Macintosh Resource Forks

Choosing File Formats for Preservation

|  |
| --- |
| **Tyler Thorsted** |
| *Church of Jesus Christ of Latter-day Saints*  *Church History Library*  *USA*  [*thorsted@churchofjesuschrist.org*](mailto:thorsted@churchofjesuschrist.org)  0000-0003-0292-0962 |

|  |
| --- |
|  |

# **Abstract - The preservation of files from early Macintosh Classic (OS <=9) may often require special handling in order to ensure long term preservation and rendering. The classic Macintosh operating system would use two “forks”, a data fork and a resource fork. Resource Forks may contain graphics, sounds, fonts, and additional code. In addition, the file system would store two 4 digit codes for each file, one identifying the creating software and another identifying the type of file as extensions were rarely used. Because of this unique information within the Macintosh file system, most modern preservation systems are only aware of the data fork and information can be lost. Round-tripping a file through a preservation system and back to the original OS can help identify potential loss.**

**Keywords – Macintosh, Resource Fork, HFS, Risk, Finder**

**Conference Topics – Exchange; Resilience**

# Introduction

All documents and files created on a Macintosh between the 1980-90’s will have additional data which is important to proper functionality and rendering. Not all files used a resource fork to store some or all of its data, but many did. This paper will take an early project format from the popular audio recording application Pro Tools to illustrate different methods of preservation of files with these attributes.

DigiDesign Pro Tools [1] was an early digital audio workstation for recording audio. The software would create a folder structure for each session which included the session file, a folder of audio clips in the SoundDesigner 2 (SD2) file format [2], and additional folder of fades also in the SD2 format. The session file and the SD2 files each used a resource for part or all of their data.

A data set was created [3] on an original Macintosh running operating system 7.5. Pro Tools version 3.4 was used to create four different sessions, with and without linked SD2 audio files.

# File Formats

## Data Set

The two main file formats in this data set are the Pro Tools version 3 project file and native audio format used by Pro Tools, SoundDesigner 2. Pro Tools session files are saved preserving all the parameters of the recording project. What made the Pro Tools session format unique is that all the parameters were stored in the resource fork of the file with nothing in the data fork (see Figure 2). For the Sound Designer II files, the raw data was stored in the data fork, but all the information on sample rate, duration, channels were stored in the resource fork. Once development was made to support Mac OS X in version 6, all of this changed.

* 1. Preservation Formats

For Preservation purposes, the following formats were chosen to test preservation processes.

* Disk Image
* Original Logical Copy
* Stuffit
* MacBinary
* AppleSingle
* AppleDouble

# Ingest

Each format was added to a folder and ingested in our preservation system. This ingest includes file identification using DROID and scanned for viruses. Validation depends on tools available per format and identification, our system does not have anything in place for these formats.

## Disk Image

The disk image was identified and ingested into our system. As seen in the chart below, the disk image was able to store all the original data, but only the disk image was visible to the preservation system, not the individual files on the disk.

## Logical Copy

A direct copy from the disk image was made and added to a ZIP file for ingest. The ZIP container included the AppleDouble “.\_” resource forks and were extracted during ingest. AppleDouble files were identified, but Pro Tools project files failed as they contained zero bytes. With no extensions, the original files produced an error. Custom folder icons also caused a failure during ingest because of illegal characters in filename.

## Stuffit Container

Adding the contents of the disk image to a Stuffit container made a single “.sit” file which was identified by DROID. Similar to the disk image, the individual files were not visible to the preservation system. One method is making individual files using Stuffit, adding info in the filename. A Stuffit file defaults to compression of the file, but this feature can be turned off, which may reduce risk.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Format | Resource Fork | Type/Creator | Individual Files Visible | Identification | Validation | Compression | Common Software | Additional Steps |
| Disk Image | X | X |  | X |  |  | X |  |
| Logical Copy |  |  | X |  |  |  |  |  |
| Stuffit | X | X |  | X |  | X | X | X |
| MacBinary | X | X | X |  |  |  | X | X |
| AppleSingle | X | X | X | X |  |  |  | X |
| AppleDouble | X | X | X | X |  |  |  | X |

Figure 1. Table showing each Preservation format method and related properties.

## MacBinary

MacBinary has been around a long time and is commonly used by Macintosh users. Encoding to MacBinary adds the resource fork, data fork and Finder information into a single file with a “.bin” extension. Many software titles exist to encode and decode into this format. This file format is not currently identified by DROID and “.bin” extension is common among other formats. Versions 1 & 2 of MacBinary don’t have static headers, making a PRONOM signature difficult. Each file in the data set can be encoded, even the custom icon for one of the folders. The format maintains original filename, even if encoded filename is changed, and also retains original timestamps.

## Apple Single

Apple Single has all the same benefits of MacBinary, but was not as popular so software titles are more limited. The format is identified by DROID, making ingest easier.

### Apple Double

Apple Double was ingested alongside the original logical copy having the prefix “.\_” for all files. Also identified by DROID, exporting and moving back to original system is cumbersome.

Text

Description automatically generated with low confidence

Figure 2. Information view of Pro Tools 3 session.

# Round Trip

Active digital preservation teaches us these ingest processes are “the actions required to maintain access to digital materials beyond the limits of media failure or technological change” [4]. Part of this process includes ensuring the files can be rendered correctly into the future.

The files ingested into the preservation system were then exported and copied back to a Macintosh for evaluation. The disk image and Stuffit file opened as expected and all content was retained. The MacBinary and AppleSingle files only required decoding using Mac OS X built-in tools [5] or a classic version of Stuffit Expander for MacBinary and a current version of Stuffit Expander for AppleSingle.

The original copied files had the most issues when trying to ingest into our preservation system, often failing entirely. A couple folders have a custom icon, which is a hidden resource fork only file with the name “Icon”, but because of an additional character at the end of the filename, errors occurred even before identification.

# Conclusion

A typical digital preservation workflow might include capture, extraction, organization, identification, validation, metadata extraction, and storage. Most preservation systems use file systems other than HFS, making this workflow and long-term preservation of this additional information challenging. Once the data from a HFS disk has been captured, determining the best processing and submission format is crucial to successful preservation.

There are many different digital preservation systems available and all may handle files differently. Plus, different preservation policies may influence decisions of preservation format. During our testing, our preservation system could not ingest files with no data fork and no extension, but when tested with a different system, these files ingest with no issues. This is not necessarily good, as it may not alert us to potential issues.

Many may choose to ingest a disk image, as it requires the least amount of effort to ensure good preservation. If this method is chosen, it might be well to add DFXML or something similar to capture the contents of the disk image into the metadata [6].

AppleDouble is often created automatically and can be preserved fairly easily, and if handled correctly, the split data fork and resource fork can be recombined in the original system. They still may need additional steps to put things back together. One downside for file formats such as Pro Tools sessions, the Apple Double file may be preserved easily, but since the session has zero bytes in the data fork, many systems will not allow the files to be preserved. This is where AppleSingle may be a better choice.

MacBinary is very easy to work with in Classic and Modern Macintosh systems and would be my choice, but currently is not identifiable by DROID.

For our purposes, we feel the best solution is a combination of a disk image along with a MacBinary/AppleSingle.

# REFERENCES

1. Wikipedia page about Pro Tools. <https://bit.ly/3MvcJqT>
2. File Format Wiki page about Sound Designer II files. <https://bit.ly/35vkKM4>
3. Google Drive containing test files. <https://bit.ly/3hN55Ks>
4. DPC Digital Preservation Handbook. <https://bit.ly/3Cml0IX>
5. MacBinary and AppleSingle tool descriptions. <https://bit.ly/37a7QU6>
6. DANNING. <https://bit.ly/3tG1ZgZ>