interfaces.

Platform(s): A particular operating system used on a computer. Different platforms are able to run different kinds of software. Windows, Linux, OSX, and Android are often referred to as platforms.

Plugin Architecture: Refers to a way of building software that allows third parties to add custom pieces of software that "plug in" to the main software in some way, allowing for the software to be extended with additional functionality or modifications.

Scripts: Text files made up of programming commands used for executing a series of tasks in a repeatable way.