

# NLM-DEVELOPED SOFTWARE AS CULTURAL HERITAGE



NICOLE CONTAXIS

A NATIONAL DIGITAL STEWARDSHIP RESIDENCY (NDSR) PROJECT

# NLM-Developed Software as Cultural Heritage

An National Digital Stewardship Residency (NDSR) Project

*Nicole Contaxis, June 2015 – June 2016*

Contact: [ncontaxis@gmail.com](mailto:ncontaxis@gmail.com)

## Table of Contents

NDSR Work Plan Framework.....	2
The Current State of Software Preservation.....	4
Criteria Document.....	22
Curation Process Report.....	27
Future Directions.....	51
Blog posts	
Inventorying Software Developed at the National Library of Medicine.....	57
The Astrophysics Source Code Library.....	61
MEDLARS I & GRACE: The Early Mainframe Experience.....	63
MEDLARS II: MEDLINE & Instantaneous Search.....	67
Grateful Med: Personal Computing and User-Friendly Design.....	71
Appendix A: Historical and Technical Components of Located Software.....	76
Appendix B: Potential Data Elements for Software Assets at NLM.....	80
Appendix C: Current Metadata for Software Assets.....	83
Appendix D: Screenshots of Software Asset Resource Page.....	83
Appendix E: Further Reading & Resources.....	84

# NDSR Work Plan Framework

An Outline of the Goals and Schedule of the NDSR 2015-2016 Project at NLM

## Major Project Goals

### *Overall Description:*

The Resident will create a pilot workflow for the curation, preservation, and presentation of a historically valuable software product, developed by the National Library of Medicine (NLM), which is deemed to be historically noteworthy due to its usage by a user community and/or its distinctive technical properties that are at risk of being lost due to obsolescence. The Resident will be involved in all stages of the process of inventory, selection, curation, preparation, and ingest of software files, and creation of a public online presentation of the software that would provide its historical context.

### *Specific Goals:*

- (1) Create an inventory of software developed across the NLM Divisions
- (2) Conduct interviews with key staff members about important pieces of software and the process of developing them
- (3) Create an environmental scan of software preservation practices and projects across external organizations
- (4) Establish a pilot preservation program with appropriate preservation techniques informed by best practices
- (5) Create a presentation for the pilot preservation program designed for outside audiences
- (6) Work with NLM's Digital Repository group to prepare and ingest any appropriate or viable files

## Overall Schedule

- Meetings with mentor:
  - o Weekly meetings to my progress, new obstacles, and other aspects of the project
- Other organizational meetings:
  - o Quarterly meetings with the Project Support Committee
  - o Individual meetings with internal experts & long-term employees
- Outside meetings:
  - o Attempt to schedule meetings with external experts:
- Other:
  - o Bi-Weekly NDSR cohort meetings to discuss out projects, the Symposium, and the Enrichment sessions
  - o Conferences and workshops including:
    - 12<sup>th</sup> International Conference on Digital Preservation (iPRES 2015)
    - AIC's TechFocus III: Caring for Software-Based Art
    - Mid-Atlantic Regional Archives Conference (MARAC 2016)

## Overall Timeline

- Months 1-4
  - o Tasks:
    1. Survey other cultural organizations and working groups to identify software selection criteria and software preservation best practices
    2. Survey and document NLM's software development
  - o Milestones:
    1. Create inventory of software developed at NLM
    2. Perform and document an environmental scan of software preservation activities at outside organizations
- Months 5-8
  - o Tasks:
    1. Conduct interviews with NLM staff involved with the design, development, management, and marketing of the software along with users of the software
    2. Develop a draft workflow, including appraisal and description of the artifacts
  - o Milestones:
    1. Choose one particular piece of software and begin to focus our efforts on contextualizing and preserving it
    2. Complete documentation describing how and why this piece of software was chosen
    3. Write a narrative history of the chosen software artifact
- Months 9-12
  - o Tasks:
    1. Prepare and ingest the software (e.g., the binaries) into the NLM repository if deemed appropriate
    2. Develop a public-facing web presentation, providing the historical context in textual or audiovisual form, and if feasible, an interactive simulation
    3. Present the findings and demonstrate the project outputs to NLM staff
    4. Consider recommendations to develop the NLM's software preservation program past the duration of the residency
  - o Milestones:
    1. Create a final report summarizing the lessons of the overall project and providing recommendations for future software preservation activities
    2. Present the findings and demonstrate the project outputs to NLM staff and possibly the public via online and in-person interactions
    3. Document recommendations for further development of a software preservation program at NLM

# The Current State of Software Preservation

In-Depth Exploration of Software Preservation as Practiced in Fall 2015

## Executive Summary

Software preservation is an important part of the stewardship of cultural, scientific and social history. This report outlines several ongoing projects and related preservation strategies. While many of these strategies include preserving software as an interactive digital object, strategies that simply document the software, like recording game play in online video games, are also included. The paper begins by outlining key tools and concepts and over-arching obstacles to software preservation, including a lack of contemporaneous documentation, the wide variety of use cases for software, and copyright legislation. Afterwards it discusses three categories of software preservation projects: software collections in libraries and archives, digital art conservation, and computational reproducibility. With in-depth discussions of the benefits and issues with many of these projects, the paper outlines how different strategies best serve different types of software, institutions, and budgets. Both technical and administrative strategies are considered. Projects addressed include the National Software Reference Library (NSRL), the Olive Archive, Emulation as a Service, the work at Rhizome, the Game Play Capture Project at the Strong Museum of Play, the Astrophysics Code Library (ASCL), and others.

## Introduction

Preserving software is a vital aspect of the stewardship of cultural, scientific, and social history. The 2015 National Agenda for Digital Stewardship calls for serious action in the realm of software preservation. It states, “[Software] is both the key to accessing and making sense of digital objects and an increasingly important historical artifact in its own right.”<sup>1</sup> This document argues that preserving software is necessary for two reasons: (1) it is a cultural artifact worthy of historic study in its own right; and (2) it is a means to access data and content held in obsolete formats. Ensuring long-term access to executable products is necessary in order to preserve the content created, managed, and accessed through software as well as to preserve the history inherent in the creative endeavor of software development itself. The digital stewardship community cannot afford to ignore executable files.

The impetus for this paper is a project at the National Library of Medicine (NLM) called “NLM-Developed Software as Cultural Heritage.” This is a National Digital Stewardship Residency (NDSR) project, funded by the Library of Congress (LC) and the Institute of Museum and Library Services (IMLS), which aims to better understand the history of software development at NLM and to create a preservation strategy for at least one of NLM’s software products.<sup>2</sup> NLM has a long history of developing software, both for internal needs and for its users. This overarching goal of the NDSR project is to preserve this history for future use.

---

<sup>1</sup> “2015 National Agenda for Digital Stewardship.” National Digital Stewardship Alliance. September 1, 2014. Accessed November 12, 2015.

<http://www.digitalpreservation.gov/ndsa/documents/2015NationalAgenda.pdf>

<sup>2</sup> “NDSR Project: NLM-Developed Software as Cultural Heritage.” National Digital Stewardship Residency. December 1, 2014. Accessed November 12, 2015. [http://www.digitalpreservation.gov/ndsr/NLM\\_NDSR\\_NLM-Developed\\_Software\\_as\\_Cultural\\_Heritage-1.pdf](http://www.digitalpreservation.gov/ndsr/NLM_NDSR_NLM-Developed_Software_as_Cultural_Heritage-1.pdf)

Examples of such software include an internal cataloging and acquisitions system, conceived in the late 1970s and created because contemporary software vendors could not meet the library's needs. Only in the late 1990s did the library begin to use a vendor's cataloging software, with serious adaptations to accommodate the library's unique holdings. MEDLARS, another example of NLM-developed software that began in 1961, was a pioneering project that provided public access to computerized library records, pre-Internet or other networked technologies.

This project is divided into two components: (1) understand of the history of software development and implementation at NLM; and (2) the creation of a preservation strategy for at least one piece of software. This latter aspect of the project necessitates in-depth research into the strategies being implemented at other institutions, which is reflected in this paper. As will be established later, there is no model framework for software preservation since the needs of the institutions, the functionality of the software, and the current state of the software can all vary wildly, but this paper hopes to provide guidance for institutions as they consider how to preserve software for their own needs and the needs of their patrons.

An extreme illustration of the need for software preservation was featured in a post on *The Signal*, the Library of Congress's digital preservation blog in an interview with Doug White, head of the National Software Reference Library (NSRL). A medical supply company had distributed a shipment of Botox that had been improperly produced, and although the FDA had all of the information necessary to perform a recall, the information was formatted for obsolete software. Luckily, the NSRL possessed a copy of the software so that the data could be read and the FDA could recall the shipment. In this instance, software preservation proved life-saving.<sup>3</sup> Less sensational examples include government emails that are created and stored in obsolete formats.<sup>4</sup> If a copy of the original software does not exist, accessing those emails may be expensive or even impossible. Preserving software, in this way, is necessary to ensure government transparency in the digital age. Information access is dependent on software preservation strategies.

While preserving software allows one to access data held in obsolete formats, it also allows one to view and understand the processes by which that data is created. In STEM fields, it is necessary to be able to reproduce findings, but as more STEM fields come to rely on computational methodologies, it may be necessary to preserve the code that performs that computation. For example, if a computational biologist runs a certain algorithm on a set of data in order to produce her results, it is necessary to preserve that algorithm as part of her larger methodology.<sup>5</sup> Although this type of research is more prevalent in STEM fields, it is also used in the humanities and social sciences with the growing popularity of digital humanities and big data in the social sciences.

---

<sup>3</sup> Owens, Trevor. "Life-Saving: The National Software Reference Library." Life-Saving: The National Software Reference Library. May 4, 2012. Accessed November 12, 2015.

<http://blogs.loc.gov/digitalpreservation/2012/05/life-saving-the-national-software-reference-library/>

<sup>4</sup> Cush, Andy. "Texas Town Is Charging Us \$79,000 for Emails About Pool Party Abuse Cop." Gawker. June 29, 2015. Accessed November 12, 2015. [http://gawker.com/texas-city-is-charging-us-79-000-for-emails-about-pool-1714757746?trending\\_test\\_d&utm\\_expid=66866090-62.YkETBclMTk2uX1oytHipyg.4](http://gawker.com/texas-city-is-charging-us-79-000-for-emails-about-pool-1714757746?trending_test_d&utm_expid=66866090-62.YkETBclMTk2uX1oytHipyg.4)

<sup>5</sup> Di Cosmo, Roberto. "Keynote - Preserving Software: Challenges and Opportunities for Reproducibility of Science and Technology." SciLab: Open Source Software for Numerical Computation. May 21, 2015. Accessed November 12, 2015. <https://www.scilab.org/community/scilabtec/2015/Keynote-Preserving-Software-challenges-and-opportunities-for-reproducibility-of-Science-and-Technology>

Software also constitutes a creative enterprise in its own right and will be useful to future political, economic, and cultural historians. Matthew Kirschenbaum, for example, writes about the history of word processing and its cultural importance in his book, *Track Changes: A Literary History of Word Processing*, to be published later this year.<sup>6</sup> In discussions of this topic, he talks about the adoption of word processing software, its depiction in popular culture, and how the word processor itself affected the creation of text and literature throughout the late 20<sup>th</sup> century. Stephen King, for example, was one of the early adapters of the word processor, and its use appears in some of his lesser-known works like the short story, "Word Processor of the Gods." While King's thoughts are important because he is a cultural figure, word processors have also changed how day-to-day activities and everyday communication now takes place. If these pieces of software are not saved, how will our future historians adequately interpret our actions, our practices, and our creations? Concerns about preserving video games, as recently acknowledged by the Library of Congress with its additional copyright exceptions for the preservation and care of historic video games, are also a key part of preserving software as cultural heritage.<sup>7</sup>

In the face of this need, how should an institution or individual approach creating a software preservation strategy? Software preservation is not always necessary. For example, a library would be unlikely to emulate Word 5.0 in order to read documents saved in that format. However, it may be necessary to emulate Word 5.0 in order to migrate those documents into new formats if they were not migrated in a timely fashion.<sup>8</sup> Furthermore, as stated earlier, it is also necessary to preserve software in order to understand how that data was created. Preserving software is necessary in order to preserve the research methodology of many burgeoning and emerging fields. In conjunction with the cultural importance of software, it is clear that it is necessary to preserve software, but considering the variety of use-cases, it is not always clear how preservation should be accomplished.

This lack of clarity exists for several reasons: (1) the ubiquity of software; (2) the complexity of software and its dependencies; and (3) the variety of software designs and applications. Software is present in almost every aspect of modern life, and due to the wide variety of use cases, it can be difficult to pin down the exact context and infrastructures that allow a particular piece of software to communicate meaning. In other words, when software encompasses everything from the algorithms for computational chemistry to the video game, *Street Fighter 2*, the preservationist needs to educate herself on the particularities of that specific piece of software in order to understand how best to preserve it.

Each piece of software has a myriad of dependencies that can be difficult to discern. Most software is dependent on specific hardware to operate properly, but code libraries can also present a serious

---

<sup>6</sup> Kirschenbaum, Matthew G. "Stephen King's WANG: A Literary History of Word Processing." Lecture, Stephen A. Schwarzman Building, Wachenheim Trustees Room, New York Public Library, New York City, December 16, 2011.

<sup>7</sup> Higgins, Parker, Mitch Stoltz, Kit Walsh, and Corynne McSheery. "Victory for Users: Librarian of Congress Renews and Expands Protections for Fair Uses." Electronic Frontier Foundation. October 27, 2015. Accessed November 12, 2015. <https://www.eff.org/deeplinks/2015/10/victory-users-librarian-congress-renews-and-expands-protections-fair-uses>

<sup>8</sup> Rosenthal, David. "Emulation and Virtualization as Preservation Strategies." [https://mellon.org/media/filer\\_public/0c/3e/0c3eee7d-4166-4ba6-a767-6b42e6a1c2a7/rosenthal-emulation-2015.pdf](https://mellon.org/media/filer_public/0c/3e/0c3eee7d-4166-4ba6-a767-6b42e6a1c2a7/rosenthal-emulation-2015.pdf)

obstacle for the software preservationist. A code library is a resource for software development that provides pre-written code for a piece of software to implement. However, if that code library is no longer available at the point of preservation, the piece of software that utilizes code from that library may no longer function in the way that it was intended or there may be limited documentation about the components taken from the code library.<sup>9</sup> The same can be said of pieces of software that pull data from APIs, which may supply vitally important data by pulling it from other sources but not storing it locally.<sup>10</sup> So, the preservationist needs to do more than educate herself on the software as a discrete object; she needs to investigate the legal and technical infrastructures that the software relies on in order to function and convey meaning. Because these dependencies are not always explicitly mentioned in documentation or described at a consumer level, this research process can be arduous.

In the face of the complexity and ubiquity of software, the tools and strategies for its preservation are varied and, in many ways, still forming. Depending on the type of software being preserved, the institution preserving the software, and the copyright holder, there are different tools that may prove useful. In order to outline what these tools and strategies are, this paper will discuss four concepts currently under scrutiny in the field: VMs, video games, digital art, and computational reproducibility. Before examining these subjects, however, it is helpful to discuss some of the universal obstacles to software preservation.

## Concepts and Tools

At this point, a quick explanation of two basic preservation strategies is necessary. An institution can choose to emulate software or migrate that software's capabilities. Emulation means re-creating a computing environment so that the software can run as originally designed while migration means preserving the behavior of a piece of software in a new context. Migration may not preserve the look and feel of a piece of software, but it can preserve the basic functionality, making it a popular strategy in computational reproducibility efforts in academic fields.<sup>11</sup> Due to recent developments in emulation, emulation is becoming more popular as a digital preservation strategy. Virtual machine projects, described more thoroughly below, present an appealing method for emulation that requires less technical expertise than earlier methods. Both migration and emulation rely on a variety of other tools in order to function.

Computer forensics offers several strategies for the preservationist. The first is a disk image. A disk image is a copy of an entire computing system. It allows a user to access hidden files, deleted files, and other aspects of a computing system that may not be available if they copied each files from a hard drive rather than creating a disk image.<sup>12</sup> Disk images will be mentioned in many of the projects highlighted below. There are a variety of tools for creating disk images including Guymager, an open

---

<sup>9</sup>"Code Libraries." Developer Network. Accessed November 12, 2015. [https://msdn.microsoft.com/en-us/library/kews042w\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/kews042w(v=vs.71).aspx)

<sup>10</sup> Giaretta, David. *Advanced Digital Preservation*. Berlin: Springer, 2011. 126.

<sup>11</sup> McDonough, Jerome, Robert Olendorf, Matthew Kirschenbaum, Kari Kraus, Doug Reside, Rachel Donahue, Andrew Phelps, Christopher Egert, Henry Lowood, and Susan Rojo. "Preserving Virtual Worlds Final Report." IDEALS @ Illinois. August 31, 2010. Accessed November 12, 2015. Page 6. <https://www.ideals.illinois.edu/handle/2142/17097>

<sup>12</sup> Ibid. Page 85.



The Archive employs EaaS because Flusser's materials were created on a rare software that is dependent on an obsolete system.<sup>16</sup> In order for patrons to view his materials, the Archive thus employs a VM so that the software can function. In this way, a user can run another computer, with a different operating system and unique content, on her own computer. With this approach, users can access cultural materials in the same manner that the original creator handled them and accessed them himself.

A similar project at the University of California, Los Angeles attempts to re-create the computing environment of writer Susan Sontag complete with her born-digital materials as part of a larger collection of her materials. Within the confines of the library, patrons can request a computer that contains all of Sontag's born-digital materials. This single IBM laptop contains the content of several of Sontag's computers, which were originally donated as two external hard drives. This IBM laptop employs a product called "Deep Freeze," which allows patrons to use the computer while restoring the laptop to its original status after it is rebooted.<sup>17</sup> In other words, regardless of what patrons do on the laptop, it always returns to its archival form. The laptop is not connected to the Internet, and the files cannot be accessed online in any way. Both the Flusser project and the Sontag project provide access to born-digital cultural heritage, but the Flusser Archive is able to provide wider access with its use of EaaS.

EaaS and the Olive Archive are currently the two most prominent virtual machine-based emulation projects. EaaS is a collaborative project from the University of Freiburg and several other partners. It aims to create legacy computing environments that people can customize to their specific needs and therefore enabling them to run other programs or environments that are dependent on those legacy systems.<sup>18</sup> One of the use cases supplied by EaaS is the conversion of obsolete Microsoft Office Word files to current formats.<sup>19</sup> A user can run the VM in order to access the original files and then migrate them to more standard, preservation friendly formats, so that they will be easier to access in the future. This sort of strategy can be quite helpful when a piece of software needs to be preserved just so that the data it reads can remain accessible. Preserving Word itself could be difficult because of copyright concern, but creating an environment in which older versions of Word can run may skirt some of these issues. Because EaaS is used, or is proposed to be used, in a variety of ways, many projects that employ EaaS will be highlighted throughout this paper. Since VMs are a flexible tool, it is important to discuss how they are implemented in different projects, not just that they are implemented.

The Olive Archive functions similarly but has been used in different circumstances. This project is housed at Carnegie Mellon University, and bills itself as an archive of VMs. Some of the machines in their collection include Netscape Navigator 1.12, NCSA Mosaic 1.0, DOOM for DOS, Microsoft Office 6.0, and

---

<sup>16</sup> "The Digital Heritage of Vilém Flusser." BwFLA: Legacy Environments at Your Fingertips. Accessed November 12, 2015. <http://bw-fla.uni-freiburg.de/demo-flusser.html>.

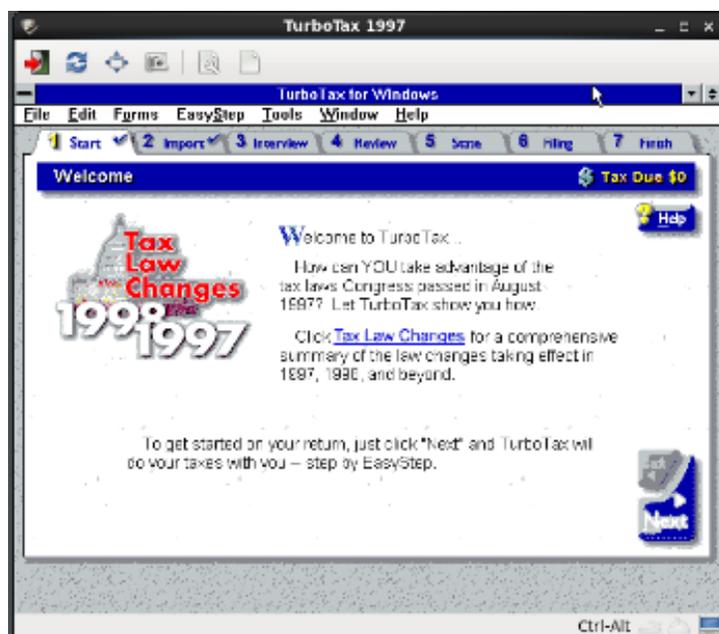
<sup>17</sup> Schmidt, Jeremy, and Jacquelyn Ardam. "On Excess: Susan Sontag's Born-Digital Archive." The Los Angeles Review of Books. October 26, 2014. Accessed November 12, 2015. <https://lareviewofbooks.org/essay/excess-susan-sontags-born-digital-archive/>.

<sup>18</sup> "Emulation as a Service." BwFLA: Legacy Environments at Your Fingertips. Accessed November 12, 2015. <http://bw-fla.uni-freiburg.de/>

<sup>19</sup> "Use Cases." BwFLA: Legacy Environments at Your Fingertips. Accessed November 12, 2015. <http://bw-fla.uni-freiburg.de/demos.html>

TurboTax 1997.<sup>20</sup> In effect, the Olive Archive has created these machines as pre-existing environments for users to implement on their own computers, having retrieved the disk image from the Olive Archive library. Rather than requiring users to have in-depth technical knowledge on the computing environment that they require, this allows them to implement an environment easily. While users need to download a client in order to run these environments, the interfaces retain their original user friendliness.<sup>21</sup>

These machines can be used to access data stored in obsolete formats or to view old and obsolete software in an emulation of its original computing environment. For example, the TurboTax1997 VM was created because of the unique historical value of TurboTax software. TurboTax is updated every year to reflect new tax laws, and political science students of the future could use these VM to better understand the individual ramifications of tax law change over time.<sup>22</sup> A screen shot of TurboTax 1997 being run through the Olive Archive, taken from the paper “Olive: Sustaining Executable Content Over Decades” by Satyanarayanan et al, is featured below.



VMs can be used in a variety of situations. As will be outlined later in this paper, VMs from both The Olive Archive and EaaS are implemented in several software preservation projects. Both EaaS and The Olive Archive provide a valuable service and are becoming increasingly important to the software preservation and digital stewardship community as more digital formats become obsolete.

<sup>20</sup> "Virtual Machines in Our Collection." Olive Archive. Accessed November 12, 2015. <https://olivearchive.org/docs/collection/>

<sup>21</sup> "Installing the Client." Olive Archive. Accessed November 12, 2015. <https://olivearchive.org/docs/vmnetx/install/>.

<sup>22</sup> Satyanarayanan, Mahadev, Gloriana St. Clair, Benjamin Gilbert, Jan Harkes, Dan Ryan, Erika Linke, and Keith Webster. "Olive: Sustaining Executable Content Over Decades." School of Computer Science at Carnegie Mellon University. November 1, 2014. Accessed November 12, 2015. Page 4. <http://www.cs.cmu.edu/~satya/docdir/CMU-CS-14-115.pdf>

## Universal Obstacles

Although there is significant variety in software, software development, and software preservation strategies, there are some universal obstacles that bridge these differences. These obstacles range from the theoretical to the practical and are generally deeply entangled. The decision of how to solve one problem directly affects the possible solutions to other problems. In this way, the process of preserving software necessitates a series of decisions that can have a domino effect on each other, and separating each decision can prove difficult.

The largest challenge for software preservation is deciding exactly what needs to be preserved. Due in part to the nature of executable files, there are several ways one can conceive of preservation and the assurance of long term access. One can consider it their obligation to preserve the code itself, the experience of using the software, or some combination of the two. Depending on how this issue is addressed, the preservation strategy will differ wildly. Choosing to migrate software may make sense for research software since they focus on preserving the functionality of the software.<sup>23</sup> Emulation, however, is more reasonable for projects that will want to provide users the opportunity to see how the interface functioned in historic software.<sup>24</sup>

It is clear that the way an institution defines what aspects of a piece of software to preserve will affect the other obstacles it faces. However, there are other obstacles that all institutions will contend with regardless of which aspects of a piece of software they prioritize in the preservation process. Versioning, for example, presents considerable practical problems. Software is generally released in many versions, and because of patches and updates, defining the exact parameters of a version of software can be difficult. Unlike a book which has clearly defined editions, the boundaries of software can be murky. Software, particularly after the advent of the Internet, can be updated bit by bit. With a patch here and a patch there, a single piece of software can undergo many changes without even being considered a new, official version.<sup>25</sup> While technology like Git now allow developers and others to better understand software versioning by tracking changes as they happen, it can still be difficult to trace which version should be preserved and where along that version's timeline it should be preserved.<sup>26</sup> As an interesting new phenomenon, Microsoft has abandoned the idea of versioning entirely and has called Windows 10 "the last version of Windows." Instead of providing concrete, new versions of the operating system, Microsoft will instead offer the operating system "as a service" that updates in real time with development changes.<sup>27</sup> While most development companies still employ versioning, if the practice of providing software as a service is to become a design trend, the opaque and fluid model may present even further difficulties for the preservationist.

---

<sup>23</sup> "Preserving.Exe: Toward a National Strategy for Software Preservation." October 1, 2013. Accessed November 12, 2015. Page 23.

<sup>24</sup> Ibid, Page 24.

<sup>25</sup> The Mac Operating System, for example, is updated frequently to deal with security risks and small changes. But these slight changes may not be considered when naming new versions of the operating system for public consumption.

<sup>26</sup> This opacity may not be purposeful. Frequently, due to the way that software is developed, the team themselves may not be clear about what updates were done when if not employing a tool like GitHub.

<sup>27</sup> Kelly, Gordon. "Why Microsoft Announced Windows 10 Is 'The Last Version of Windows'" Forbes / Tech. May 8, 2015. Accessed November 12, 2015. <http://www.forbes.com/sites/gordonkelly/2015/05/08/microsoft-windows-10-last-windows/>

Practical considerations can also inform strategy choices. When attempting to preserve software that is already long out of use, the code available and the condition it is in can force many decisions. Even if it becomes apparent that one version of the software was seminal within that software's lifetime, examples of that version may no longer exist. Although one may be able to locate documentation about the version, establish that it is the ideal version to preserve, and define where along the version's timeline to capture the information, it may still be impossible to preserve that version depending on how the copies of that version have been cared for. Finding the code itself may be the biggest issue for some projects.

Regardless of the eventual preservation strategy, there are universal issues for software preservation. Two of these are copyright and a lack of documentation. These obstacles are more tied to the history and philosophy of software development than intrinsic to the structure of the software itself. This distinction is important because it draws attention to the fact that these obstacles will differ slightly between institutions and may change radically over time. At the moment, these issues are universal to software preservation projects, but they may not be in the future.

Copyright exceptions that allow libraries to preserve materials do not exist for software in the same way as they do for books, prints or other analog materials. Libraries, for example, are legally able to digitize books for preservation; they do not need to worry about being sued for, technically, reproducing the book.<sup>28</sup> However, because the Anti-Circumvention Clause of the Digital Millennium Copyright Act (DMCA) makes it illegal for libraries to break Digital Rights Management (DRM) software, it can be difficult to even view the source code of some software in order to preserve it.<sup>29</sup> Although this paper will not address copyright issues directly, it is important for individuals or institutions involved in software preservation to research their copyright standing.<sup>30</sup> Copyright can present a formidable obstacle to software preservation.<sup>31</sup>

Copyright is less of an issue for certain projects. At NLM, for example, the software under consideration for the preservation aspect of this project was developed in-house and most federal government products cannot be copyrighted. It is possible that a piece of software may have been developed through a partnership with another institution or business, and that partner may have a stake in copyright ownership. However, we have yet to locate an example of this in NLM's collection of software. Other institutions, like the Internet Archive, may be more risk-tolerant and skirt copyright law by preserving software whose copyright holder no longer exists, termed 'abandonware' and to be discussed later. While this is not technically legal, it does act as a functioning workflow. Copyright may not be an overwhelming concern for either of these scenarios, but it is important to still keep copyright in mind as an aspect of risk management.

Researching copyright status can be quite difficult. In fact, finding any information about the development process can be difficult, particular when investigating an older piece of software. Without

---

<sup>28</sup> "17 U.S. Code § 108 - Limitations on Exclusive Rights: Reproduction by Libraries and Archives." Legal Information Institute. Accessed November 12, 2015. <https://www.law.cornell.edu/uscode/text/17/108>

<sup>29</sup> "17 U.S. Code § 1201 - Circumvention of Copyright Protection Systems." Legal Information Institute. Accessed November 12, 2015. <https://www.law.cornell.edu/uscode/text/17/1201>

<sup>30</sup> "Digital Millennium Copyright Act." Legal Information Institute. Accessed November 12, 2015. [https://www.law.cornell.edu/wex/digital\\_millennium\\_copyright\\_act](https://www.law.cornell.edu/wex/digital_millennium_copyright_act)

<sup>31</sup> "Preserving.exe." Page 28.

a robust records management plan, documentation of the development process is frequently lost, and tracking down relevant information can be quite time-consuming. While the amount of documentation needed will vary according to the exact preservation strategy implemented, having information about the development process can answer questions about copyright, creator intent, and user interaction with the software.

Knowledge-finding in an environment of frequently sparse documentation can be frustrating, but there are still options. At NLM, for example, there is a plethora of documentation for educating patrons on user-facing software. This documentation helps illustrate both how users interacted with the software and how developers understood user needs. This information can be helpful in choosing how to preserve software or deciding whether it requires preservation at all. As software development has changed since the early years of development, a deeper understanding of the need of documentation has manifested through tools like Git, and hopefully the issue of finding documentation will be less prevalent for software produced later.

The variety of preservation strategies currently being implemented deal with these obstacles in different ways and to varying degrees of success. It is important to keep in mind, however, that regardless of the differences for the strategy and type of software being preserved, there is a common ground and common concerns across these projects. Preserving software may be a wide and varied practice, but it aims to solve issues that will become more and more present as time goes on.

## Solutions

There are three main domains where ongoing projects are attempting to preserve software: (1) Software and computing special collections; (2) Digital art collections; and (3) Computational reproducibility efforts. Each of these domains reflects a different type of project currently underway regarding software preservation. These projects rely on varying intellectual histories in order to form and implement their strategies, including art conservation, librarianship, and data science. While these three domains are not discrete as concepts and practices overlap, they represent three pushes in software preservation currently.

Many of these issues and projects are mentioned in or stem from the Library of Congress report, *Preserving.exe: Toward a National Strategy for Software Preservation*. Reporting on the 2013 National Digital Information Infrastructure and Preservation Program (NDIIPP) summit of the same name, this document focuses on the need for software preservation, identifying valuable and at-risk software, and ways to ensure long-term access to executable files. Two of the goals of the summit were to encourage new partnerships to support software preservation and to advise the Library of Congress about next steps to ensure long-term access to software and the data under its care.<sup>32</sup> Similarly, NLM's software preservation project is a continuation of these goals and a first attempt at confronting these issues in the same way as many projects described below.

### *Software and Computing Collections*

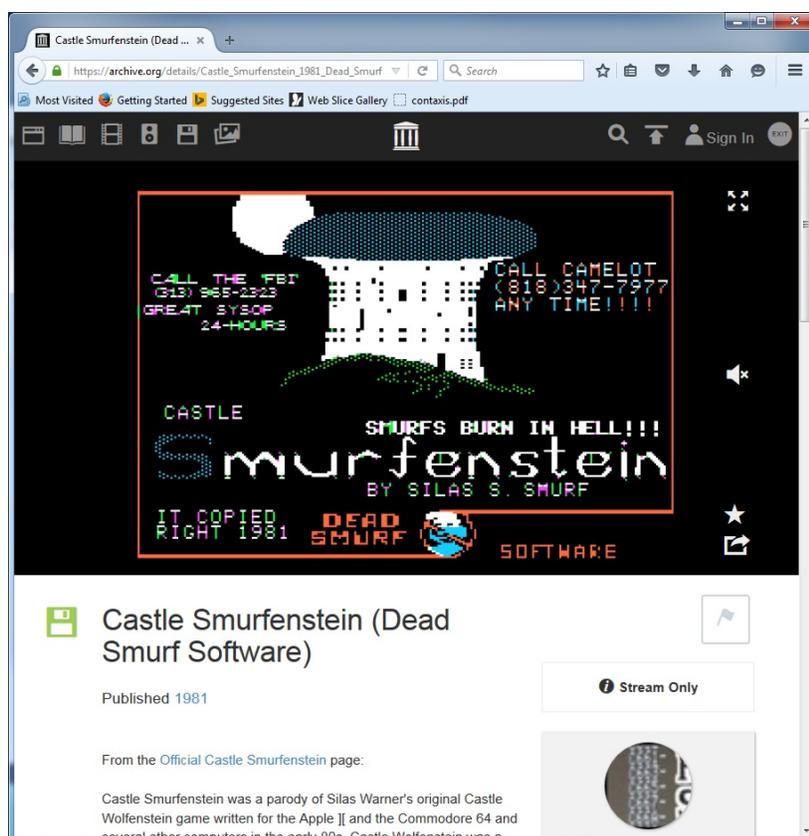
Materials about software and computing can be found in many libraries and archives. The Computer History Museum in Mountain View, California, holds large amounts of archival material and allow for some access to those collections online. These types of institutions need to preserve software in a way

---

<sup>32</sup> "Preserving.exe." Page 2.

that is cohesive with larger understandings of archival and library practice. In other words, these institutions have to preserve and provide access to software collections through the mechanisms, like catalogs, that they currently employ for a wide range of mediums and content formats. While individual practices vary between institutions, these libraries and archives have large collections and need to create access and preservation strategies with limited resources. In this approach, there is less of an interest in individual-object conservation and more of a focus on batch-preservation or mass-preservation strategies.

Perhaps the most well-known project for software and computing preservation is the Internet Arcade, which allows users to play old video games online through an emulation. However, the Internet Arcade is only a subset of a much larger collection of historic software being preserved by the Internet Archive. It currently holds 110,168 pieces of software in their online collections, including VisiCalc (1979), Apple DOS v3.3, and Castle Smurfenstein.<sup>33</sup>



Castle Smurfenstein, pictured above as an emulation in a modern computing environment, is an example of abandonware. Abandonware is software that is ignored by its owner or manufacturer, generally because that owner no longer exists. As a subset of orphan works, there is rarely a clear or enforcing copyright holder for abandonware. Most, if not all, of the Internet Archive software holdings are abandonware.<sup>34</sup> The Internet Archive is relatively risk-tolerant and posts software because, many

<sup>33</sup> Screen shot taken on September 28, 2015 from <https://archive.org/details/software>

<sup>34</sup> Scott, Jason. "Change Computer History Forever: Well, Here We Are." ASCII Text Files. April 13, 2013. Accessed November 12, 2015. <http://ascii.textfiles.com/archives/3947>

times, there is no existing entity likely to claim copyright ownership. Many libraries and archives are uncomfortable with this level of risk management, but it seems to work for the Internet Archive.

The technical implementation underlying the IA's approach is the JSMESS emulator, which ports the machine emulator MESS to Javascript.<sup>35</sup> JSMESS is open source and freely available on GitHub.<sup>36</sup> JSMESS runs in a contemporary browsers and only runs the requested software application. In other words, while EaaS is frequently used to run entire computing environments, JSMESS only runs the particular piece of software in question. Furthermore, unlike the Olive Archive, Internet Archive users employing the JSMESS emulator do not need to download a VPN or additional software in order to run it. In effect, a user can enjoy the JSMESS emulator without knowing what an emulation is or how it functions. Instead, they can simply enjoy playing Castle Smurfenstein.

The practices at the National Software Reference Library (NSRL) offer a stark comparison. As a National Institute of Standards and Technology (NIST) project, NSRL is a reference library, not a lending library. While people often come to them asking to borrow historic pieces of software, they do not lend software to outside agencies. The purpose of the National Software Reference Library is "to collect software from various sources and incorporate file profiles computed from this software into a Reference Data Set (RDS) of information."<sup>37</sup> Originally intended to assist federal, state, and local law enforcement with computer related crimes, encouraging and allowing widespread access to its materials is not the principal mission of NSRL. Because it generally functions as a reference library, it is primarily concerned with providing local access within their own computing environments. This lowers risk and the number of obstacles that they need to contend with.

The NSRL currently partners with the Stephen M. Cabrinety Collection at Stanford University Library. Both organizations are working to disk image large amounts of software. The Stephen M. Cabrinety Collection is a donated collection of materials that deal with the history of mini-computing and computing more generally.<sup>38</sup> While this process is ongoing, Henry Lowood, Curator for History of Science & Technology Collections and Film & Media Collections at the Stanford University Libraries, is considering a variety of other options, such as using the Olive Archive to run the historic software.<sup>39</sup> Of course, running the software and making it available and accessible are two separate considerations. However, in the case of the Cabrinety Collection, a VM seems like it will be integral to the future functioning of the archive, regardless of how accessible it will be.

---

<sup>35</sup> Scott, Jason. "Still Life, With Emulator: The JSMESS FAQ." Internet Archive Blogs. December 31, 2013. Accessed November 12, 2015. <https://blog.archive.org/2013/12/31/still-life-with-emulator-the-jsmess-faq/>

<sup>36</sup> "JSMESS." GitHub. Accessed November 12, 2015. <https://github.com/jsmess/jsmess>

<sup>37</sup> "National Software Reference Library." National Institute of Standards and Technology. Accessed November 12, 2015. <http://www.nsl.nist.gov/>

<sup>38</sup> "Cabrinety-NIST Project." Stanford University Libraries. Accessed November 12, 2015. <http://library.stanford.edu/projects/cabrinety-nist-project>.

<sup>39</sup> Guins, Raiford. *Game After: A Cultural Study of Video Game Afterlife*. Cambridge, Massachusetts: MIT Press, 2014. 292.

Owens, Trevor. "Video Game Preservation at Scale: An Interview with Henry Lowood." The Signal: Digital Preservation. February 1, 2013. Accessed November 12, 2015. <http://blogs.loc.gov/digitalpreservation/2013/02/video-game-preservation-at-scale-an-interview-with-henry-lowood/>.

Other organizations do not use emulation at all, and instead resort to a more simplistic preservation strategy. The Video Game Play Capture Project at The Strong National Museum of Play addresses preservation in a very different way. Rather than preserving the gaming software and providing continue access to the interactive aspects of the software, the Video Game Play Capture Project records videos of players actually playing the games using the original gaming consoles.<sup>40</sup> Clearly, this strategy has some limitations. It will not allow future users to interact with what was an interactive object. Users will only be able to view someone else's experience interacting with the software. However, this strategy has several benefits, both for the institution and possible future users. First, it removes many of the legal and technical preservation issues for interactive objects. There is a longer history of preserving a/v materials than there is of preserving interactive materials, and the institution will be able to call on this history to inform their own strategies. Second of all, the user may benefit from watching the original, intended users interact with the game.

Many of the games in the Video Game Play Capture Project are Massive Multiplayer Online Role-Playing Games (MMORPGs), and in order for them to function accurately, the player needs to interact with other Internet-connected players. For future historians to understand what it was like to play the game, it may be more effective to view game play rather than to play the games themselves. The social interactions on these games are historically important, and game play capture will allow them to better understand how users talked to each other and planned game action contemporaneously. This aspect of game play would be lost even if the game was perfectly preserved for future use. Additionally, future users will not need to learn how to play the game in order to enjoy it. These benefits, however, may not outweigh the loss of the actual digital object and its interactivity.

Perhaps because many of those spearheading these current attempts at software preservation are themselves video game and computing enthusiasts, there does not seem to be an extensive conversation about the issue of knowing how to use or interact with the software in the future, even if that software is accurately preserved. While game capture does not allow future users to play the game or interact with the software, it does avoid the issue of teaching future users the controls of a video game. My own experience trying to play games on the Internet Arcade illustrates how difficult it can be to teach oneself the controls without proper documentation, especially when a console-based game is being emulated on a desktop PC without all the same functionality as the original console.

Lastly, *Preserving Virtual Worlds* was a collaborative research project focused on how to preserve interactive media, particularly *Second Life*, a MMORPG. While the initial project was completed in 2008 and the subsequent one, *Preserving Virtual Worlds 2.0*, has also been completed, the partners were able to outline what specifically was at risk in video game preservation and offered early case studies of how to deal with video game archiving.<sup>41</sup> Much of their reporting is now outdated, however they did outline the uses of disk imaging, emulation, metadata, and hardware preservation that have been successfully employed in later projects.

Software metadata will not be discussed at length here, but it is an important aspect of software preservation since it allows for access and assists in identifying software, a current concern for

---

<sup>40</sup> "Video Game Play Capture Project." The Strong National Museum of Play. April 29, 2014. Accessed November 12, 2015. <http://www.museumofplay.org/about/icheg/play-capture>

<sup>41</sup> "Preserving Virtual Worlds Final Report."

organizations like NSRL.<sup>42</sup> However, because the uses for certain software applications can vary immensely, metadata approaches tend to vary according to the specific use case in question. For example, Jerome McDonough, from the *Preserving Virtual Worlds* project at the University of Illinois at Urbana Champaign, argues for using a combination of OAI-ORE and METS to best accommodate the metadata needs of game preservation.<sup>43</sup> However, Angela Dappert, writing on metadata for computing environments, argues that the best approach combines TIMBUS, an EU project on the long-term digital preservation of business processes and services, and PREMIS.<sup>44</sup> Because these two researchers are working on different kinds of software, they have isolated different priorities for the metadata preservation process. Creating standards, or even a set of best practices, for software preservation seems to be greatly impeded by the variety of uses and structures of software.

### *Digital Art Collections*

There is significant overlap between digital art collections and software and computing collections. Video games, for example, could fall into either of these categories, depending on how they are treated and understood. The game Pong may be included in a software collection, but it is also archived and exhibited at the Museum of Modern Art (MoMA). The distinction between these two arenas is less about the exact nature of the software in them and more about the way that the software is understood and handled. While software and computing collections work with large aggregations of materials in a mass production and preservation environment, digital art collections often work in the tradition of art conservatorship and spend more time and resources on the details of a particular piece.

Rhizome, a digital art museum partnered with the New Museum in New York City, bridges these approaches. With an extensive collection of digital art, Rhizome employs EaaS to create environments for digital and “net” art to function.<sup>45</sup> Rather than focusing on individual art pieces, they emphasize batch preservation, attempting to decrease the amount of work necessary to complete the preservation of their large collection. This preservation model, which employs disk imaging and emulation of computing environments, currently provides a reasonable working model for both digital art and software collections. It allows for continued interaction with the software in a historically accurate interface and also provides a larger sense of the software’s cultural and social context.<sup>46</sup>

Two other approaches to use as a comparison are the Internet Archive’s and San Francisco Museum of Modern Art’s (SFMOMA). The Internet Archive makes large amount of material available through emulation, but the documentation is limited and description may be at the collection-level rather than the item level. SFMOMA, on the other hand, deals with a single piece of digital art in depth, and

---

<sup>42</sup> Guttman, Barbara. "The National Software Reference Library." Digital Preservation at the Library of Congress. 2013. Accessed November 12, 2015.

[http://www.digitalpreservation.gov/meetings/documents/preservingsoftware2013/Presoft\\_Pres-Guttman.pdf](http://www.digitalpreservation.gov/meetings/documents/preservingsoftware2013/Presoft_Pres-Guttman.pdf)

<sup>43</sup> McDonough, Jerome. "A Tangled Web: Metadata and Problems in Game Preservation." In *Preserving Complex Digital Objects*, edited by Janet Delve. London: Facet, 2014.

<sup>44</sup> Dappert, Angela. "Metadata for Preserving Computing Environments." In *Preserving Complex Digital Objects*, edited by Janet Delve. London: Facet, 2014.

<sup>45</sup> Espenschied, Dragan. "Preserving 2000 Net Artworks: Practice Approaches to Object Identification, Risk Assessment and Permanent Access at Scale." Lecture, TechFocus III: Caring for Software-Based Art, Solomon R. Guggenheim Museum, New York, September 25, 2015.

<sup>46</sup> "EaaS: Image and Object Archive - Requirements, Implementation and Example Use-Cases." Open Preservation Foundation. Accessed November 12, 2015. <http://openpreservation.org/blog/2014/07/23/eas-image-and-object-archive-requirements-implementation-and-example-use-cases/>

describes its many dependencies with in-depth documentation. While a video game can be considered a piece of digital art, and there is a certain amount of creative effort in all software production, the distinction between these two approaches lies less in an innate quality of the preserved object itself than in the curatorial manner in which that object is conceived and handled.

SFMOMA has done more than simply preserve the pieces in their collection. They have been working to create a documentation methodology to assist other institutions. This documentation, called the 'Technical Narrative' was recently discussed by Mark Hellar as a means of preserving new media art. New media art is complex and may not only consist of software components. As Hellar stated in his talk, "These artworks exist as multiple components and a complex system of behaviors."<sup>47</sup> In other words, new media art can be a large system of parts and the art itself is the way that those pieces behave together. Preserving the software that assists in running these behaviors is an important part of the conservation requirements.

The Technical Narrative, as Hellar outlines it, acts as guidance for art conservators to document the various parts and behaviors that constitute new media art. Documentation is an important part of software preservation, and understanding the different components that constitute a single piece of software is an important step to adequate software preservation. Hellar separates proper documentation into four parts: (1) a high level description of how the work operates; (2) a modular examination of the components, what they do and how they relate; (3) a detailed description of the artwork as it exists upon acquisition; (4) and an analysis of the current technology and an evaluation of its longevity. Outlining documentation in this way allows the new media art conservator to be more confident that they have documented all relevant aspects of the artwork that may affect its longevity.

With the assistance of this documentation process, SFMOMA was able to recreate "Agent Ruby," an early example of AI that was created by Lynn Hershman Leeson and commissioned by SFMOMA in 1999.<sup>48</sup> The public can view and interact with "Agent Ruby" online.<sup>49</sup> In order to make this piece of art accessible online through modern browsers, SFMOMA needed to update code and discuss any updates with Leeson. Although the piece was not adequately documented in 2008 at the time of official acquisition, the documentation process outlined by Hellar shaped the process of re-creating and then conserving the "Agent Ruby" piece.<sup>50</sup> Effectively, this documentation process helps outline the necessary knowledge to conserving software-based art over long periods of time.

A similar project related to digital art and software-based art documentation is Source Code Documentation for Computational Art at the Museum of Modern Art. Headed by Deena Engel of the Department of Computer Science at New York University, this project aims to create documentation for software-based art by adapting the software engineering community's approach to documentation. Engel, similar to Hellar, suggests a three part documentation process. Her suggested documentation can

---

<sup>47</sup> Engel, Deena, and Mark Hellar. "Technical Narratives and Software-Based Artworks." Lecture, Technology Experiments in Art: Conserving Software-Based Artworks, National Portrait Gallery and Smithsonian American Art Museum, Donald W. Reynolds Center, Washington, DC, January 17, 2014.

<sup>48</sup> "Lynn Hershman Leeson, Agent Ruby, 1999-2002." SFMOMA. Accessed November 12, 2015. <https://www.sfmoma.org/artwork/2008.230>.

<sup>49</sup> Leeson, Lynn Hershman. "Agent Ruby." Accessed November 12, 2015. <http://agentruby.sfmoma.org/>

<sup>50</sup> Engel, Deena and Mark Hellar. "Technical narratives and Software-Based Artworks

be outlined as follows: (1) Documentation and annotation in the source code; (2) Narratives; and (3) Visuals.<sup>51</sup>

Because these pieces of documentation are built from software engineering standards, they require some explanation. In source code, it is possible to include comments in-line with the operational code. These comments are not intended for the machine to read. Instead, they are used for developers to talk to each other within the code base and as bread crumbs for future developers describing what each piece of code is meant to do. Engel suggests using a similar tactic with digital art so that documentation becomes an intrinsic part of the code. "Narratives," Engel's second descriptive element, are similar to Hellar's "high level description." Narratives provide background on the software, what it is intended to do, and how it is intended to do it. Lastly, "Visuals" are used during software development to help ensure adequate communication between developers, the design team, and other stakeholders about the architecture, look and feel, and intent of the software. Visuals can include workflows, wireframes, and other types of diagrams.

Hellar and Engel's documentation methodologies are designed for digital art projects. They are too detailed to efficiently work for projects that have large collections or need to ingest large amounts of information. Instead, these documentation models are ideal for working with intrinsically valuable artifacts. This is not to say that these model could not be useful in a different situation, but it would need to be adapted to reconcile the outcomes with the time and resources restraints of the institutions with large collections or wider collection strategies.

Digital art preservation efforts are not limited to documentation concerns. The Rose Goldsen Archive of New Media Art at Cornell University works with a wide variety of digital art that sometimes relies on software-based technology in order to properly operate. The manner in which these materials are preserved varies according to the particular needs of the art piece. Here, there seems to be no push for a single preservation strategy, but rather ones that are tailored to individual artworks or collections within the archive.<sup>52</sup>

Lastly, there are several institutions that research how humans interact with new media and digital art, and these institutions can provide helpful background for software preservation projects. Groups like "Time-Based Media and Digital Media," "Matters in Media Art" and "The Pericles Project" conduct research on how artists, museum-goers, and curators interact with new media art, particularly when the art is transmitted and created in new ways. Similarly, "The Trope Tanks" at MIT work extensively on humanist studies of computer use and can provide important background on the history of computing, the history of digital art, and sociological studies of computing and the human experience.

#### *Computation Reproducibility Efforts*

Software preservation is a large concern in the STEM fields as well as in cultural heritage institutions. STEM fields, however, are concerned with software preservation as a way to ensure that their research

---

<sup>51</sup> In her talk, Engel separates visuals and UML diagrams, although she admits that these two categories are closely related, if not the same. For the sake of this paper and for the sake of simplicity, I have combined these two categories since UML diagrams do not necessary serve a separate purpose than other visuals in the development process.

<sup>52</sup> "Rose Goldsen Archive of New Media Art." Cornell University Library. Accessed November 12, 2015. <http://goldsen.library.cornell.edu/>

can be re-computed in the future as a way to verify or support original findings. These motivations within the STEM field are largely concerned with ensuring access to data over long periods of time and being able to prove that data were created in proper ways. There are several projects that are addressing these issues in fields like computational chemistry, computational mathematics, and astrophysics.

Not every project deals explicitly with software preservation. The Astrophysics Source Code Library (ASCL), for example, acts as a repository more than as an archival or preservation project. Originally intended to help astrophysicists cite code that they use to complete the computations necessary for their scholarship, the ASCL has slowly developed into an institution interested in archiving the code as well as allowing scholars to share and cite the code they use. The manner in which they are archiving their code is still under debate. However, as an organization, ASCL recognizes the importance of archiving their code collection and is investigating how to best ensure that future astrophysicists will be able to access the code they need to in order to understand the scholarly data that they rely on.<sup>53</sup>

Similar projects in STEM fields concentrate on ensuring that all research projects can be re-created and validated in the future. Recomputation.org is working on ways to ensure that the work of computational mathematics and sciences can be validated in the future, regardless of the software those experiments relied on.<sup>54</sup> A blog post on Recomputation.org states the problem with some assistance from *The Hitchhikers Guide to the Galaxy*. To paraphrase, if you build a computer to tell you the answer to a question, you need to write down the answer and the question in order for the knowledge to remain useful in the future.<sup>55</sup> It is important to note that these projects are interested in preserving the computational ability of software, not the interface or the aesthetic qualities outside of the code. While contextual information remains important to understand how and why these computational abilities were authored, the concerns with social and cultural relevance that one sees in digital art and software collection preservation projects are absent here.

## Conclusion

Preserving software is an important aspect of an overall digital stewardship practice, but there are no agreed upon best practices for accomplishing the multi-faceted desires and goals within this emerging field. While it is always important to establish best practices, it is important to remember that software is created for many purposes. With a wide variety of disciplines interested in the subject, there is a large amount of expertise from which to pull, and any individual or institution interested in starting a software preservation program will need to customize a strategy to fit their exact needs, the needs of their users and the nature of their collection. Best practices will need to be flexible in order to reflect the many use cases for and types of software.

VMs, in particular, present an interesting resource, as they may be of use in many situations and can accommodate software with many different dependencies from many different time periods. However,

---

<sup>53</sup> "Welcome to the ASCL." Astrophysics Source Code Library. Accessed November 12, 2015. <http://ascl.net/>

<sup>54</sup> "Recomputation.Org: If we can compute your experiment now, anyone can recompute it 20 years from now." Recomputation.org/ Accessed November 12, 2015. <http://recomputation.org/>

<sup>55</sup> "How Do We Know the Answer is Not 43?" Recomputation.org. Accessed November 12, 2013. <http://recomputation.org/blog/2014/09/14/why-isnt-the-answer-43>

regardless of tools, it is clear that documentation of the development and preservation process is a key aspect to all strategies. Whether an institution can afford in-depth, individual documentation, like that done for Agent Ruby at SFMOMA<sup>56</sup>, knowing what is being preserved and how will be vital to future users and preservationists as they try to access executable content in the years to come.

As NLM continues to research and pursue its own software preservation strategy, we hope to show our work, our mishaps, and our successes, in order to ensure that the digital stewardship community as a whole can grow and begin to tackle software in an enlightened and thorough manner.

---

<sup>56</sup> Sterrett, Jill, and Mark Hellar. "Virtualizing Agent Ruby: Collecting Web Art." Lecture, 2010 Summit of Documentation and Conservation of the Media Arts Heritage, Universite Du Quebec a Montreal, Montreal, March 5, 2010.

# Criteria Document

Explanation of Why “HowTo Grateful Med” was Chosen for the Pilot Project

## Introduction

As part of the National Digital Stewardship Residency (NDSR), the software preservation pilot project at the National Library of Medicine (NLM) aims to contribute to the stewardship of software and interactive digital materials by devising a preservation strategy for in-house developed software at NLM. The project is organized around the creation of a preservation strategy for one piece of historic NLM-developed software that can be expanded to other pieces of software in NLM’s collections. Due to reasons that will be explained later, any possible expansion of this project would require extensive adaptation.

This document details and defends the criteria used to pick the piece of software that was chosen for this project, “How-To” Grateful Med. Although it would be ideal for this document to serve as a guideline for future appraisal of software-based historic materials at NLM, it is not advisable to use the criteria used for “How-To” Grateful Med as the basis of a wider selection program due, in part, to the peculiarities of the history and practice of software development at NLM, which will be briefly described shortly. With this in mind, this document details the selection criteria and process for the pilot project and then describes how the criteria could be adapted to future selection processes.

## Background and Context

The first phase of this project consisted of researching the history of software development at NLM and attempting to locate working copies of historic software. This initial phase proved difficult. When this project was originally scoped, there was an implicit assumption that we would have many working copies of software from throughout NLM’s history to choose from. The opposite has proven true. After six months of research, interviews, and investigation, nine functioning copies of software programs were located that were not copies from the Institutional Archives. While copies of software programs on obsolete media exist in the Institutional Archives, it was decided that it would be prudent to test workflow and equipment on non-accessioned items first. It is important to note that any copy of software in the archives was accessioned as part of a larger collection which could include marketing materials, design specifications, and other analog materials. While particular pieces of software are well represented in the archives, like Grateful Med, software as a medium is not actively collected at NLM.

The appraisal and description of software artifacts moving forward cannot be done in a backward looking model, as this project was designed. Relying on chance to retain copies of software will result in loss. We need to start actively collecting software as it is being developed or while it is still in use. In an ideal situation, appraisal would occur either concurrently with development or closely thereafter. Regardless of where in the software development and use workflow appraisal would occur, it is clear that it needs to be done far sooner in order to save these materials.

The need for earlier appraisal is going to become direr as time goes on. Locating software held on tangible media is easier than locating software held in cloud environments or on institutional servers. This is true for two reasons. The first is the variety of security protocols to access the servers on which software may be held, and the second is the nature of server use itself. At NLM, gaining access to a

server as someone outside of a department or unit is difficult. It requires knowing who to contact and being able to convince them and their team that you should be able to view their work in progress with little supervision. It is not an easy administrative task. Furthermore, even if one were able to access a server, it is unlikely that a historic copy of a software program would be saved over a long period of time on an active server. Some of the software located on tangible media had been sitting in an office for thirty years, unnoticed on a 3.5" floppy disk. Software on a server is unlikely to last that long by chance.

## The Selection Process

Regardless of the specific strategy for software preservation, software preservation requires many stakeholders to work together and cooperate on an advanced technical and cultural practice. In this way, the selection process for a piece of software in this context should rely on consensus more than anything else. "How-To" Grateful Med was chosen through several meetings that were informed by a series of documents that described the history of each piece of software and the technical challenges of preserving that piece of software.

Although a vote was eventually taken to make the decision on which piece of software to preserve, each person was given the space to speak about which software program they found to be the most important and which technical challenges they saw as the biggest issue. While the choices were limited, focusing on the consensus process throughout selection allowed each stakeholder to voice their opinions and participate in the future of the project in a meaningful way. Using a consensus-based approach helps ensure the sustainability of the project because it allows more people to feel involved and valuable to the overall project.

## What We Mean When We Say "Preserve"

The current preservation strategy for software at NLM is to take the bytes found on tangible media, disk image it, and put it on the pre-existing NLM digital repository. These files will then be made available through the Digital Collections. Users will be responsible for creating their own virtual machines since the NLM Digital Collections does not yet have the capacity for in-browser emulation. Although a change in strategy would not necessarily change the criteria listed below, it could. While it may be ideal to create selection criteria that only recognizes the historical importance of a piece of software, it is impossible to ignore technical issues. Considerations of technical feasibility appear in this criteria and will probably continue to appear in any selection criteria created in the future or adapted from this document.

With this background in mind, it is possible to discuss the selection criteria for software preservation at NLM. The criteria will be broken into two sections: (1) technical factors, and (2) historical factors. The remainder of the document will outline how this criteria could be adapted for future use.

## Criteria

### *Technical Factors*

There are three technical factors for the selection of a software program. While they may seem simple at first glance, these factors can be difficult to address. They are: (1) There is a copy; (2) That copy is readable; (3) The extent to which the piece of software could be run in a manner that respects the original intent of the developers. In order to explain what these factors are and how they affected our

eventual decision, we need to continue our discussion the history of the project and the culture of software development at NLM.

Finding copies of a piece of software is not necessarily easy. Yes, many copies of a single piece of software may have existed at some point in time, but preserving or collecting software has not been a part of NLM's official records management strategy. One staff member in particular had kept a large amount of materials on her own volition, and without her assistance, fewer viable copies of software programs would have been found. Actively collecting copies of our in-house developed software is an important issue for the future of this project and establishing a collection or acquisition strategy presents a challenge for the ongoing preservation of NLM-developed software.

Locating readable copies on tangible media can also present a challenge. Some of the floppy disks which may have held valuable material had been bent. Not all copies of software on tangible media have been held in proper conditions, particularly because these copies used in this pilot project were saved by individuals rather than by an institution with carefully agreed-upon best practices.

While technical dependencies – like operating systems requirements or hardware requirements – can be addressed through techniques like emulation, it still may not be possible to run the software as it was originally intended to run. Many of NLM's hallmark programs provided access to NLM's data, and the programs need access to that data in order to operate properly. NLM has been diligent about at migrating data into new formats, so running historic software requires access to the data as it existed at that point in time. If the software program is to run properly, it would be necessary to locate un-migrated versions of the data or to migrate the data back to older formats.

This type of dependency does not stop the software program from running, but it can stop future researchers from being able to view the software as original users did and to ascertain proper meaning from it. For example, Grateful Med relies on several external files in order to be able to search NLM's bibliographic data. First, there needs to be data files in the format that Grateful Med read. Second, those data files need to use the controlled vocabularies and thesauruses that Grateful Med uses. NLM creates and updates controlled vocabularies, subject headings and thesauruses about medicine and health each year. The data files from 1980 will use the vocabularies from 1980. In order to have Grateful Med search data files, it would be necessary to find data files from that year or we would need to migrate the data back to an old format and create crosswalks for the historic controlled vocabularies.

Due in part to concerns about this type of dependency, "How-To" Grateful Med was chosen for the pilot project. "How-To" Grateful Med is a tutorial program on how to use Grateful Med effectively and is a self-contained. Without the same dependencies as Grateful Med itself, "How-To" Grateful Med will be easier to provide access to and to ensure that future users may be able to use it and experience it in a way that is not anachronistic or unhelpful.

### *Historical Factors*

Determining the historical importance of a particular artifact is a subjective decision. There are multiple factors for historical importance, and these factors need to be weighed against each other in order to present a full and detailed depiction of how a particular piece of software fit into and perhaps changed the world around it. Before outlining these factors, it is necessary to discuss a key distinction found in the software programs isolated for the pilot preservation project.

The types of programs located for this project can be split into two categories: (1) pieces of historic software; (2) pieces of old software that demonstrate historic software. Grateful Med was the only piece of historic software that was located in a readable format. The other pieces of software were tutorials or demonstrations of historic software. “How To” Grateful Med, our eventual choice, is a tutorial about how to use Grateful Med. It includes example searches, demonstrations of how to install Grateful Med, and an explanation of how it works. In this way, it is self-contained. Unlike Grateful Med, it does not have any knowledge-based dependencies and presents a much easier preservation process, but it also is not the historic piece of software itself. While it would be ideal to preserve the piece of software itself and allow future users to interact with it, technical issues already dictated that it would be impossible for future users to fully interact with Grateful Med. “How-To” Grateful Med demonstrates and documents Grateful Med in a way that written documentation, photographs, and videos cannot. The chosen program includes interactive elements and a limited set of data that lets the users use a demo version of Grateful Med. In this way, technical factors outweighed this historical distinction, for better or for worse.

There were several other tutorial programs to choose from that were self-contained and technically feasible. Historical factors did affect which of these tutorials was chosen. The options were “How-To” Grateful Med, CHEMLEARN, ELHILL LEARN, TOXLEARN, and MEDTUTOR. CHEMLEARN, TOXLEARN, and MEDTUTOR were tutorials for particular databases that NLM produces. They were specific to one database each. Because of this, these programs were removed from consideration relatively early. As far as the breadth of coverage goes, these three programs covered the least historically important works. ELHILL LEARN was another contender, and it was the only program that was fully considered in conjunction with “How-To” Grateful Med.

These two programs were fully considered because of the importance of the software that they demonstrate. This importance was determined with the following historical factors: (1) breadth of use of that piece of software by patrons and staff members of NLM as an indicator of possible involvement in advances in medical research and practice; and (2) technical advancements illustrated by that piece of software. Through a consideration of these two factors, “How-To” Grateful Med was chosen over ELHILL LEARN.

ELHILL LEARN is a tutorial on ELHILL, a software program designed to search MEDLARS databases. ELHILL also functioned as the backend of Grateful Med. It was used widely, but its use was greatly facilitated by the creation and popularization of Grateful Med. ELHILL was a technical advancement, for sure, but it does not represent a philosophical or theoretical change to the public access of bibliographic data; Grateful Med does. Prior to Grateful Med, search of NLM’s bibliographic data would need to be done by a specialist, generally a medical librarian in an academic or hospital library. That librarian would need to be trained on how to use that particular database or search function and would need to be conversant in the particularities of that one piece of technology.

Grateful Med is part of a transition away from that particular design strategy and towards a design strategy that prioritized the experience of the users and aimed towards user-friendly design. While tutorials and guides still ended up being necessary, the end goal of the design process for Grateful Med was that the user would be able to interact with the program without having outside training. In this way, the burden of comprehension was moved from the user to the designer, and this philosophical transition and Grateful Med’s participation in it demonstrates an important part of design and

computing history. So, although ELHILL was also an important program that was used by many in the medical community, Grateful Med represents an interesting change in computing history and was widely used by the medical community. It introduced doctors and medical researchers to the process of digital searching, and its impact is immense although difficult to quantifiably measure.

## Adaptions for Future Selection

As stated in the introduction, this document will need to be adapted for future use. This criteria were shaped by what software programs were available. If it is possible to work software into NLM's collection strategy and to move the appraisal process closer to the development and use of the software program, then this criteria would need to be fleshed out more thoroughly. That said, the criteria outlined here could prove useful in the future. If software was collected earlier, issues related to dependencies could be mitigated. There are multiple avenues for preserving software. If appraisal was moved earlier into the software lifecycle, more of these options would be available to the institution. However, any changes made to the preservation strategy may affect what criteria is used and how that criteria is implemented.

That said, several constants can be ascertained regardless of future changes. First, it is possible that the process for selection could remain the same. Emphasizing consensus and involving all departments that would be necessary for a functioning software preservation program could be valuable to future projects. Second, separating technical and historical factors in the criteria helps stakeholders weigh their options. Compromise is generally necessary for a process as complex as software preservation, particularly when working with limited resources. Understanding these two categories as separate but related makes it easier to weigh the pros and cons. Lastly, separating the effects of a piece of software on medical history and on computing history further helps stakeholders understand the options and the scope of the effects of the software on history.

When a project requires as much co-operation as software preservation does, it is helpful to use consensus as part of the decision-making model. By using consensus, clear communication becomes incredibly important. Separating categories for criteria allows people to discuss their opinions and weigh the options in a clear and concise way. That, more than anything, is how this document could be adapted for future use, and although future recommendations for software preservation will be more thoroughly addressed in another document, a criteria for selection should first and foremost aid in communication across stakeholders regardless of the choice of software or the choice of software preservation technique.

# Digital Curation Process Report

Technical Guide for Software Preservation at the National Library of Medicine

## Introduction

This document describes how to prepare software and other digital assets held on 3.5" floppy disks, 5.25" floppy disks or CD-ROMS for ingest in the National Library of Medicine (NLM) digital repository. Included are instructions on hardware and software needs, connecting obsolete hardware to modern devices, metadata creation, and prepping the created files for repository ingest. A photo-documented example of the workflow is included as an appendix, as well as a guide to additional resources.

## Necessary Equipment & Tools

The following equipment has already been purchased and installed. However, in case the workstation is moved or adjustments need to be made, the equipment needs are documented below.

Tool	Use
NLM administrator privileges	In order to use FTK Imager to create disk images, it is necessary to have NLM Administrator Privileges. Contact NLM IT Service Center for further information.
3.5" drive	Many of our software assets are held on 3.5" floppies, and a USB-connected drive is needed to read these materials.
CD drive	Although CD drives are currently included in most desktop computers, they are becoming less of a standard feature. For any assets held on CD, it will be necessary to have a drive to read them.
FTK Imager	This software is available for free & open source at <a href="http://accessdata.com/product-download/digital-forensics/ftk-imager-version-3.4.2">http://accessdata.com/product-download/digital-forensics/ftk-imager-version-3.4.2</a>
TEAC FD-55GFR 5.25" Internal Floppy Disk Drive	This is used to read 5.25" floppy disks, but additional equipment is necessary because 5.25" drives cannot connect to a computer via a USB. While there are different models of the TEAC FD-55GFR drive, we currently are using FD-55GFR 7149-U5.
Device Side's 5.25" Floppy Drive Controller	We are currently using Device Side's model, but it is possible that a better controller will become available in the future. This allows the obsolete 5.25" drive to be connected to a modern computer via USB.
5.25" Floppy External Enclosure	In order to minimize damage to old drives, it is helpful to have a case to hold the drives and wires and keep them neat.
Power chord	Molex connectors instead of SATA power connectors
DeviceSide software	This was included in the controller and allows this computer to read the materials on the 5.25" floppy. It is necessary to read and disk image 5.25" floppy disks.

## Setting Up the Workstation

Please review that you have the necessary materials and permissions to make use of the workstation. Once you've done that, you can begin set up. This section is split into two sub-sections: (a) 3.5" floppy disks and (b) 5.25" floppy disks. Instructions on setting up hardware and software needs for the workstation are included in each section.

### *3.5" Floppy Disks*

Setting up a workstation to disk image 3.5" floppies is the easier of the two set-ups. It is important to note that this set-up can also work for CD-ROMS in case a computer is ever used for the workstation that does not have a CD/DVD drive built into it. Both CD/DVD drives and 3.5" floppy drives are sold as USB-connected devices.

We are not currently using a write-blocker. A write-blocker can be added to this set-up since it utilizes USB connected devices, if desired. We are also not concerning ourselves with detected malware. Since the software I have been working with was developed in-house, it is not necessary to screen for computer viruses. Software from the General Collection should also not present an issue with malware, but further consideration of malware detection and write-blockers may eventually be needed.

With these caveats in mind, the setup for 3.5" floppy disks is relatively easy. We currently have an external 3.5" floppy drive that connects via USB and is plug-and-play.

**Step 1:** Connect the 3.5" external floppy drive to the computer via USB chord



Figure 1: Connected 3.5" floppy drive via USB

**Step 2:** Next, you will have to download FTK Imager. FTK Imager is part of Forensic Toolkit (FTK), a suite of computer forensics software developed by AccessData. The version of FTK Imager that we are using is 3.4.2, released in October 2015. You will need NLM administrator privileges in order to download, install, and run FTK Imager.

Go to the [accessdata.com/product-download/digital-forensics](http://accessdata.com/product-download/digital-forensics) and select “FTK Imager” from the list of Current Releases.

**Step 3:** Follow the download and installation process. While the application is free, you will need to provide contact information in order to download the product.

**Step 4:** Open FTK Imager. It should look like the image included below.

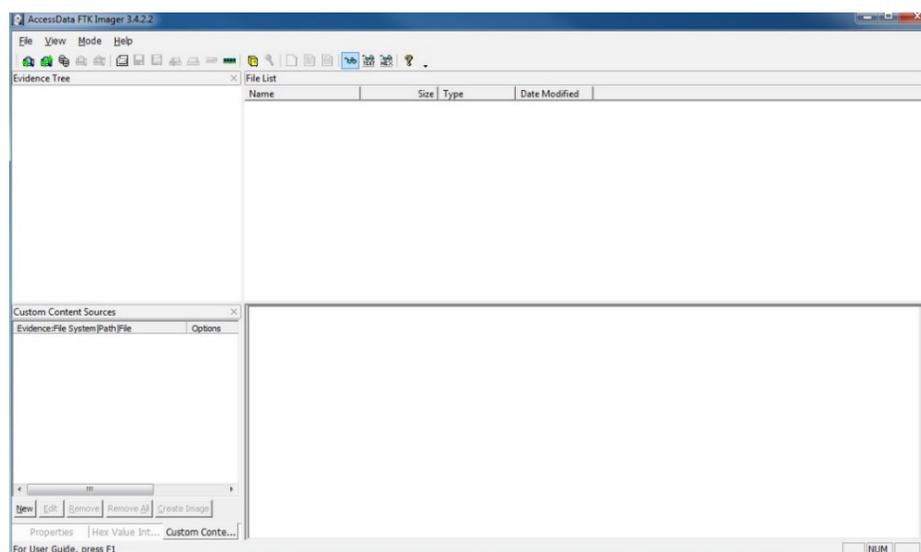


Figure 2: FTK Imager without devices attached

Thus, you have the software and hardware requirements for processing 3.5” floppy disks set. The section “Creating the Files” will give further instructions on using this set up to create the files needed for the Digital Repository.

### 5.25” Floppy Disks

Unlike 3.5” floppy drives or CD/DVD drives, there are no external 5.25” floppy drives that connect via USB to a modern computer. This is why Device Side’s USB 5.25” Floppy Controller Model FC5025 is necessary. You can use the DeviceSide Manual as an additional resource. The Manual is available on the CD-ROM that comes with the controller itself.

#### Step 1: Jumper the drive

The DeviceSide Controller was built for TEAC 55-GRF floppy drives. However, there are multiple versions of the 55-GFR drive, and the different versions will need to be jumpered differently. Refer to the DeviceSide Manual for further instructions. Jumpers are a set of small pins that can be covered with a small plastic box, known as a jumper block, that allow the computer to close electrical circuit as necessary on the circuit board. They are used to configure the settings for components of a computer. Below, you will see a photograph of how FD-55GFR 7149-U5 should be jumpered. It is easiest to do this

first, since the circuit board will be difficult to access once the drive is connected. The red arrows highlight particularly important sections of the circuit board for this process.

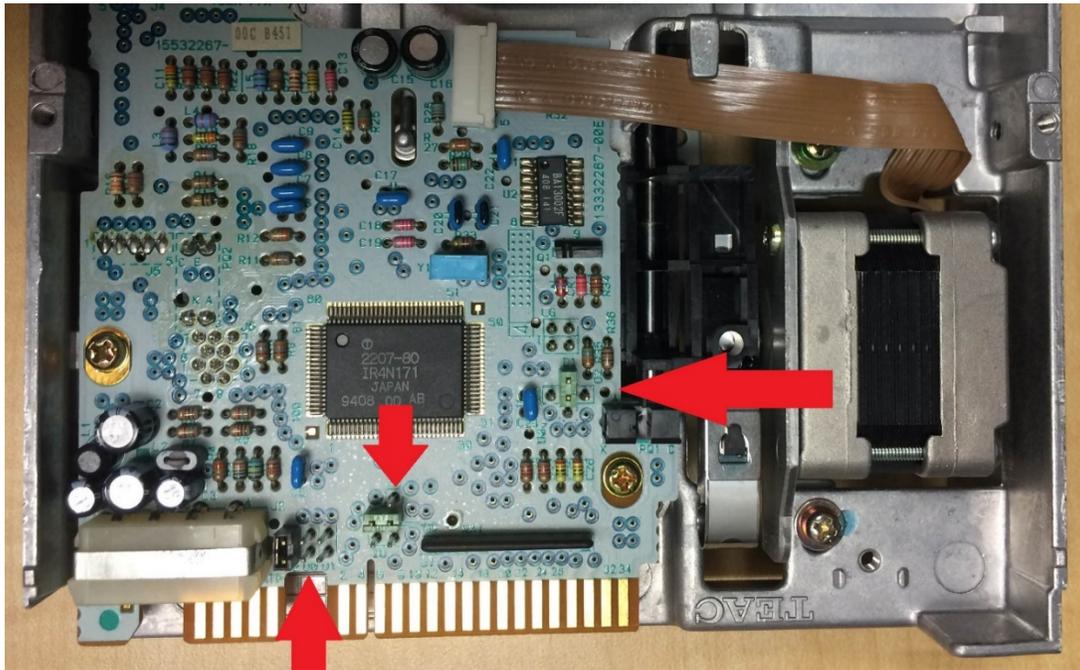


Figure 3: Properly jumpered FD-55GFR 7149-U5 drive

## Step 2: Attach the drive to the controller

Once the drive is properly jumpered, it can be connected to the controller. To connect the drives, first attach the ribbon cord to the controller. The red line at the side of the ribbon cord should be connected to the first pin, which is helpful labeled on the controller.

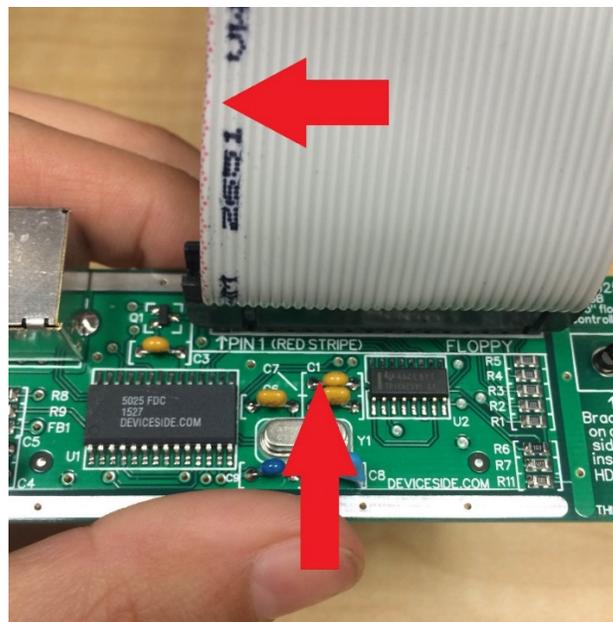


Figure 4: Correctly connected controlled, with red strip and first pin labelled with arrows

The ribbon chord can only be connected to the drive in one direction. Please do not try to force the ribbon chord onto the drive if there is resistance; instead see if it fits the other way.

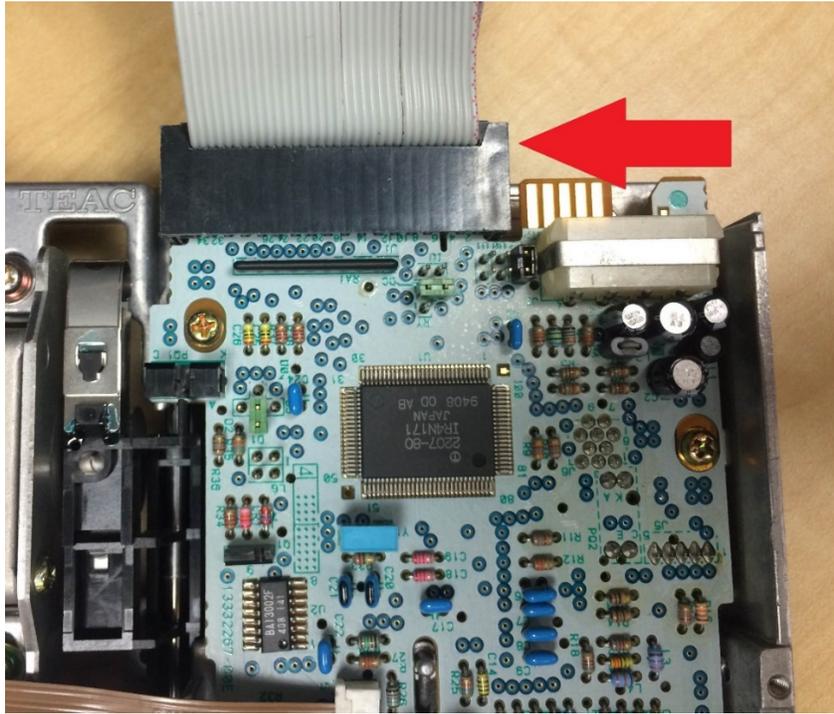


Figure 5: Corrected connected drive

**Step 3:** Place drive and other equipment in external enclosure

Place the drive in the external case, and nestle the controller at the back with the USB connection accessible.



Figure 6: Front view of the 5.25" drive in case



Figure 7: Side view of 5.25" drive in case

**Step 4:** Connect drive to external power source

Now, you will need to attach the drive to an external power source. Unlike the 3.5" floppy drive, it cannot be powered through the computer. In our current equipment set up, the power cord is supplied through the external case. Providing power is as simple as connecting the MOLOEX connector to the drive.

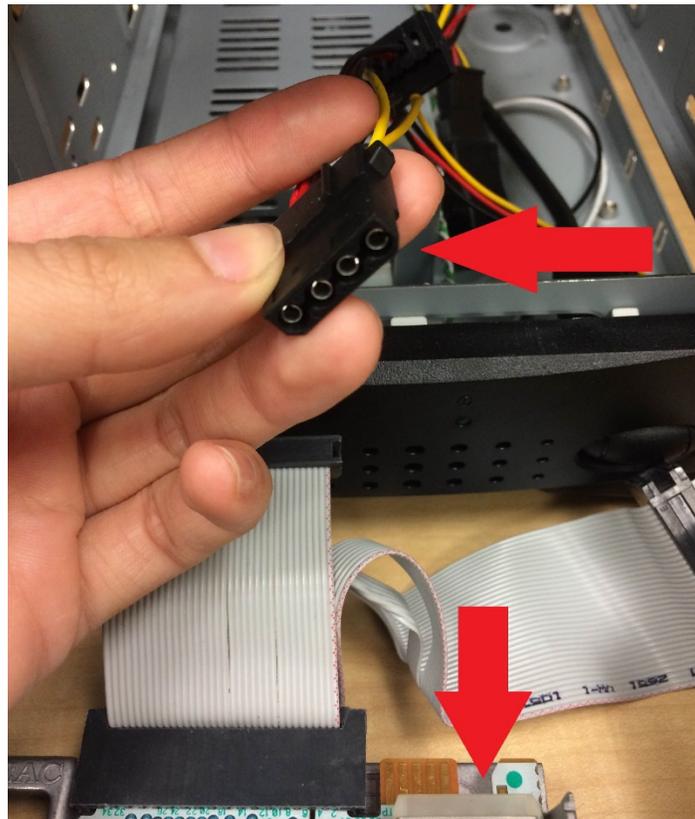


Figure 8: Molex connected and point of connection on the drive highlighted with red arrows

**Step 5:** The last step for the hardware set-up is to connect the DeviceSide Controller to the modern computer via a USB chord.



*Figure 9: Completed set-up for 5.25" drive*

You will notice that our external case has space for two drives. This was done for two reasons: (1) it was very difficult to locate an external enclosure with one bay; and (2) it allows for space in case a backup drive is ever purchased. All 5.25" drives are used and they are prone to issues. For the same of cleanliness, you can tape off the extra bay, but this is not strictly necessary.

Now that you have the physical media set up, it is time to set up the software. Currently, Device Side includes a CD-ROM with software, manuals, and source code for its product.

**Step 6:** Insert the DeviceSide CD-ROM with instructions and DeviceSide software.

**Step 7:** Install the software.

Again, to install and use their software, you will need NLM Administrator Privileges. You will need to click on the proper file for your operating system.

**Step 8:** Follow the instructions for installation.

Once you follow the instructions for installation, you should be able to open DeviceSide's disk imaging tool called, "Disk Image and Browse." When you open this tool, it will look like it does below.

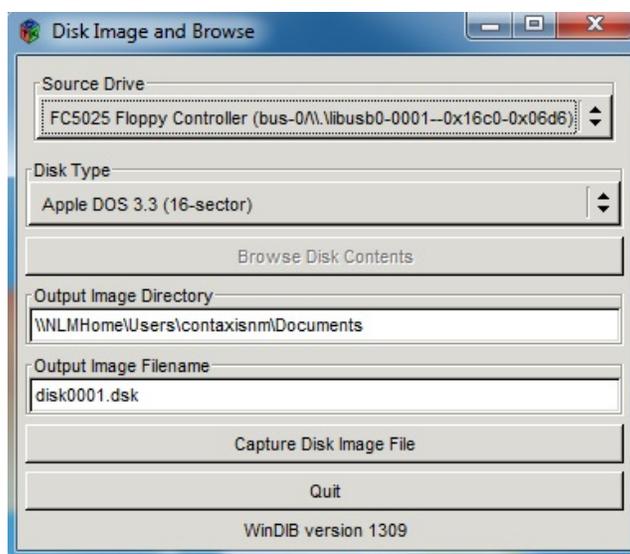


Figure 10: Opening screen shot of the DeviceSide software

At this point, you have successfully set up your workstation, and you are ready to begin creating the files to deliver to the Digital Repository.

## Creating the Files

While it would be ideal to have the same workflow for both sizes of floppy disks, the FC5025 controller only works with their software and cannot be adapted to work with the USB connected 3.5" external floppy drive. For that reason, this section will also be split into two subsections, one for 3.5" floppy disks and the other for 5.25" floppy disks. When you are working with CD-ROMS, please follow the directions for 3.5" floppy disks, as you will be using the same software for that media.

At this point in time, the files that need to be include for each software resource are: (1) the disk image in AFF format; (2) the disk image in dd raw format; (3) a zip file of all the files held on the disk image; (4) the guidance document for accessing and using the files; (5) images of the media that the asset was held on; and (6) screen shots of the asset if possible. While this document will not discuss how you should photograph the media or take screenshots, the guidance document is included in the appendix.

### 3.5" Floppy Disks & CD-ROMS

As stated earlier, this workflow will work for all media that isn't a 5.25" floppy disk. Below you will find see how to work with AccessData's FTK Imager 3.4.2.2.

**Step 1:** Open AccessData's FTK Imager 3.4.2.2, insert the media, and add all attached drives.

When you open FTK Imager with the media inserted, the media will not show up in the interface if you do not add all attached drives. In order to do that, follow the example illustrated below.

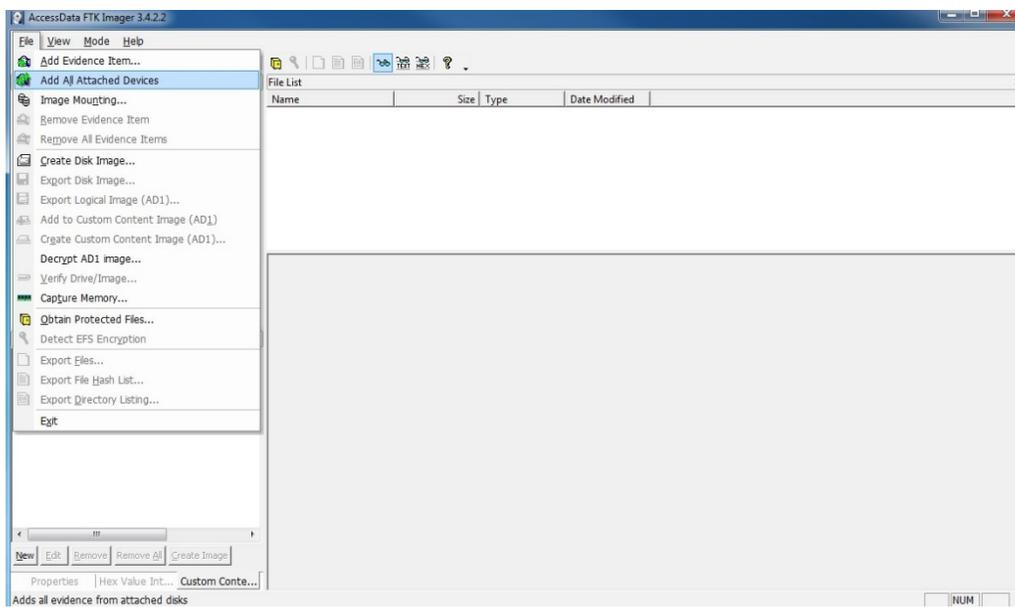


Figure 11: Where to find "Add All Attached Devices" on FTK Imager.

## Step 2: Select the correct drive.

At this point, you will begin to see a bunch of “code” seeming things in the interface. This is hexadecimal notation. Any digital data is simply an array of hex bytes, and this is a hex editor, which allows you to edit data in ways you normally can’t and shows the true contents of a file.

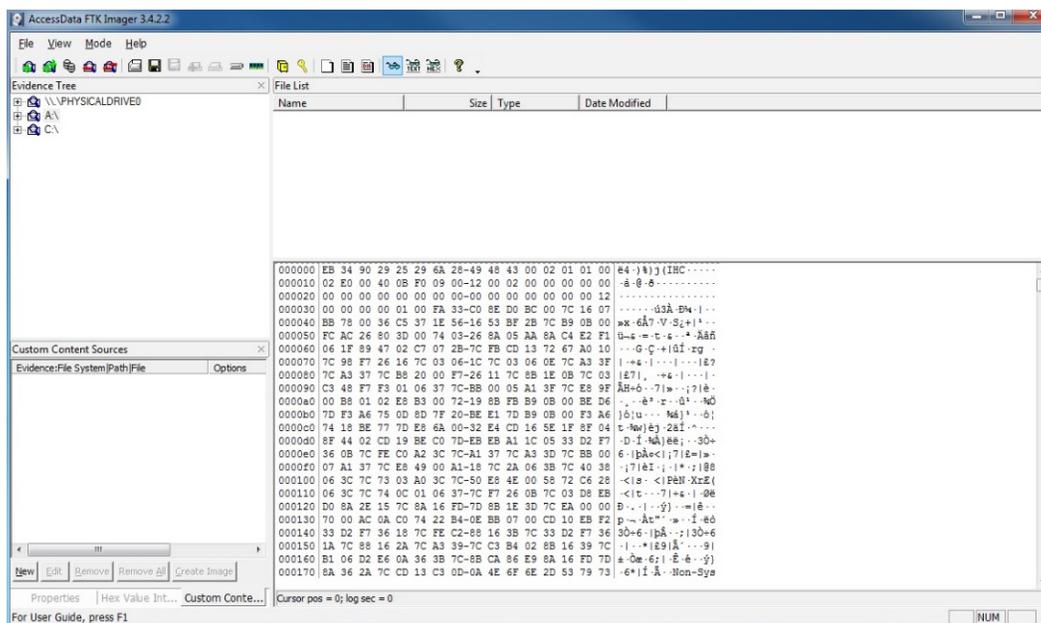


Figure 12: Select the correct drive, in this case A:./



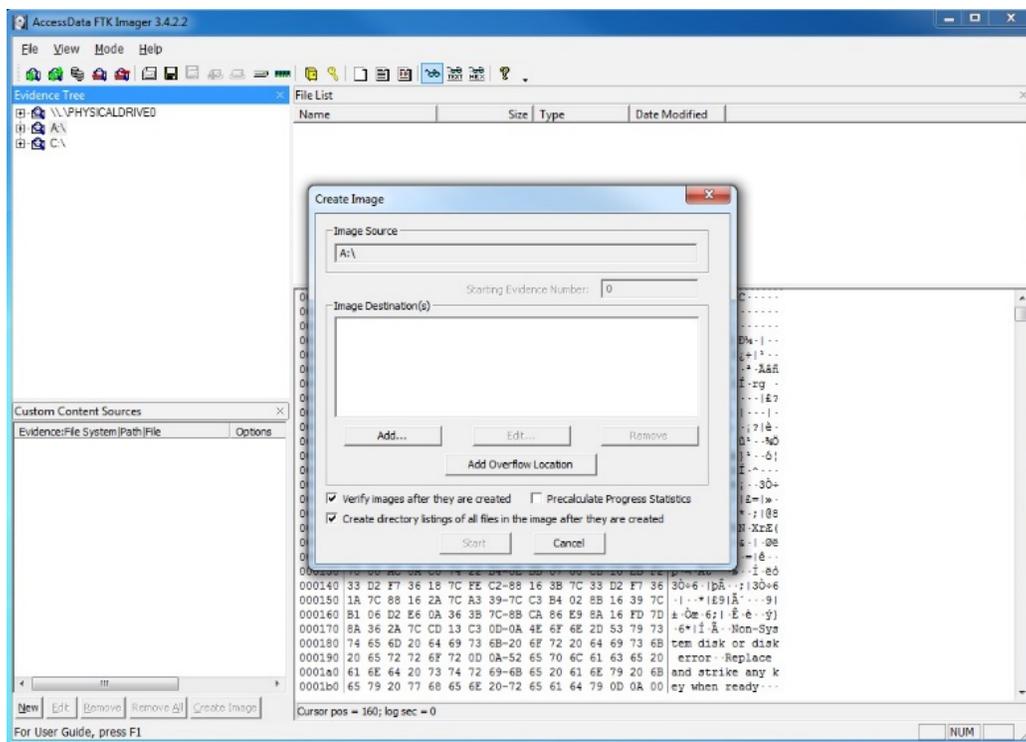


Figure 14: Interface to create the disk image

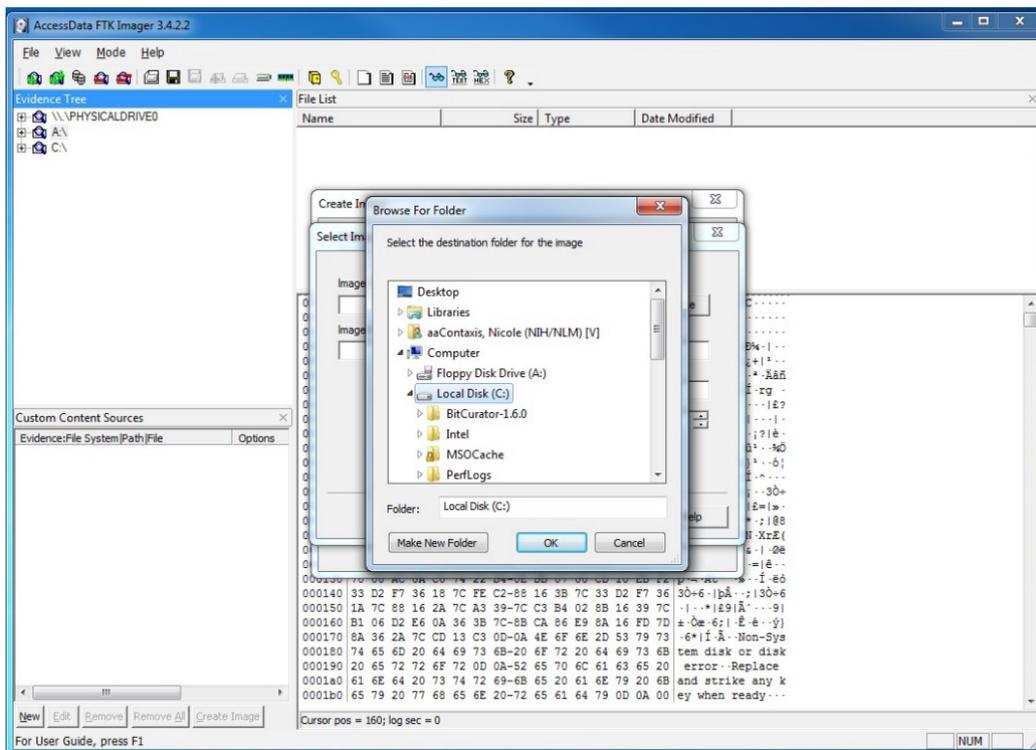


Figure 15: Selecting the destination of the disk image file

**Step 5:** Select the file format.

There are several available formats for disk images. Before it is an open source format with compression, we have chosen to work with the AFF format and dd raw. If this changes, which it may, select the proper format for your current workflow. In this case, you will be making two disk images.

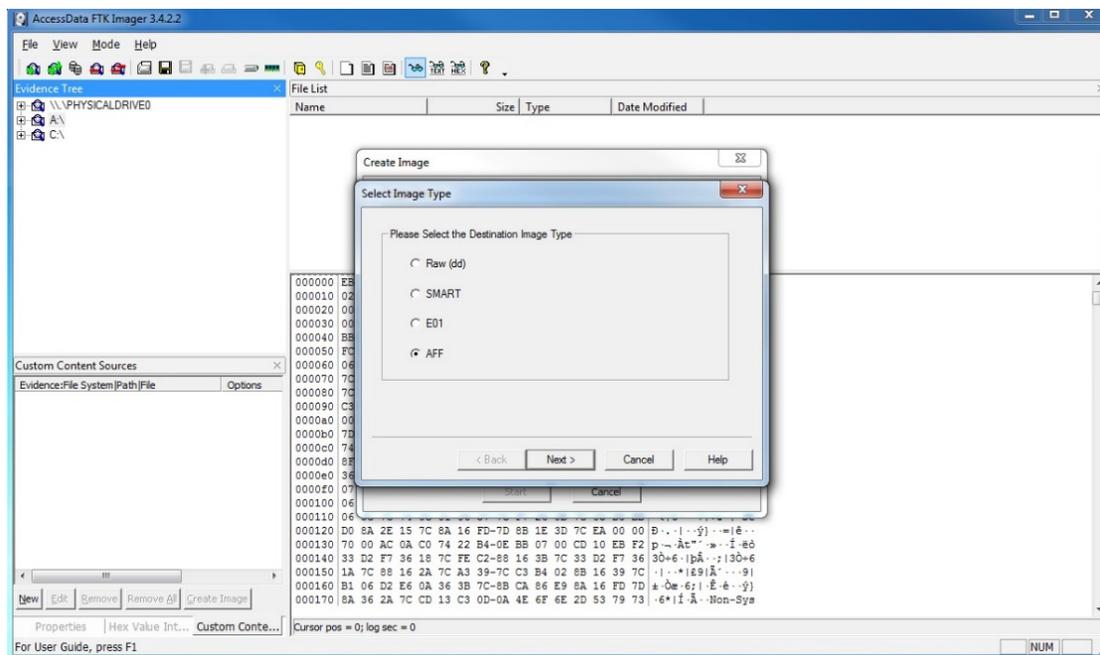


Figure 16: Selecting the format for the disk image, in this case AFF

**Step 6:** Enter metadata.

Because FTK Imager was built as a digital forensic tool, the metadata fields are more relevant for police organizations than archivists. However, I've created a crosswalk so that the metadata included with the file will be helpful for future users. The metadata will be included in a text file when the disk image is made that will also include information on checksums and any errors that occurred during the disk imaging process. Below, you can see a screenshot of the metadata elements included:

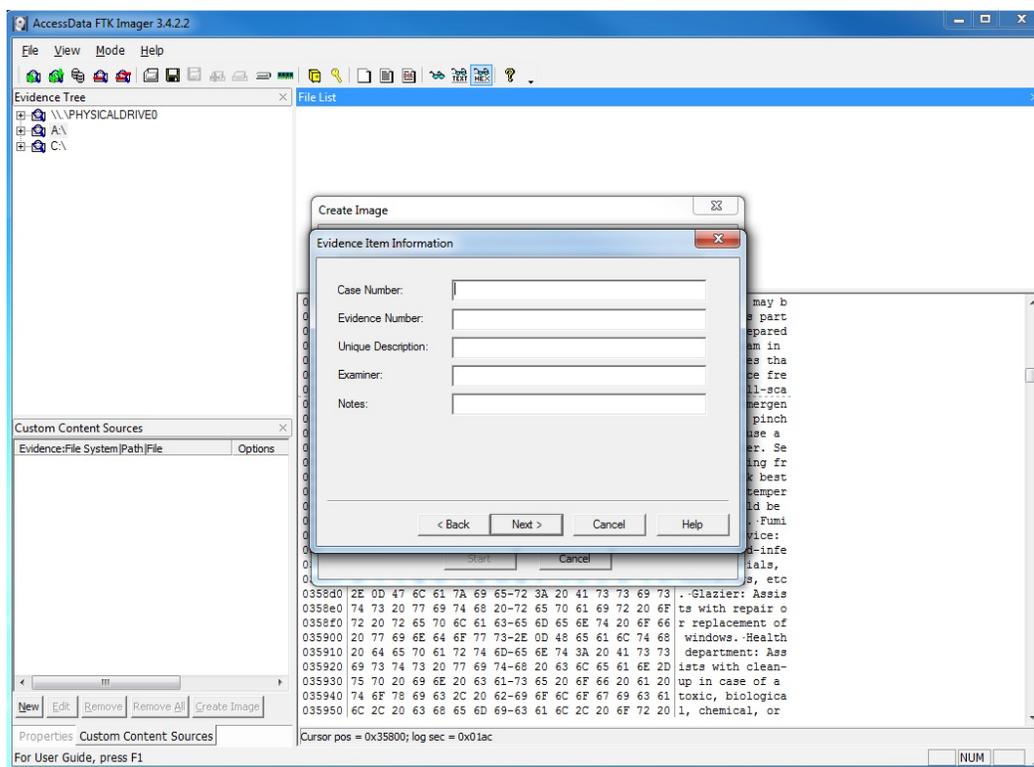


Figure 17: Enter that metadata!

- Case Number – Software title and version number
- Evidence Number – Disk number, if applicable
- Unique Description – NLM UID
- Examiner – Name of person who creates the disk image
- Notes – N/A

```

HowToGM.aff.txt - Notepad
File Edit Format View Help
Created By AccessData® FTK® Imager 3.4.2.2

Case Information:
Acquired using: ADI3.4.2.2
Case Number: HowTo Grateful Med
Evidence Number: Disk 1/1
Unique description: NLMUID 101679914
Examiner: Nicole contaxis
Notes:
-----

```

Figure 18: An example of the text file that FTK imager will create for each disk image with example metadata included.

### Step 7: Create disk image.

FTK Imager will create several files when it makes the disk image. The first is the disk image itself, both as an AFF file and as a dd raw file. The second is a CSV file that acts as a directory for all files on the disk image. The third is the text file depicted above, which includes a description of the process of making the disk image below the metadata.

	A	B	C	D	E	F	G
1	Filename	Full Path	Size (byte)	Created	Modified	Accessed	Is Deleted
2	[root]	HOWTO [F	3584				no
3	VBR	HOWTO [F	512				no
4	[unallocat	HOWTO [F	0				no
5	FAT1	HOWTO [F	1536				no
6	FAT2	HOWTO [F	1536				no
7	HINSTALL.	HOWTO [F	1094	N/A	1989-Feb-14 05:40:00		no
8	HOWTO.B	HOWTO [F	1162	N/A	1989-Feb-14 05:38:28		no
9	SPEEDSCR	HOWTO [F	4982	N/A	1986-Sep-01 12:00:00		no
10	H1.COM	HOWTO [F	1521	N/A	1988-Jan-04 00:30:58		no
11	CPI.EXE	HOWTO [F	57746	N/A	1987-Jul-07 13:39:32		no
12	HTEQ	HOWTO [F	39	N/A	1985-Jul-07 00:06:50		no
13	HTXD	HOWTO [F	20	N/A	1988-Aug-09 08:05:00		no
14	HTSC	HOWTO [F	58	N/A	1988-Nov-23 01:06:30		no
15	HTCO	HOWTO [F	11	N/A	1988-Nov-23 14:44:10		no
16	HTMS.PIX	HOWTO [F	10066	N/A	1989-Feb-15 12:01:30		no
17	HTM1.PIX	HOWTO [F	26866	N/A	1989-Feb-15 11:44:56		no
18	HTM2.PIX	HOWTO [F	25665	N/A	1989-Feb-15 11:44:58		no
19	HTM3.PIX	HOWTO [F	26334	N/A	1989-Feb-15 11:45:00		no
20	HTM4.PIX	HOWTO [F	19784	N/A	1989-Feb-15 11:45:00		no

Figure 19: Example of the CSV file created by FTK Imager

```

Created By AccessData® FTK® Imager 3.4.2.2

Case Information:
Acquired using: ADI3.4.2.2
Case Number: HowTo Grateful Med
Evidence Number: Disk 1/1
Unique description: NLMUID 101679914
Examiner: Nicole Contaxis
Notes:
-----

Information for C:\HowToGM:

Physical Evidentiary Item (Source) Information:
[Device Info]
Source Type: Logical
[Drive Geometry]
Bytes per Sector: 512
Sector Count: 1,440
[Physical Drive Information]
Removable drive: True
Source data size: 0 MB
Sector count: 1440
[Computed Hashes]
MD5 checksum: ba9058b423556a19945cad9bf1c72e79
SHA1 checksum: 47185b3160a1a4988e1331900edaff7047a1b719

Image Information:
Acquisition started: Fri Apr 01 10:14:34 2016
Acquisition finished: Fri Apr 01 10:14:34 2016
Segment list:
C:\HowToGM.aff

Image Verification Results:
Verification started: Fri Apr 01 10:14:34 2016
Verification finished: Fri Apr 01 10:14:34 2016
MD5 checksum: ba9058b423556a19945cad9bf1c72e79 : verified
SHA1 checksum: 47185b3160a1a4988e1331900edaff7047a1b719 : verified

```

Figure 20: Full example of the txt file FTK Imager will create

**Step 8:** Access the disk image you just created.

At this point you have created the disk image, but now you have to create copies of all the files on the disk image and put them into a zip file so that users can access the files more easily.

You will need to Add the Evidenced Item, i.e. the AFF file. The following images give a step-by-step example.

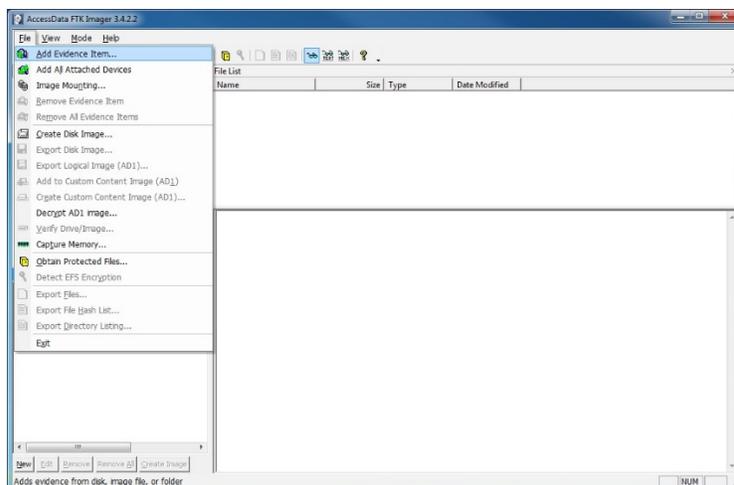


Figure 21: You will find "Add Evidenced Item" under "File"

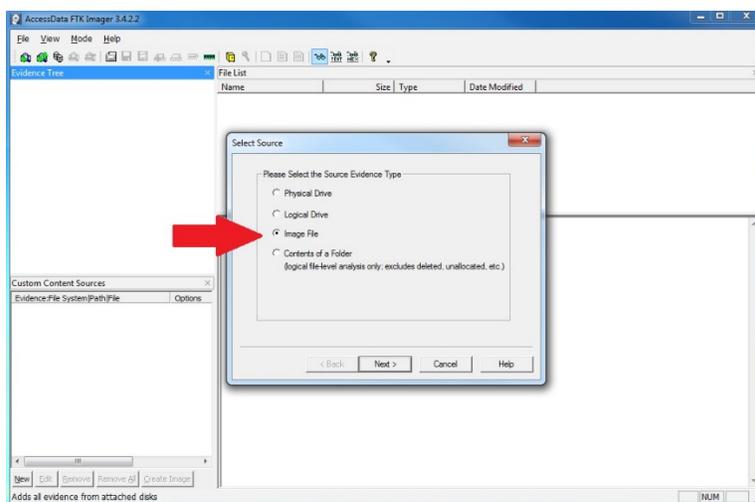


Figure 22: To add a disk image file, you will want to click "Image File"

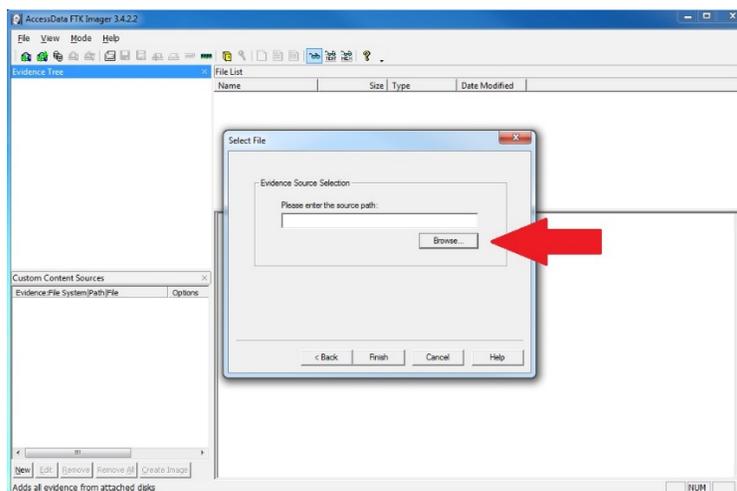


Figure 23: Click browse to select the disk image file you made earlier

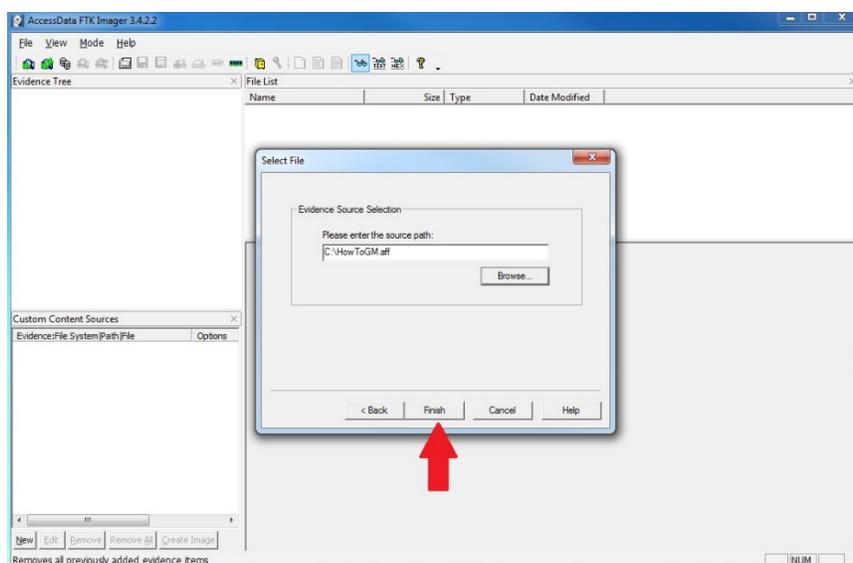


Figure 24: Click Finish to add the Evidenced Item

In this example, the AFF file was stored directly on the local machine to help deal with usability issues stemming from the Administrator Privileges account you need to use to use FTK Imager. Wherever you decided to put the AFF file is where you will need to go in order to add that evidenced item.

### Step 9: Export the files from the disk image.

Now that you have the disk image file accessed, you will need to open the evidence tree in order to view the files inside of the disk image file. The following images show how to export the files onto the local machine.

Be sure to remove the evidenced items after you export all of the files.

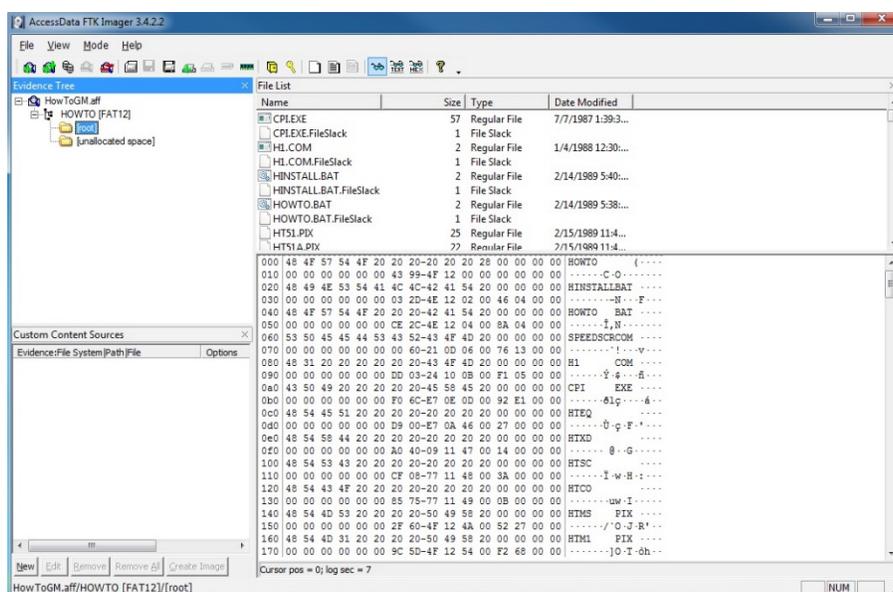


Figure 25: Open the evidence tree to view all files in the disk image file

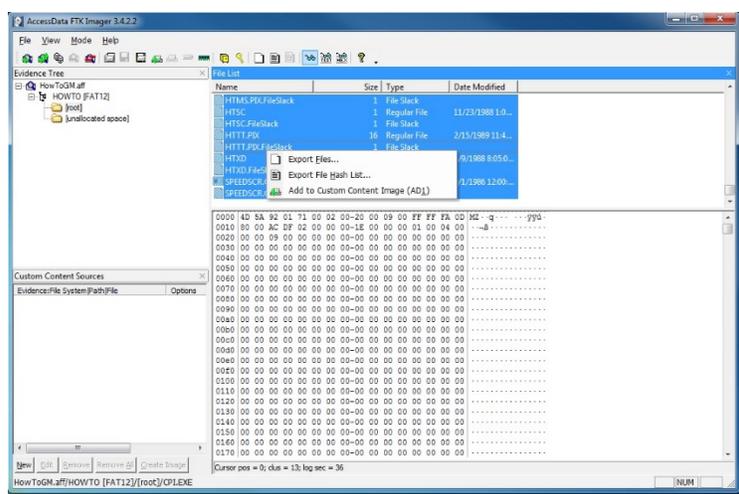


Figure 26" Select all files and left click to export

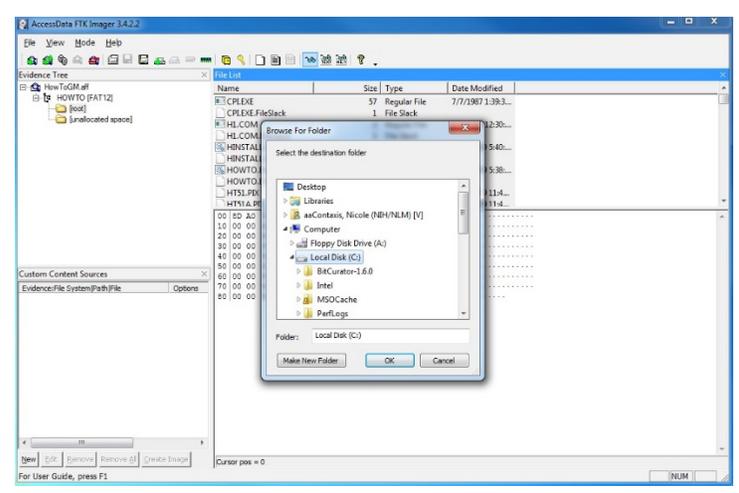


Figure 27: You can put the files wherever you like, but the C: drive may be the easiest option considering the complications of using NLM Administrator Privileges

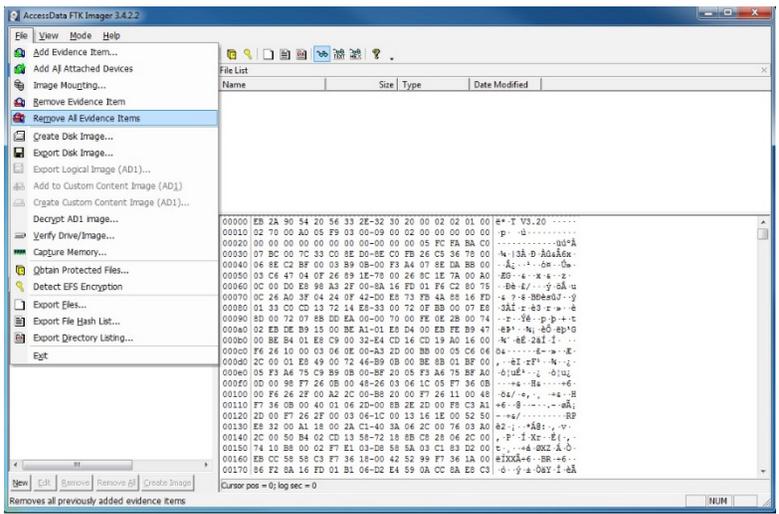


Figure 28: Remove Evidenced Items so that you can move the files later on without getting error messages

**Step 10:** Create a folder with all necessary files and move all files to that folder

You will want to move all of the files you just exported from the disk image file, the disk image file itself, the text file, and the CSV file that FTK Imager created.

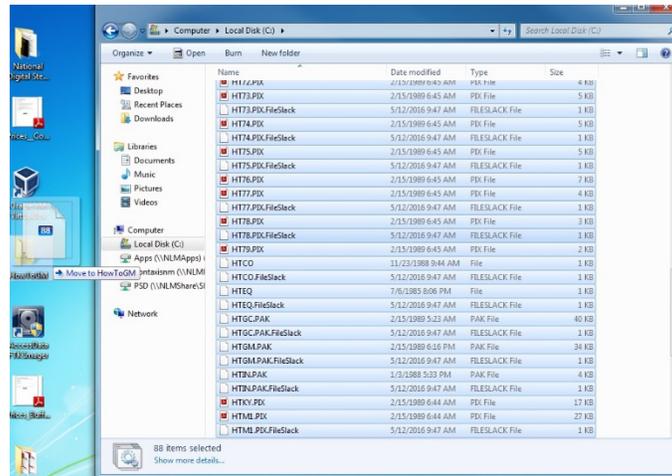


Figure 29: Getting all of the files into one folder

**Step 11:** Zip the files that were exported

In order to ease use, you will need to zip together all of the files you exported. Be careful not to include the disk image file, the text file, or the CSV file.

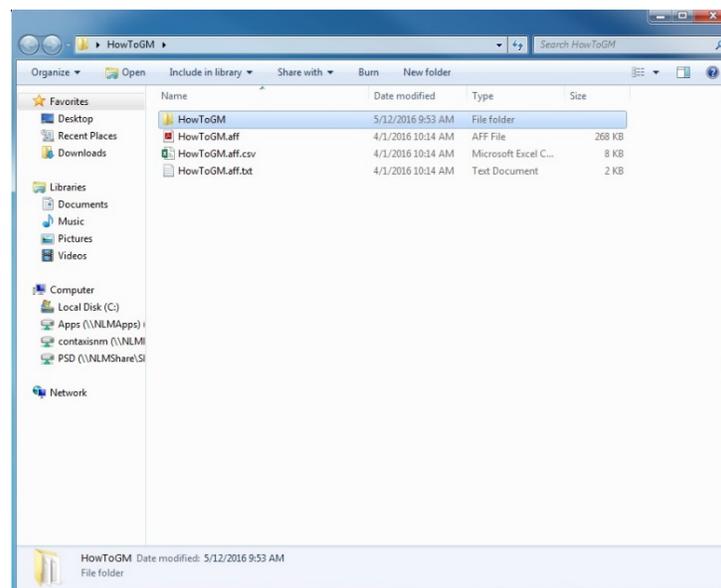


Figure 30: Zip it real good.

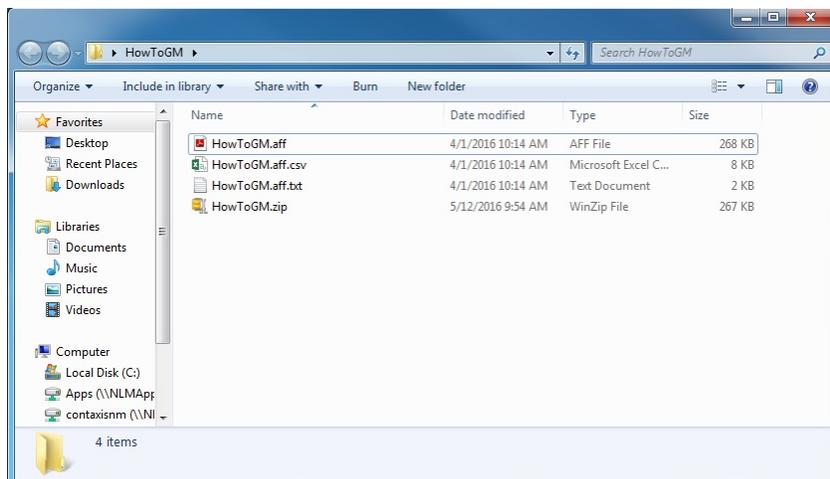


Figure 31: Now you are done.

### 5.25" Floppy Disks

As stated before, the DeviceSide controller comes with and works best with its own software. The only time you should use this software is when working with 5.25" floppy disks. Otherwise, please follow the directions for 3.5" floppy disks and CD-ROMS listed above.

**Step 1:** Open DeviceSide's "Disk Image and Browse" program.

You will need to update the Output Image Director to the where you would like the disk image file to go on the computer. Although using "Disk Image and Browse" doesn't require administrative rights like FTK Imager, you will eventually need to open the disk image you create here in FTK Imager. So, in order to make your life as easy as possible, you will want to save the image on the local machine rather than on a server. I put my images in my user folder on the local machine: C:\Users\contaxisnm\Documents.

The Output Image Filename will also need to be updated. The file name convention, as it stands now, is to use "Title of Software Disk of Total Disks." An example of this is "HowToGM1-3." The DeviceSide program will also write the disk image as an ".img" file. Using FTK Imager, you will eventually convert it to whatever file format you prefer. At this point, the preferred format is AFF. We will also be storing the dd raw format since it is not compressed. Further directions on that will be available later in this document.

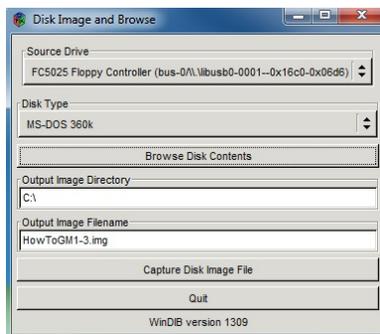


Figure 32: How DeviceSide's "Disk Image and Browse" looks when it is opened with the Output Image Director and the Output Image Filename correctly entered

**Step 2:** Select the correct disk type.

This step may involve some guess work. There may not be documentation on what disk type the disk is. For the most part, if you do not know what type it is, start with MS-DOS. It is the most likely option.

To preview what is on the floppy disk and to ensure that you have correctly selected the disk type, you can click “Browse Disk Contents.” This will allow you to see the files on this disk, and there will be an error if you select the wrong disk type.

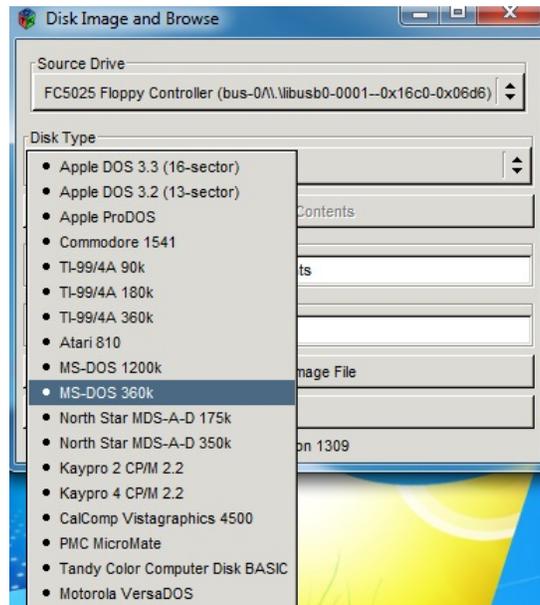


Figure 33: Select the disk type

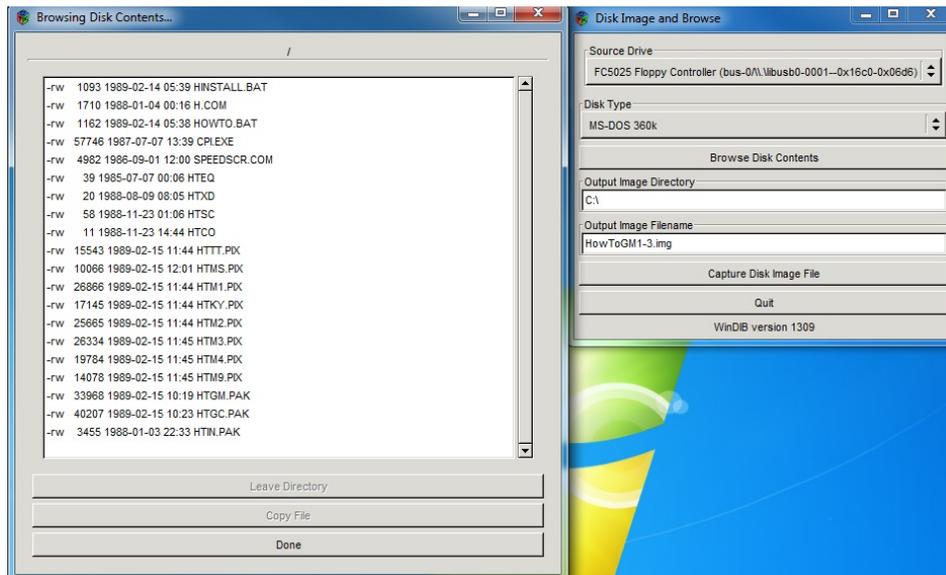


Figure 34: Browse the disk contents

### Step 3: Create the disk image

Simply click “Create Disk Image File,” and the disk image will appear in the Output Image Director.

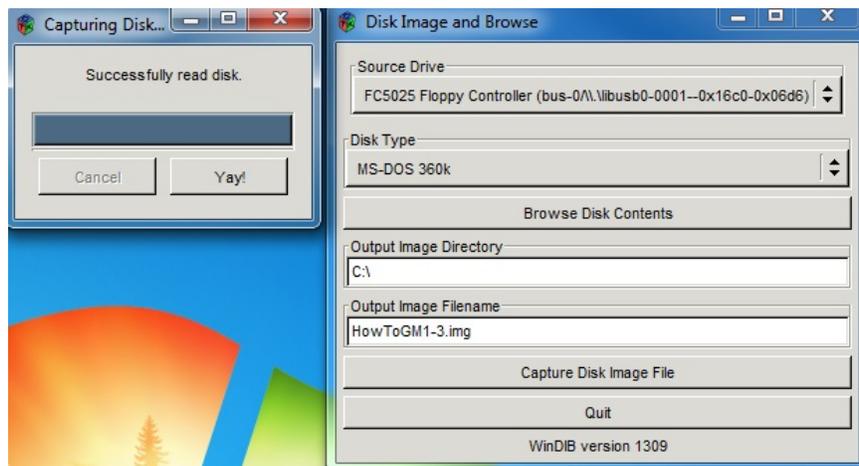


Figure 35: The program supplying good news!

### Step 4: Repeat these steps for as many disks there are for that software program.

At this point, it is probably clear that creating disk images from 5.25” floppy disks is more time intensive than 3.5” floppy disks. This is mostly because 5.25” floppy disks hold far fewer bytes.

### Step 5: Access the disk images in FTK Imager

You will need to open the new disk image files as Evidenced Items. You can follow the directions Step 8 from the 3.5” floppy disks Directions to do this.

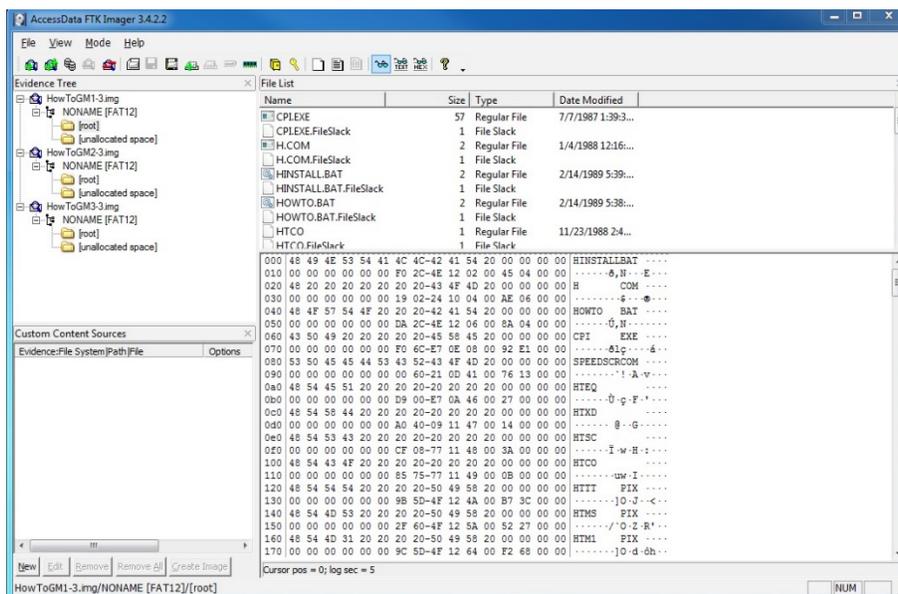


Figure 36: How FTK Imager will look when all the disk image files are added as Evidenced Items

**Step 6:** Convert your “.img” files to your desired format and enter in the relevant metadata.

To do this, you will need to left click on each Evidenced Item and click “Create Image.” This will allow you to turn the “.img” file into whatever file format you prefer. Refer to Step 4-5 for the 3.5” floppy disk directions for more details.

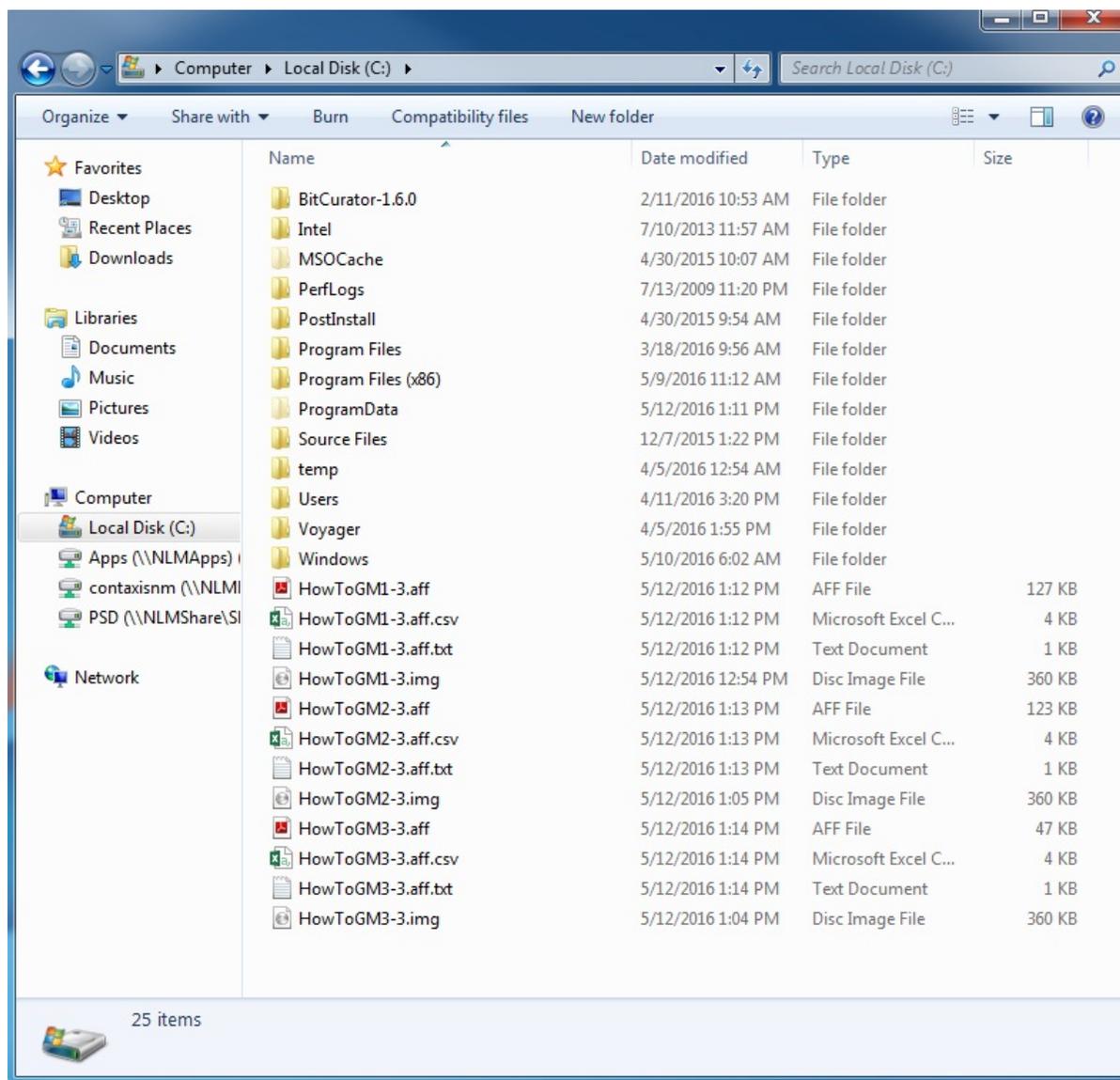


Figure 37: Your C:\ drive will look like this once you follow steps 4-6 from the 3.5” floppy disk section for each of the disks

**Step 7:** Export the files from your disk image files

For this, you can follow Steps 8 & 9 from the 3.5” floppy disk section of this document. You will need to be sure to keep your files organized so that you only group together files that are on the same disk. If you export all files from all the disk images at once, you will not be able to tell while files came from which disk image. Be careful and stay vigilant!

### Step 8: Group all files together

In order to keep everything well organized, create a folder for each disk within the larger folder.

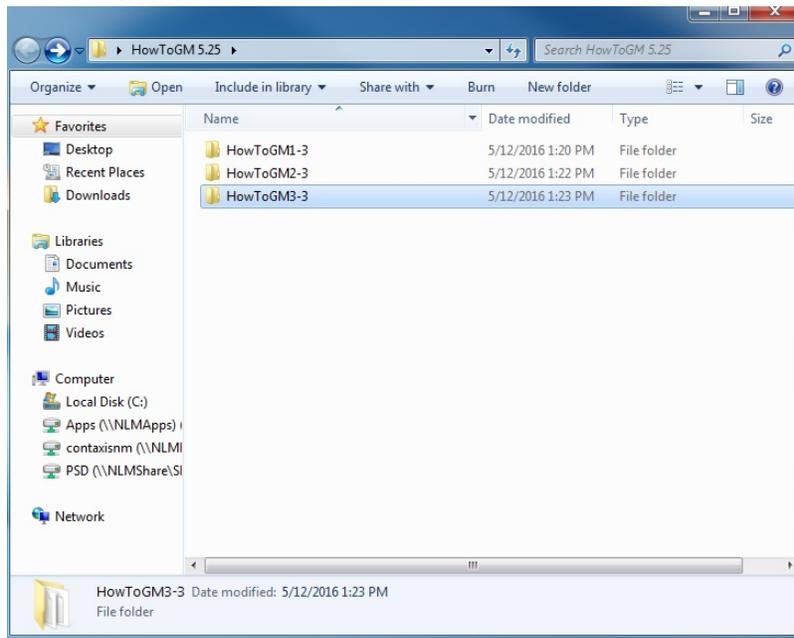


Figure 38: An organized folder is an accurate folder

Within each disk image folder, you should zip together, as with 3.5" floppy disks, all the files that you exported from the disk image. Each sub-file should look like the image below:

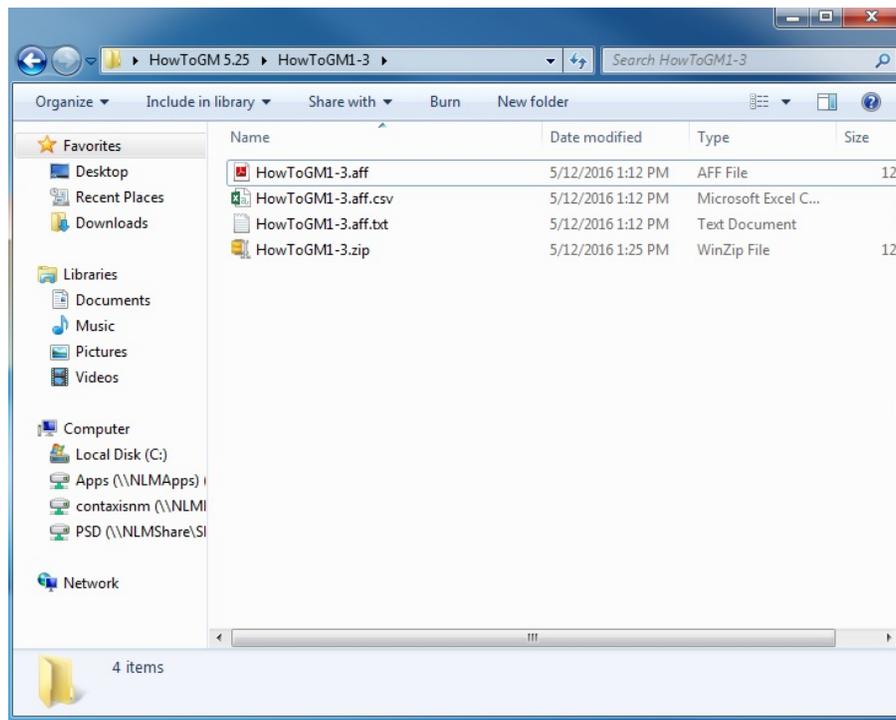


Figure 39: An organized sub-file is an accurate sub-file

## Further Reading For Disk Imaging and Technology How-To:

*User Guide: Forensic ToolKit: Find Computer Evidence Quickly and Easily*. Access Data Corporation.

Available at: <http://myweb.cwpost.liu.edu/cmalinow/ftk/ftkusersguide.pdf>

*Advanced Forensics Format (AFF)*. Forensics Wiki. Available at: <http://www.forensicswiki.org/wiki/AFF>

*FC5025 Software Instructions*. MITH's Vintage Computers. Available at: <http://mith.umd.edu/vintage-computers/fc5025-software-instructions>

*Rescuing Floppy Disks*. ArchiveTeam. Available at:

[http://www.archiveteam.org/index.php?title=Rescuing\\_Floppy\\_Disks](http://www.archiveteam.org/index.php?title=Rescuing_Floppy_Disks)

# Future Directions

Software Preservation and the National Library of Medicine in the Near, Intermediate, and Long Term

The National Digital Stewardship Residency (NDSR) Project, “NLM-Developed Software as Cultural Heritage,” was designed to explore the history of software at the National Library of Medicine (NLM) and possible strategies to preserve that history. At the onset, there were many unknowns for the project, including what software would be found, what software would be readable, what strategies were being implemented at other institutions, which strategy would best suit NLM, or what metadata elements were necessary for software assets at NLM. While some of these questions were addressed through the year, some of them remain unanswered. This document seeks to highlight existing efforts and outline recommendations, opportunities and considerations for the near, intermediate, and long term that would enable the preservation and access of interactive digital objects, like software, at NLM.

Although this document outlines future recommendations and the reasons for those recommendations, it does not supply step-by-step instructions on how they should be implemented; best practices, technology, standards, and policies in this space are continually evolving and this document should be read in that light. Several documents produced in the process of this residency may provide assistance in the technical matters specific to software preservation, including the Inventory of Legacy Software Products at NLM and the Curation Process Report. Furthermore, an extended reading lists of outside resources is available in the final report. These resources may prove valuable to future software preservation projects of programs.

This document offers recommendations for near, intermediate, and long term activities that NLM should consider to build on the work completed for this residency. Each will likely require additional resources or infrastructure currently not in place within Library Operations, or spending political capital to make them successful. The suggestions for the near term relate largely to current preservation and access efforts of known born digital software and their physical media across Library Operations. The inherent fragility of these materials carries with them the additional burden of acting sooner rather than later. The intermediate term recommendations address issues that will take significant time and preparation and are additionally subdivided into considerations including selection and appraisal, metadata, and access. Long term opportunities seeks to identify larger high-level management considerations about the future goals and strategic priorities at NLM. They also address legal matters outside of the NLM purview. If the stewardship of interactive complex digital objects, either collected or produced by NLM, is to be a priority, proactive steps are required to ensure their long term preservation.

## Near Term

*Recommendation: Identify digital files stored on obsolete media and transfer them to networked storage*

As articulated in Harrison Inefuku’s 2010 ARL Fellows report for the HMD’s Images and Archives Section and the National Digital Stewardship Alliance’s Levels of Preservation, some of the most important actions that NLM can take to better secure the long term preservation and access of its born digital

collections is identifying digital files held on obsolete media and transferring the content off of the media and into networked storage systems. This recommendation is for all digital files, not just software. Obsolete media, including CD-ROMS, 3.5" floppy disks, and 5.25" floppy disks, deteriorates, and the hardware and operating systems required to read them can be hard to procure. While the notion of benign neglect of library material stored in appropriate physical and environmental conditions may be considered on phase of a preservation program for paper-based materials, this paradigm does not satisfy the requirements of digital media.

Building on Inefuku's ARL Residency, HMD's Archives and Modern Manuscripts Program instituted a collections survey to identify and record the location of digital media, as well as created new policies to better capture this information for new acquisitions. Staff are also actively transferring those digital files from their source media to networked storage. Similarly, PCM has inventoried much of the digital content found in General Collection's commercial produced instructional and learning publications. It is recommended that both HMD and PCM continue to advance this work in partnership and that PCM prioritize the transfer of General Collection digital files off their original media.

The residency's pilot project exposed the existence of degradation of digital objects and content loss. For example, several 3.5" Grateful Med demo floppy disks in the NLM Institutional Archives could be opened, but the disks held no data. This could be the result of degrading disks or bad production of the original. However, the cause is unknowable, due to the significant passage of time. Obsolete media data recording technology, as well as outsourcing solutions, exists, but investigating this potential was outside the residency's scope.

Disk imaging is one common best practice for transferring digital files of original media, especially for executable software products. Disk images are computer files containing the contents and structure of a disk volume or an entire data storage device. A disk image is usually made by creating a sector-by-sector copy of the source medium, thereby perfectly replicating the structure and contents of a storage device independent of the file system. The Digital Curation Process Report outlines some recommended disk image formats, production software and hardware, and procedures for implementing a disk imaging workflow. Alternatively, disks used as storage media for files produced by desk top applications such as word processing documents generally do not require disk imaging. Simply copying the files using write blocking techniques and hardware and packaging them with appropriate technical and provenance metadata often suffices.

It is important to note that because different departments are responsible for differ parts of the collections, it may not be feasible or prudent to create a single approach to digital materials. Moreover, the specific and peripatetic nature of born digital collections requires an intellectually curious mentality in order to understand the nature of the content, articulate use cases and success outcomes, and choose the appropriate preservation and access solution. Generalizable workflows can be designed to suit a variety of scenarios, but a successful digital preservation program will likely require additional professional and technical level staffing working in a collaborative, perhaps cross-divisional environment.

## Intermediate Term

*Recommendation: Include software in collections and acquisitions efforts*

There are many reasons to preserve software, not just as a cultural heritage object. Software plays an important role in big data research, in the computational sciences, and in the consumer market. Collecting and preserving software is, then, arguably, a central business case for libraries. For big data research and computational science, software is a necessary component to ensure that experiments are reproducible and that the scientific enterprise is fully open to scrutiny. A common refrain heard today from the open access scientific community is that the published article is only the research's advertising; the real contribution to new knowledge are the data produced as well as the tools and techniques used to analyze them to produce conclusions. Preserving software for purposes of research reproducibility would logically fall under the responsibility of a department other than HMD or PCM. Similarly, if the collections at NLM are to be the nation's most comprehensive library for medical information, commercial and consumer software, hybrid print-digital publications, and the like created for physicians, educators, scientists, and the public should be actively collected and preserved.

There are three main considerations for this work: (1) Appraisal and Selection; (2) Metadata and Description; and (3) Access and Use. Each suggestion will likely require extensive cross-division cooperation, although their implementations range in difficulty. Cooperation and communication, regardless of the recommendation, will be the most challenge aspect of software preservation at NLM.

### *Appraisal and Selection*

The most important recommendation for appraisal and selection for the preservation of software is that it needs to happen earlier in a software product's lifecycle. Regardless of the nature of the software or the division responsible for its development and care, concerns for preservation should begin as early as possible. For the residency's pilot preservation project, appraisal and selection happened many years after the initial development of the software. The piece of software for the pilot project, HowTo Grateful Med v. 4.0, was developed in 1988 but selected in 2016. This thirty-year gap presented several obstacles, including locating the software, creating a workstation to read the media on, and handling the software dependencies. Outside of the materials in the General Collections, it seems that the History of Medicine Division may be best suited to address the needs of software preservation for cultural heritage.

In order to perform appraisal and selection earlier in a software product's timeline, there are several strategies that NLM may want to consider. The first is to add software explicitly to its collections development strategy. Software will need to be added to the collections strategy with a strong and selective curatorial eye. Measured assessment of the impact and importance of a piece of software should occur as a part of appraisal and selection, as well as the technical difficulties associated with preserving that particular piece of software. In their article, "Born Digital: Guidance for Donors, Dealers, and Archival Repositories," by Gabriela Redwine et al, selection and appraisal criteria should include: (1) the general technical characteristics of the media, including format, file type, and extent; (2) the volume of digital materials associated with the piece of software; (3) the nature of the relationship between the piece of software and other materials in the collection; (4) available information about the context and

content of that software product, including RFP's and other technical documentation; and (5) particular preservation challenges for that piece of software.

To decide whether or not to collect a software product will require weighing these different attributes with the available resources at NLM. Each of these five points will play a role in how well a piece of software can be contextualized and how well it can be preserved for future access and use.

The development environment and products created for 21<sup>st</sup> century software potentially offers a more reusable infrastructure for libraries and archives to intervene in a product's lifecycle and ensure its selection and preservation. For example, social media-based development communities and tools such as GitHub offer centralized collection points. GitHub is a social media site that allows programmers to share code that is either open source or public domain. A Git repository also helps track the various versions of a piece of software. This recommendation could be particularly helpful because locating non-tangible software that lives only on NLM servers presents security and administrative difficulties that may be harder to address than the actual process of software preservation. One example of GitHub use at NLM is the People Locator project. Collecting software in this way could be a cost and time effective way to address the difficulties in locating NLM-created software.

#### *Metadata and Description*

Bibliographic metadata for HowTo Grateful Med v. 4.0 conforms to the current standards of descriptive metadata for the NLM cataloging and discovery systems in the NLM Digital Collections. Best practices and standards for descriptive and preservation metadata for software are still evolving as communities of interest are still in their developmental infancy, but there are additional descriptive, administrative, and technical data elements that could improve access to and help ensure the future safety of interactive digital assets. In the appendix of the final report, there is an example of HowTo Grateful Med's catalog records as well as a list of desirable data elements for new software assets. Tools such as FTK Imager and Bagit, currently being used for disk imaging and digital assets packaging in PCM and HMD, provide technical and administrative metadata in text files. These files include information on fixity as well as user-created metadata about title, version, and the process of creating the disk image or package. Metadata from these files could be leveraged by other pieces of the NLM preservation and access infrastructure.

Software Identification (SWID) Tags present interesting asset management potential for some of the software assets in the General Collections and software that NLM has developed over the years. SWID Tags are a part of ISO/IEC 1997-2:205, which established specifications for tagging software to collect, store, and process software identification information such as software publisher, product name, and version. While SWID Tags are not universally applied, capitalizing on this metadata may optimize the amount of information captured in software inventories while minimizing the amount of labor involved.

#### *Access and Use*

Constrained by the timelines of the residency program, NLM has developed rudimentary public access to HowTo Grateful Med v. 4.0 through the NLM Digital Collections website. Users must download the executables in their own local emulation of a MS-DOS system in order to run the software. Emulation is an enticing option for future access to software assets at NLM. While it is possible to create an emulation platform at NLM, exploring opportunities at several partnering organizations present interesting alternatives and benefits.

The Olive and Emulation as a Service (EaaS) are relatively new organizations that work to provide in-browser emulation. Both organizations have successfully partnered with cultural institutions. The Olive Archive currently partners with Standard University on the Cabrinety Collection, a historic collection of micro-computing software programs. EaaS partners with Yale University as well as Rhizome, a digital art museum in New York. These collaborations demonstrate a possible way forward as NLM conceives of ways to provide easier access to software assets in a way that suits current resources and staff expertise.

Another option is to work with the Internet Archive (IA) and its growing collections of operating system emulators and historical productivity software, although there are noting issues with IA's approach that would require vetting with NLM senior management. Unlike the Olive Archive and EaaS, using the Internet Archive's emulation services would not integrate access to software assets directly within NLM Digital Collections. Instead, a copy of the software would be stored at IA and NLM Digital Collections would redirect users to IA's services. Users could possibly become confused about what is and is not in the NLM Collection. However, NLM currently provides alternate access points to some of its digital objects through IA, and this framework might also be suitable for software assets. Rather than providing for in-browser emulation, the emulators on IA could provide a secondary access point to NLM software assets and improve usability over the current implementation.

## Long-Term

*Recommendation: Become actively involved and influence the future of software collection, description, and preservation*

These few long-term recommendations address issues that likely depend on actions outside of the control of NLM, including copyright, grant funding for computational research, and the services to support big data research. Ensuring long-term access to software is impacted by each of these issues, and it will be important to develop policies that position NLM in such a way that the issues can be adequately addressed based on the NLM mission. These issues also present spaces where NLM can become a thought leader and influencer of best practices.

The state of software preservation and copyright is in flux, and windows of opportunity for preservation are widening. For example, the United States Copyright Office recently made an exemption to copyright legislation, making it easier to preserve video games. Effectively, the Library of Congress granted an exemption to the circumvention of authentication firewall as long as the game content is stored on an individual player's computer. The implication is that if a network-based video game is discontinued, players will be able to modify their copy of the game so that they can continue playing on their own servers. This exemption is helpful because it also allows libraries, archives, and museums to copy and modify games so that those games can run after the server is shut down. In general, this law allows users more control over content than the Digital Millennium Copyright Act (DMCA) originally allowed.

NLM is still protected by the library rights given under Section 108 of the Copyright Law, allowed it to create preservation copies of copyrighted software. These rights do not allow NLM to break into software code to create an emulation nor allow the library to provide public access. However, software assets can at least be migrated off unstable and obsolete formats, which can fail without warning. NLM can make NLM-developed software accessible online, and it can make preservation copies of

copyrighted software. However, the institution should continue to actively monitor and report on copyright legislation specific to software.

Software also plays a growing role in academic research, particularly biomedical research. As the possibilities of Big Data become more apparent, researchers leverage data sets to better understand trends and relationships in health-related issues. While considerable work is being done to ensure that data sets will be shared and stored, the software that makes the data understandable and creates new knowledge from that data is still largely being ignored. In order to guarantee that the data-based research is reproducible and understandable in the future, the research community will have to address questions related to the preservation, sharing, and maintenance of research software. NLM can position itself to play a role in this future and should consider the possibilities of software preservation as an intrinsic part of preserving medical research and history.

Data management requirements associated with grant funding plays into this issue as well. NIH funded grants over \$500,000 require that researchers submit data management plans to ensure that the research data is well-cared for and shared with others, thus maximizing the potential impact of that funding. Because software is an important part of data collection and analysis, more funders and publishers are requesting access to researcher's source code as well as their data. If this trend continues, there is the distinct possibility that source code will be added to mandatory data management plans, including those at NIH. Preserving software created under NIH grants can be viewed as an equal responsibility for NIH as preserving research data.

## Conclusion

Software plays a central role in the history of computing in terms of data creation, processing, and access, teaching and learning, and consumer and business productivity. As NLM's mission evolves to include thinking digital first, NLM can have significant voice and provide leadership in this knowledge space. To do so will require additional resources or re-provisioning of existing ones. As a national library we should consider such work a central piece of our responsibility to the history of medical computing, specifically, and medicine writ large. Deputy Director Betsy Humphries often offers this inspirational and aspirational thought—"To do important work one needs to ask important questions." The challenges and opportunities offered by software preservation certainly embodies that call to action.

# Inventorying Software Developed at the National Library of Medicine: An NDSR Project Update

Blog Post from *The Signal: Digital Preservation*.<sup>57</sup>

The [National Library of Medicine](#) (NLM) has a fifty year tradition of developing software in-house for its own use and for the use of its patrons. As part of the National Digital Stewardship Residency (NDSR) program, I am [currently researching this history and devising a pilot preservation strategy](#) (PDF) for NLM-developed software found in NLM's archives and the offices of staff members.

The first step of this project was to create an inventory of these software programs, including that have been located as executable files or source code and those that have not. It acts as a checklist of software programs that need to be located and preserved while also acting as a reference when trying to understand NLM's history of software development. As opposed to cataloging, it is a more informal list intended for internal use. Creating this inventory, however, became a complicated process that highlighted some of the issues facing archivists working with software. Perhaps the biggest issue is deciding which software programs deserve their own inventory record when each program works co-operatively within a complex computing environment.



Figure 40: Some of the software related materials we are currently working with at NLM. Credit: Nicole Contaxis

<sup>57</sup> View Online at: <http://blogs.loc.gov/digitalpreservation/2016/01/inventorying-software-developed-at-the-national-library-of-medicine-an-ndsr-project-update/>

Before I began the inventory, I knew I would run into significant issues when choosing what data elements to include in an inventory record. While an inventory of books or papers can rely on pre-existing understandings of the material (i.e. “One letter, five pages long” or “one copy of *Do Androids Dream of Electric Sheep*”), it is difficult to create a software inventory because of how widely software structures can vary. Because [NLM has produced software from the 1960’s to the present day](#), an inventory would need to be able to accommodate the variations of software developed for batch-processing computers as well as software developed for mobile phones. While designing the inventory record was a challenge, it was not the only hurdle of the inventory process.

As I began to populate the inventory records, I was surprised to realize that it was not clear what did and what did not deserve its own inventory record. What constitutes a separate and individual piece of software within a complex computing system is not straightforward, especially when adequate documentation may not exist.

Here is an example to illustrate this challenge. Grateful Med, one of NLM’s hallmark software programs introduced in 1986, was an end-user friendly search system that allowed physicians and nurses to search NLM’s bibliographic data to help perform research and treat patients. It was such a notable piece of software at the time because it allowed end-users to search the data themselves rather than relying on computing specialists and truly considered the users point of view and experience. A new version of Grateful Med was created each year with updated vocabularies and new features.

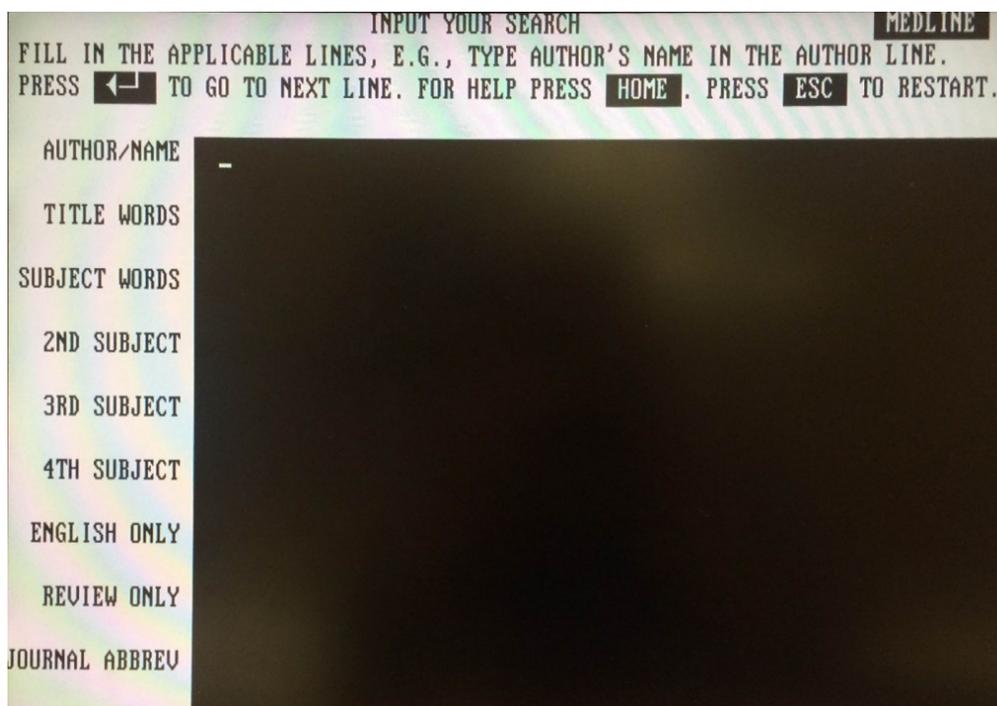


Figure 41: Grateful med v 4.0 Search Interface

Coach Metathesaurus was a software program that was developed in 1991, in part, to serve as one of these new features of Grateful Med. Designed to assist end-users with controlled medical vocabularies, it was meant to hook into Grateful Med seamlessly and provide assistance to the user when the user’s search queries returned inadequate responses. Although the piece of software was fully developed and tested in NLM’s Reading Room, it was not implemented across all versions of Grateful Med.

Here, the question becomes, does Coach Metathesaurus deserve its own inventory record? In this instance, I decided that it does. Because it was not implemented across all versions of Grateful Men, I could not confirm that the functionality and history of Coach Metathesaurus would be included in a preserved version of Grateful Med, and it would also be unlikely that anyone would know to search for it.

However, what if Coach Metathesaurus was implemented in all versions of Grateful Med? In this scenario, would Coach Metathesaurus deserve its own inventory record? It is at this line of questioning that we can begin to see the difficulties of delineating between individual pieces of software within a complex system, one of the major obstacles to performing an accurate software inventory.

As I considered this question, both for Coach Metathesaurus and for similar software programs, I established two basic guidelines. The first is: what information may be lost or gained by including a new inventory record for this piece of software. If, as with Coach Metathesaurus, information or functionality would be lost without an additional inventory record, I would make a new record. On the other hand, if information or functionality could be included in a larger record that adequately reflected the history of that piece of software, it would be included in the larger record.

The second guideline is less practical but still quite important: when deciding if something deserves its own inventory record, it is helpful to consider the ways in which people experienced that software. For the sake of these decisions, I separated these experiences into roughly two categories: the experiences of the development team and the experiences of the user community. Although experiences within both of these categories can vary wildly, these categories remain helpful when considering what information developers can access and what information users can access. For example, the user did not know that Coach Metathesaurus was a separate piece of software that performs different functions, but the developer would.

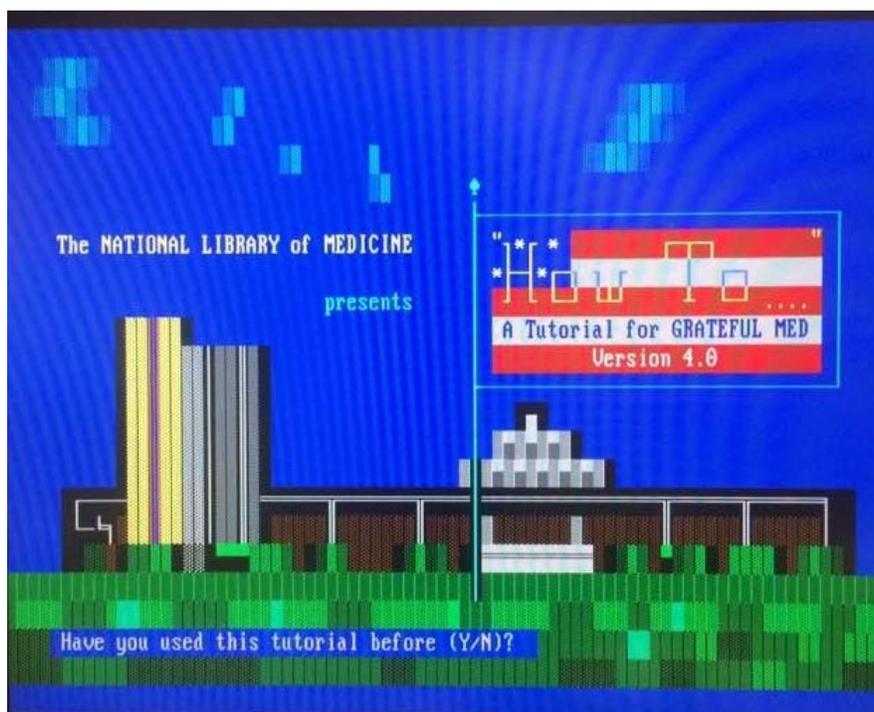


Figure 42: A tutorial program designed to help user's search with Grateful Med

Contemporaneous documentation is necessary when attempting to consider the developer's and user's experience of a piece of software. There was clear documentation about how to use Grateful Med, as it was built for external users, rather than for in-house use. However, documentation of the development process itself can be difficult to locate because internal communications, like emails and memos, may be long lost. This lack of documentation is exacerbated when software is developed for internal use rather than external. As such, the archivist working with software intended for internal use needs to perform a significant amount of intellectual labor in order to document the how and why of software development, if that knowledge itself has not already been completely lost to time.



*Figure 43: Nicole Contaxis presenting research to NLM staff members. Credit: Ben Petersen.*

Describing software and drawing boundaries around what does and what does not constitute an individual piece of software is not an easy endeavor. At this point at NLM, decisions about what pieces of software deserve their own inventory record are made on a case-by-case basis. These decisions are made with respects to the developer's experience of the product, the user's experience of the product, the available documentation, and the practical needs of NLM. While these guidelines provide a useful roadmap, they require a significant amount of intellectual labor on the archivist's part, resulting in complex answers. Some of the major obstacles hindering software archival practice lie in creating boundaries between individual pieces of software within computing systems and in respecting both the development and use of software within the description process. Thankfully, at NLM, we've been able to recognize these concerns, making the inventory process of our in-house developed software much easier to understand.

# The Astrophysics Source Code Library

Blog Post from *The Signal: Digital Preservation*<sup>58</sup>

ASCL.net  
Astrophysics Source Code Library

Making codes discoverable since 1999

Home About Resources Browse Submissions News Forum Dashboard

## Browsing Codes

Results 1-100 of 1249 (1240 ASCL, 9 submitted)

Previous 1 2 3 4 5 6 7 8 9 10 11 12 13 Next

Order: Title ▲, Date ▼; Mode: Abstract, Compact; Per Page: 50, 100, 250, All

[ascl:1102.023] **21cmFAST: A Fast, Semi-Numerical Simulation of the High-Redshift 21-cm Signal**  
Mesinger, Andrei; Furlanetto, Steven; Cen, Renyue

21cmFAST is a powerful semi-numeric modeling tool designed to efficiently simulate the cosmological 21-cm signal. The code generates 3D realizations of evolved density, ionization, peculiar velocity, and spin temperature fields, which it then combines to compute the 21-cm brightness temperature. Although the physical processes are treated with approximate methods, the results were compared to a state-of-the-art large-scale hydrodynamic simulation, and the findings indicate good agreement on scales pertinent to the upcoming observations ( $> \sim 1$  Mpc). The power spectra from 21cmFAST agree with those generated from the numerical simulation to within 10s of percent, down to the Nyquist frequency. Results were shown from a 1 Gpc simulation which tracks the cosmic 21-cm signal down from  $z=250$ , highlighting the various interesting epochs. Depending on the desired resolution, 21cmFAST can compute a redshift realization on a single processor in just a few minutes. The code is fast, efficient, customizable and publicly available, making it a useful tool for 21-cm parameter studies.

[ascl:1505.015] **2dfdr: Data reduction software**  
AAO software team

Figure 44: Example submissions for the Astrophysics Source Code Library: <http://ascl.net/code/all>

On April 12<sup>th</sup> 2016, [Alice Allen](#), editor of the [Astrophysics Source Code Library](#), came to the [National Library of Medicine](#) to speak with [National Digital Stewardship Residency](#) participants, mentors and visitors about the importance of software as a research object and about why the ASCL is a necessary and effective resource for the astronomy and astrophysics academic communities.

Astrophysicists and astronomers frequently write their own code to do their research, and this code helps them interpret and manipulate large data sets. These codes, as an integral part of the research process, are important to share for two reasons: (1) they increase the efficiency of work by allowing code to be re-used and (2) it helps ensure the transparency of scientific research.

Yet, difficulties persist when it comes to encouraging researchers to share source code, regardless of the benefits. Allen talked about how researchers are reluctant to share code that may be “messy” and that creating this source code library requires community engagement and change management. She spoke about studying the impact of non-traditional scholarly outputs, like code, and the issues of scholarly publishing. Allen showed how ASCL has helped allow journal authors cite code, which had been a far more difficult procedure earlier. The ASCL assigns unique identifiers, called ASCL IDs, so that future

<sup>58</sup> View Online at: <http://blogs.loc.gov/digitalpreservation/2016/04/the-astrophysics-source-code-library/>

academics can cite that code, even if that code is not featured in a journal article. Every major astronomy journal accepts ASCL IDs in citations.



*Figure 45: Image from the Visible Human Project at the National Library of Medicine*

The discussion turned to the difficulties of grant-based funding. The ASCL is basically unfunded, and all labor, including Allen's, is voluntary. While talking about other code libraries that have lost funding and closed, Allen talked about how grant-funding, which runs on two- to five-year cycles, does not provide enough time to fully engage a community with a resource, regardless of how well that resource is designed, implemented and managed. Funding, as a universal source of concern, was a common point of interest, even for attendees without experience working with software or code.

The session included a tour of [Visual Human Project](#), which is an NLM project that collects extensive data on a male and female cadaver, allowing artists and researchers to visualize that data in new and exciting ways.

# MEDLARS I & GRACE: The Early Mainframe Experience

Blog Post from *Circulating Now: From the Historical Collections of the World's Largest Biomedical Library*<sup>59</sup>

Providing access to bibliographic data has long been a part of the National Library of Medicine (NLM) mission. Through a variety of means, NLM has publicized and shared medical research nationally and internationally since its inception. In 1879, it published the first edition of *Index Medicus*, a list of current medical literature, and in the 1940s, it began to experiment with early mechanized systems. The advent of computing brought a series of new techniques that dramatically altered the scope of the NLM goals and actions, and these changes began in earnest in 1960 with the formal proposal of [MEDLARS](#)—the Medical Literature Analysis and Retrieval System.



Figure 46: John Smith engages a team of computer specialists at the National Library of Medicine, ca. 1965, National Library of Medicine #10164816

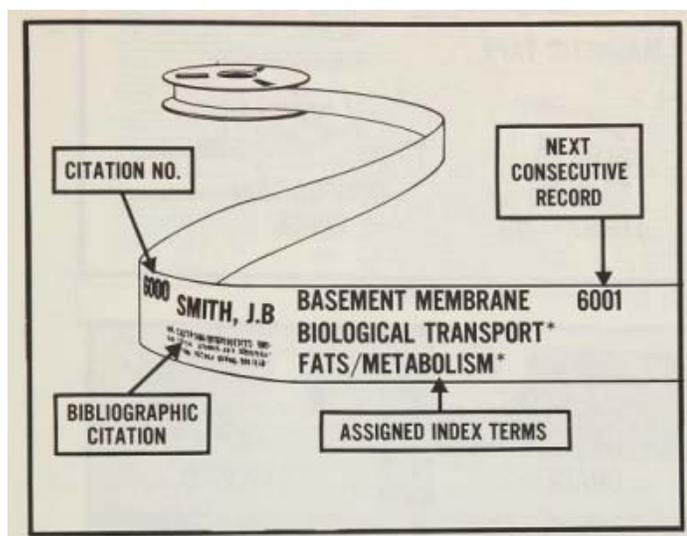
MEDLARS [began](#) with a request for proposals for a “study to investigate the feasibility of using electronic digital computers for the publication of the *Index Medicus* and also as a basis for the construction of an efficient reference and bibliographic service.” MEDLARS was thus designed to kill two birds with one stone. It was meant to assist in the publication of *Index Medicus* and other medical bibliographies as

---

<sup>59</sup> View Online at: <https://circulatingnow.nlm.nih.gov/2016/02/25/medlars-i-grace-the-early-mainframe-experience/>

well as provide for on-demand searches of the computerized data. In 1962, the planning and designing stages were completed and construction could begin.

Getting the system operational and maintaining it required a significant amount of labor and planning. For example, data on journals and journal articles had to be input into MEDLARS using [punched paper tape](#), and then this data was added to the [magnetic tapes](#) that were used to store the majority of the bibliographic data. New bibliographies and on-demand searches would be created using these magnetic tapes.



At this point in history, room-sized mainframe computers that employed batch-processing dominated computing. MEDLARS, which used a Honeywell-800 mainframe computer, [followed this model](#). Batch-processing means that the computer was given a series of jobs and then the users would return, sometimes days later, for the results. People had to schedule their time around when the computer was available, and it could take several attempts to have a particular job, or program, run successfully. It was a [time-consuming process](#). However, from the point of the view of the medical researcher, the process of submitting a search request was relatively simple, even if results were not always received quickly enough. A search request would be mailed via a [form](#) that detailed the researcher's interests, needs, and contact information. While researchers did not need to pay for searches, they were required to respond to the library about the quality of search results they received. These responses helped inform strategic and technological changes.



1969. Although it suffered from malfunctions throughout its operational history, GRACE was groundbreaking and is now housed at the Smithsonian.



*Figure 47: Graphic Arts Composing Equipment (GRACE) receives maintenance from Tom Rush, a contract technician, National Library of Medicine #101656559*

Coinciding with the creation of the MEDLARS system, NLM began providing its bibliographic data via magnetic tape to other medical libraries throughout the United States and eventually internationally. These medical libraries became [MEDLARS Centers](#) that could process MEDLARS searches in the same way that NLM did. Search analysts and librarians from these MEDLARS Centers would be sent to NLM to receive training. In this way, MEDLARS was not simply a computer or a piece of software; it became an international information-sharing network that relied on human co-operation as well as unique computing power.

Throughout its operation, its ability to fulfill its design goals depended on the amount of demand placed on the system. By the late-1960s, MEDLARS was no longer able to handle the workload and provide the services that users had come to expect. It could take two weeks or more to fulfill a MEDLARS request. This increased workload has been traced to expansions in scientific literature, medical research, and medical specializations throughout the 1960s. At this point, NLM began to plan for MEDLARS II, which would replace MEDLARS in 1975.

# MEDLARS II: MEDLINE & Instantaneous Search

Blog Post from *Circulating Now: From the Historical Collections of the World's Largest Biomedical Library*<sup>60</sup>

MEDLARS I, as described in [my previous post](#), was a great step forward in providing access to bibliographic data and facilitating biomedical research, nationally and internationally. However, by the mid-1960's, customer demand for MEDLARS I services was far outstripping capacity. Planning a next-generation MEDLARS II began in 1966.

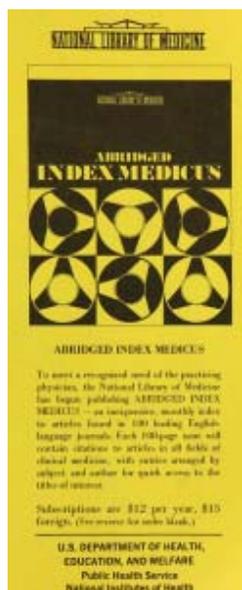


Figure 48: *Abridged Index Medicus Brochure, ca. 1980*

MEDLARS II and [MEDLINE](#) are now synonymous, but that was not always the case. MEDLINE is an abbreviation for MEDLARS Online, but MEDLARS II was not initially designed to be an online search system. At first, it was intended to be an improved version of the decentralized batch processing system that MEDLARS I employed. By implementing an online, interactive search system rather than a batch processing system, NLM empowered users to interact directly with the computer, and reconceptualized how computerized search could be done in a library. Three separate projects led to this service evolution and the creation of MEDLINE as it was implemented: the initial proposed design for MEDLARS II, the [State University of New York \(SUNY\) Biomedical Communications Network](#) in 1968, and [AIM-TWX](#), an experimental project at NLM that established interactive search of the *Abridged Index Medicus* (AIM).

The initial proposed design for MEDLARS II followed the model of MEDLARS I, as noted in the design specifications created by the Auerbach Corporation, under contract at NLM. The initial design proposed a faster batch processing computer in order to minimize search times. This computer would rely on the same model established for MEDLARS I and would not provide for online or instantaneous searching of

<sup>60</sup> View Online at: <https://circulatingnow.nlm.nih.gov/2016/03/30/medlars-ii-medline-instantaneous-search/>

citation data. However, before this design could be fully realized or implemented, [NLM Director Martin M. Cummings](#), commissioned the Systems Development Corporation (SDC) in Santa Monica, CA to create a MEDLARS II that could provide for online search and, thus, MEDLINE was born.

The SUNY Biomedical Communications Network and AIM-TWX were important models for MEDLINE. These two projects allowed for interactive, online search of biomedical information, which shortened search times to minutes instead of weeks. More than simply improving the efficiency of the MEDLARS I batch-processing system, these projects relied on an entirely new model and helped change what computerized bibliographic search could mean.

The SUNY and AIM-TWX projects themselves built on earlier work with [computerized communication](#) that paved the way for this reconceptualization. Networked computing existed far before the creation of the Internet, which is itself a network of previously existent computer networks. As early as 1933, [Telex](#) used teleprinters to deliver messages in Nazi Germany. Unlike telegraphs, Telex used less expensive phone lines to communicate. Rather than relying on a point-to-point communication, Telex was a network of teleprinters that aided in communication. However, it was another thirty years until this networked-style of communications was applied to computers with the advent of time-sharing computers. Time-sharing computers allowed multiple people to use a computer via individual terminals. It was a much faster system than batch-processing computers and allowed for remote access to computing resources. In the early years, these types of computers were most frequently used in military and academic settings. MEDLINE, as it existed in the 1970s, relied on a time-sharing computer.



*Figure 49: Yvonne Scott dials in to MEDLINE, ca. 1971*

These types of innovations are important because they formed the foundations of [online, interactive computerized searching](#). In 1968, SUNY began to offer online searching of MEDLARS data. Using special terminals, the SUNY network allowed users to search for citation data in real time. Davis McCarn, an NLM staff member who contributed to AIM-TWX, MEDLINE, and eventually Grateful Med, studied the SUNY network and presented it to NLM as a better, more efficient, and more user-friendly option than the decentralized batch processing method employed in MEDLARS I. After demonstrating the strengths of the SUNY project, McCarn began to work on AIM-TWX, the first online search system for NLM.

AIM-TWX is an abbreviation of *Abridged Index Medicus* – Teletypewriter Exchange System (TWX). TWX terminals, rather than phone lines, were used because they had already been installed in 500 libraries as part of the interlibrary loan system. Connecting the TWX terminals to a time-sharing computer at the Systems Development Corporation now made online searching of the *Abridged Index Medicus*, the abbreviated version of the full MEDLARS citation database, possible. AIM-TWX also allowed up to thirty users to search at the same time, increasing overall access to the database. The full one million citations in MEDLARS were too much for the remote access system to handle. In 1971, the full MEDLARS database became available as MEDLINE, and its [success was notable](#).

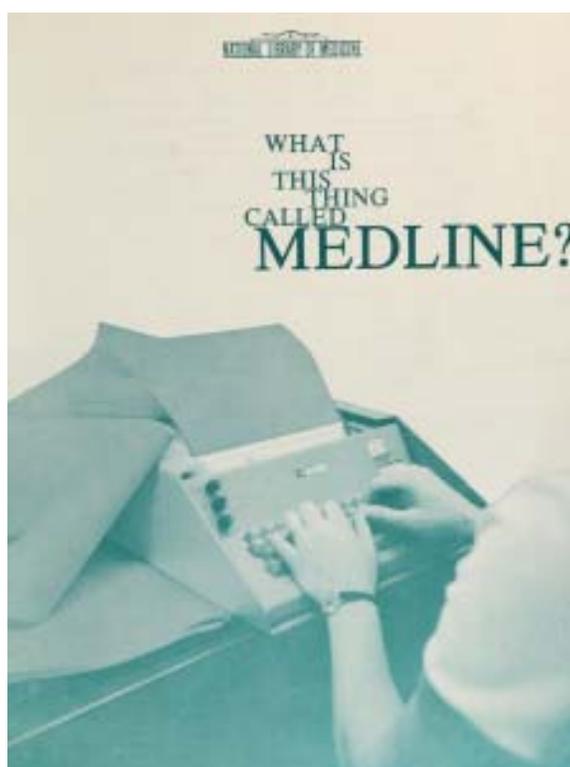


Figure 50: *What is this thing called MEDLINE?* Pamphlet, 1972

User satisfaction with online search was so compelling that [Director Cummings](#) had MEDLARS II redesigned to follow the example of the SUNY Biomedical Communications Network and AIM-TWX. MEDLINE went online in 1971 and fully replaced MEDLARS I in 1975. While the initial version of MEDLINE only facilitated access to one database, NLM staff members worked to increase access to other databases over time. While continually adding new databases to the service proved to be a challenge, with MEDLINE, NLM was able to change how they provided access in a revolutionary way.

Instantaneous search was a huge step forward for NLM and biomedical research, and marked the advent of a new era at NLM. Now, instead of taking weeks, a search could take minutes.

Yet, NLM did not stop growing, changing, and inventing, and the creation of new technologies was just around the corner. Grateful Med would facilitate access to MEDLINE in new, user-friendly ways, and its creation will be featured in depth in a post next month.

# Grateful Med: Personal Computing and User-Friendly Design

Blog Post from *Circulating Now: From the Historical Collections of the World's Largest Biomedical Library*<sup>61</sup>

Grateful Med was an NLM-developed software program that was intended to expand and ease access to the NLM databases, including MEDLINE. Supported and in use between 1986 and 2001, this piece of software is part of an important trend in the history of computing and the history of NLM itself. Before explaining the history of Grateful Med, I need to explain first what purposes Grateful Med served, how it interacted with the rest of the NLM infrastructure, and how it relates to the history of computer and software development.

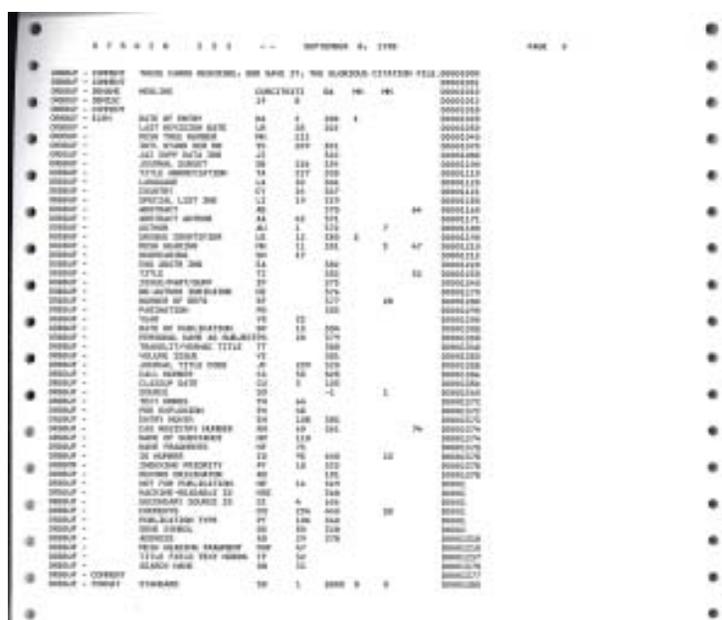


Figure 51: Source code of ELHILL featuring the comment, "These cards describe, God save it, the glorious citation file." 1995, National Library of Medicine

Grateful Med functioned as the user-friendly front-end to ELHILL. ELHILL was the retrieval system for MEDLINE searches, and although it has not been explicitly mentioned yet in this series of blog posts, it was an integral part of the NLM infrastructure. Originally developed in 1969, various iterations of the program played a part in AIM-TWX, MEDLINE, and eventually Grateful Med. However, ELHILL was not easy or intuitive to use, so in 1984, newly appointed NLM Director Donald A.B. Lindberg MD began a project to create a user-friendly system for people to search NLM databases. Based on the idea that NLM data should be searchable by non-trained, non-specialist researchers, Grateful Med opened up NLM databases to a wider community, although the searches themselves were still powered through ELHILL as they were before.

<sup>61</sup> View Online at: <https://circulatingnow.nlm.nih.gov/2016/04/28/grateful-med-personal-computing-and-user-friendly-design/>

Although Grateful Med was designed to be as user-friendly as possible, many users still required training, even if this training was not nearly as extensive as the training that medical librarians received to work with previous search systems. To provide Grateful Med training, NLM created computer tutorials and other resources that would walk a user through the variety of tools that the software provided. The tutorial for Grateful Med was called "HowTo Grateful Med," and it demonstrated how computer networks worked, provided example searches, and explained how best to construct search queries using medical vocabularies and Boolean logic. NLM publications complimented these tutorials and provided advice and updates on using Grateful Med. One such publication, *Gratefully Yours*, ran from 1990 until 1998.

Grateful Med was a successful and important advancement to provide access to NLM resources, but perhaps the first thing people notice about it today is its name. Federal institutions, after all, aren't known for naming their products after bands, particularly bands like [Grateful Dead](#). The majority of proposed names were more predictable, including "MICROMEDLARS," "PCMEDLARS," and "NLMSEARCH." These names generally alluded to the fact that the program ran on personal computers, known at the time as microcomputers. Several suggested names stick out, including "TALKING HEADS," "DISCOMED," and "GRATEFUL MED." TALKING HEADS also referred to a [band](#), and DISCOMED referred to disco music. However, as visible in the image of a memo below, Director Lindberg chose Grateful Med because, "It is just too good to pass up." And, thus Grateful Med was given a pleasantly unexpected name.

ROUTING AND TRANSMITTAL SLIP		Date
		11/25/85
TO: (Name, office symbol, room number, building, Agency/Post)		Initials Date
1.	<i>John Audena</i>	
2.		
3.		
4.		
5.		
Action	File	Note and Return
Approval	For Clearance	Per Conversation
As Requested	For Correction	Prepare Reply
Circulate	For Your Information	See Me
Comment	Investigate	Signature
Coordination	Justify	
REMARKS		
<p style="text-align: right;">DEC 03 REC'D</p> <p>no harm in new microform based user interface to Medline. Let's use <u>GRATEFUL MED</u>. It's just too good to pass up. MicroMedians isn't accurate. You have not yet done a micro-based version of Medline.</p>		
DO NOT use this form as a RECORD of approvals, concurrences, disposals, clearances, and similar actions		
FROM: (Name, org. symbol, Agency/Post)		Room No.—Bldg.
<i>Donald A. B. Lindberg, M.D.</i> Director, NLM		2E17 38
		Phone No.
		495-6221
5041-102	OPTIONAL FORM 41 (Rev. 7-76)	
☆ GPO : 1983 O - 381-529 (317)	Prescribed by GSA FPMR (41 CFR) 101-11.206	

Figure 52: Memo in which Director Lindberg picked the name "Grateful Med," 1985, National Library of Medicine

Grateful Med's initial and continued development interacted with a variety of other NLM projects like ELHILL, and its success depended on the success of many other aspects of the NLM infrastructure, both technological and administrative. In this way, Grateful Med didn't revolutionize the services that NLM provided on its own; instead, it was part of a larger array of projects that aimed to expand access to NLM resources. Grateful Med, as the end-user interface, was simply a very visible part of these larger changes.



Figure 53: Grateful Med v 4.0 from 1988. Credit: Nicole Contaxis, 2016

Concerns about making computers and networks more user-friendly did not exist only at NLM. Research groups from the 1960s forward have investigated how humans interact with computers. Some examples of the more notable early work include Douglas Englebart's at the [Augment Research Center](#) at Stanford University, where the [computer mouse was pioneered](#), as well as the research at [XEROX PARC](#) toward the "office of the future." With the growth of personal computers in the 1980s, discussion on the relationship between humans and computers expanded, as exhibited by the coining of the term "human-computer interaction" in 1983 with the publication of *The Psychology of Human-Computer Interaction*. Research at corporations, government agencies, and academic institutions began to address issues in technology design with techniques and conceptual frameworks from ethnography and psychology.

It is not a coincidence that this type of research grew as personal computers became more popular. Personal computing made it financially advantageous to have user-friendly technologies. Prior to the creation of personal computers, most computing was done in academic, corporate, or military settings where employers hired specialists to interact with the computer. After personal computers began to be more popular, corporations had a financial incentive to address usability concerns, and in this way, the possible uses for computers began to expand. Grateful Med, as a software program that was built for

personal computers, was a part of this larger historical narrative. It was conceived, designed, and implemented with usability and the reality of personal computing in mind.



Figure 54: Staff configuring Grateful med, ca. 1985, National Library of Medicine #101648264

Grateful Med was an immensely successful project and use of the software product increased quickly, both domestically and internationally. For example, the number of authorized users increased by over 50% in under a year in 1987. As use of Grateful Med increased, so did overall access to NLM resources. BY 1994, Grateful Med accounted for 80% of the over four million ELHILL searches of NLM databases. It was even featured on an episode of the television show *Doogie Howser, MD* in 1991.

In 1996, NLM launched Internet Grateful Med to provide access through the internet rather than other computer networks, which helped further increase Grateful Med's popularity. The internet dramatically altered the shape of NLM services and infrastructure. As PubMed and other internet-based resources were developed, NLM gradually transitioned away from earlier products. Grateful Med was [officially retired](#) in 2001.

EPILOGUE: As a part of my time as the National Digital Stewardship Resident at the National Library of Medicine, I have been working to preserve a piece of NLM-developed software. We chose to preserve "HowTo Grateful Med" because of the importance of Grateful Med and the difficulties associated with preserving Grateful Med itself. Grateful Med is a piece of a larger infrastructure and requires that larger infrastructure to operate properly, but "HowTo Grateful Med" is self-contained and easier to re-create on modern computers. "HowTo Grateful Med" contains example searches on Grateful Med so it can give users an idea of what Grateful Med was like while avoiding the technical difficulties of preserving Grateful Med itself. The goal of the residency is to make the software program available through the NLM Digital Collections within the coming months.

# Appendix A: Located Software

Hand-out Describing the Historical and Technical Characteristics of Each Piece of Software Considered for Preservation

Software Name	Historical Relevance	System Requirements	Media	Other Dependencies
Grateful Med v 1.0 for Windows	Although there are differences between each of these versions of Grateful Med, the historical relevance of each version is roughly equivalent. Grateful Med represents the bleeding edge of end-user search and networked infrastructure in the late 20 <sup>th</sup> century. With extensive user studies and surveys, Grateful Med paired innovative communications research with diligent attention to user needs and user-friendly design. Preserving Grateful Med would mean preserving an important piece of technological and design history.	Windows 3.1 or higher or Windows 95 or Windows NT; Hard disk with 6MB free disk space; modem with telephone or Internet Access (see other dependencies)	Three 3.5" floppies	The most challenging dependencies for Grateful Med are uniform across its versions. Although the exact files they are dependent on may vary, the architecture of the software remains consistent, producing consistent external dependencies. Because Grateful med was a search platform for data held on external computers, we need to address how Grateful Med would function without those external computers and the data they held. While we have discussed creating a dummy data set so that future users could better understand how Grateful Med searched its datasets, the creation of that dummy data presents immense difficulties. We would need to accommodate older and outdated data formats and MeSH. This is a significant, but not impossible, obstacle.
Grateful Med v 2.0 for PC		IBM or PC family compatible; DOS v. 2.0 or higher, 256K RAM or more; Hayes Smartmodem or fully compatible modem (see other dependencies)	Three 5.25" floppies; One 3.5" floppy	
Grateful Med v 3.0 for PC			Three 5.25" floppies; One 3.5" floppy	
Grateful Med v 4.0 for PC			Three 5.25" floppies; One 3.5" floppy	
CHEMLEARN	Part of a quartet of tutorials produced and packaged with SIS, LO, and contractors, CHEMLEARN is a microcomputer (i.e. personal computer) based training for CHEMLINE and ChemID. The tutorial itself is not particularly historically relevant, but CHEMLINE and ChemID, as	IBM-PC, PC-XT, PC-AT, PS/2 and other compatible computers; 512 RAM; DOS 2.0 or higher	Three 5.25" floppies; One 3.5" floppy	Outside of system requirements, there are no external dependencies in order for CHEMLEARN to function in a meaningful way for future users. All we would need to do is save the bits and later agree on a way to provide access to this self-contained software program.

	<p>some of the early information resources provided through networked communication at NLM, are. In this way, the tutorial serves as a demonstration and emblem of more historically important products. It is worth noting, however, that CHEMLINE is not technically "software." Instead, CHEMLINE is a database, discontinued in 1998 after the advent of the Internet. While the search apparatus for CHEMLINE is technically software, CHEMLINE itself is more notable as a database than as an executable. ChemID, on the other hand, remains operational and is accessible through the Internet. ChemID allows users to search for particular chemicals by name, registry number, structures, classification code, toxicity, physical property, or classification code. In this way, ChemID is services that relies directly on software and CHEMLEARN can act as a demonstration of ChemID .</p>			
ELHILL LEARN	<p>Part of a quartet of tutorials produced and packaged with SIS, LO, and contractors, ELLHILL LEARN is a microcomputer (i.e. personal computer) based training for ELHILL, the search and retrieval software that supported the majority of MEDLARS databases. The ELHILL LEARN tutorial pre-dates CHEMLEARN, TOXLEARN, and MEDTUTOR, and gives a wider ranging view of NLM's information resources. Out of all the tutorials, ELHILL LEARN presents the most information</p>	<p>IBM-PC, PC-XT, PC-AT, PS/2 and other compatible computers; 512 RAM; DOS 2.0 or higher</p>	<p>Three 5.25" floppies; One 3.5" floppy</p>	<p>Outside of system requirements, there are no external dependencies in order for ELHILL LEARN to function in a meaningful way for future users. All we would need to do is save the bits and later agree on a way to provide access to this self-contained software program.</p>

	about how NLM created, stored, and transmitted its data, and it would be the best choice for preservation out of this set of tutorials.			
TOXLEARN	Part of a quartet of tutorials produced and packaged with SIS, LO, and contractors, TOXLEARN is a microcomputer (i.e. personal computer) based training for TOXLINE. The tutorial itself is not particularly historically relevant, but TOXLINE, as one of the early information resources provided through networked communication at NLM, is. However, ELHILL LEARN represents a much larger view of NLM's information resources and seems to be a better choice for preservation.	IBM-PC, PC-XT, PC-AT, PS/2 and other compatible computers; 512 RAM; DOS 2.0 or higher	Three 5.25" floppies; One 3.5" floppy	Outside of system requirements, there are no external dependencies in order for TOXLEARN to function in a meaningful way for future users. All we would need to do is save the bits and later agree on a way to provide access to this self-contained software program.
MEDTUTOR	Part of a quartet of tutorials produced and packaged with SIS, LO, and contractors, MEDTUTOR is a microcomputer (i.e. personal computer) based training for MEDLINE. The tutorial itself is not particularly historically relevant, but MEDLINE as some of the early information resources provided through networked communication at NLM, is. MEDTUTOR only demonstrated MEDLINE, unlike ELHILL LEARN, which demonstrates a larger swath of MEDLARS databases. MEDTUTOR goes further into detail about MEDLINE, its data structures, and search strategies. However, ELHILL LEARN represents a much larger view of NLM's information resources and seems to be a better choice for preservation.	IBM-PC, PC-XT, PC-AT, PS/2 and other compatible computers; 512 RAM; DOS 2.0 or higher	Three 5.25" floppies; One 3.5" floppy	Outside of system requirements, there are no external dependencies in order for MEDTUTOR to function in a meaningful way for future users. All we would need to do is save the bits and later agree on a way to provide access to this self-contained software program.

<p>“How to” Grateful Med v. 4.0 for DOS</p>	<p>Like the tutorials above, the “How to” program is not historically important in itself. However, it does demonstrate Grateful Med, its purpose, and its use for the future user without the external dependencies of Grateful Med itself. While the program is several hours long and I have not yet been able to go through every aspect of it, the program describes how to search on Grateful Med, how Grateful Med was programmed and how to make adjustments for expert users, and the overall look and feel of DOS computer programs.</p>		<p>Three 5.25” floppies; One 3.5” floppy</p>	<p>Outside of system requirements, there are no external dependencies in order for “How to” to function in a meaningful way for future users. All we would need to do is save the bits and later agree on a way to provide access to this self-contained software program.</p>
---------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	--------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Appendix B: Potential Data Elements for Software Assets at NLM

Covering Descriptive, Administrative, and Technical Metadata Elements

## Descriptive

- Identifier
- Title
- Version
- Creator
- Description / Abstract
- Publisher
- Date of beginning of development
- Date of beginning of use
- Date of end of active support
- Language
- Rights information
- Relation with other titles

## Process History

- Administrative
  - Checksum/fixity
  - Selection information
  - Preservation level information
  - Rights information
  - Version/format/usage (access, master, etc.)
- Technical
  - Original physical media (if applicable)
    - Manufacturer
    - Model name
    - Format (3.5" floppy, etc.)
    - Condition note
  - Preservation Event
    - Exact nature of event
      - Either pulling from obsolete media, NLM Git, or NLM server
    - Agent – person who performed this event
    - When it occurred
    - Environment (What was used):
      - Software
        - Name
        - Version

- Operating System
    - Role
    - Description
  - Hardware
    - Name
    - Version
    - Operating system
    - Role
    - Description
- Digitized file format
  - Format name
  - Format version
  - Registry name
- File (master, access, player)
  - Filename / location
  - File size
  - Checksum / fixity (type, value, date)
  - Language

# Appendix C: Current Metadata for Software Asset

```
- <oai_dc:dc xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
  <dc:title>HowTo Grateful Med</dc:title>
  - <dc:creator>Lister Hill National Center for Biomedical Communications.
    </dc:creator>
    <dc:subject>Grateful Med.</dc:subject>
    <dc:subject>Information Storage and Retrieval</dc:subject>
  - <dc:description>An interactive tutorial program on how to search and use Grateful Med.
    </dc:description>
    <dc:publisher>
    Bethesda, Md. : National Library of Medicine, Lister Hill Center, 1989
    </dc:publisher>
    <dc:date>1989</dc:date>
    <dc:format>Software, multimedia</dc:format>
    <dc:identifier>nlm:nlmuid-101679914-sw</dc:identifier>
    <dc:identifier>101679914</dc:identifier>
    <dc:identifier>http://resource.nlm.nih.gov/101679914</dc:identifier>
    <dc:language>English</dc:language>
    <dc:rights>NLM believes this item to be in the public domain.</dc:rights>
</oai_dc:dc>
```

# Appendix D: Screenshots of Software Asset Resource Page


**U.S. National Library of Medicine**  
*Digital Collections*

[About](#)  
[Help](#)  
[Web Service](#)

---

### HowTo Grateful Med

Contributor(s): Lister Hill National Center for Biomedical Communications.

Publication: Bethesda, Md. : National Library of Medicine, Lister Hill Center, 1989

Language(s): [English](#)

Format: Software, multimedia

Subject(s): [Grateful Med](#).  
[Information Storage and Retrieval](#)

Abstract: An interactive tutorial program on how to search and use Grateful Med.

Rights: NLM believes this item to be in the public domain.

Extent: 4 computer discs : 3 1/2-5 1/4 in.

Edition: Version 4.0

NLM Unique ID: [101679914 \(See catalog record\)](#)

Permanent Link: <http://resource.nlm.nih.gov/101679914>

System Note(s): System requirements: IBM PC; 256K RAM or more; DOS version 2.0 or higher; Hayes Smart modem; 360K double-sided, double-density diskette drive; 1.5MB hard disk space (optional)  
 Disk characteristics: floppy disk



- Installation Files (ZIP)
- Screenshots (ZIP)
- Metadata (Dublin Core)

---

[Copyright](#), [Privacy](#), [Accessibility](#), [Site Map](#), [Viewers and Players](#)  
 U.S. National Library of Medicine, 8600 Rockville Pike, Bethesda, MD 20894  
 National Institutes of Health, Health & Human Services  
[Freedom of Information Act](#), [Contact Us](#)



# Appendix E: Selected Resources on Software Preservation Practices

- Bearman, David. *Collection Software: A New Challenge for Archives & Museums*. Pittsburgh: Archives & Museum Informatics, 1987. Located online:  
[http://www.archimuse.com/publishing/bearman\\_col\\_soft.html](http://www.archimuse.com/publishing/bearman_col_soft.html)
- Delve, Janet, and David Anderson. *Preserving Complex Digital Objects*. London: Facet, 2014.
- Engel, Deena, and Glenn Wharton. "Source Code Analysis as Technical Art History." *Journal of the American Institute for Conservation* 54.2 (2015): 91-101. Web.  
<http://authenticationinart.org/pdf/literature/1945233015Y.pdf>
- Engel, Deena, Mark Hellar, Matthew Kirschenbaum, Alex Cooper, Lincoln Schatz, James Murray, and Aaron Straup. *TECHNOLOGY EXPERIMENTS IN ART: Conserving Software-Based Artworks*. National Portrait Gallery and Smithsonian American Art Museum, Washington, DC. Smithsonian Institution, 2014. Web. 2015. <http://www.si.edu/tbma/symposiums>
- Lee, Jin Ha, Rachel Ivy Clarke, and Andrew Perti. "Empirical Evaluation of Metadata for Video Games and Interactive Media." *Journal of the Association for Information Science and Technology* 66.12 (2015): 2609-625. Wiley, 30 Jan. 2015. Web. 1 Mar. 2016.  
<http://onlinelibrary.wiley.com/doi/10.1002/asi.23357/full>
- Malone, James, Andy Brown, Allyson L. Lister, Jon Ison, Helen Parkinson, and Robert Stevens. "The Software Ontology (SWO): A Resource for Reproducibility in Biomedical Data Analysis, Curation, and Digital Preservation." *Journal of Biomedical Semantics* 25.5 (2014): n. pag. *BioMed Central*. 2 June 2014. Web. 20 May 2016. <http://jbiomedsem.biomedcentral.com/articles/10.1186/2041-1480-5-25>
- Marchese, Francis T. "Software Archaeology and the Preservation of Code-based Digital Art." *Archiving 2013 Conference Final Program and Proceedings*. Society for Imaging Science and Technology, 2013. 25-30. Print.
- Matthews, Brian; Shaon, Arif; Bicarregui, Juan; Jones, Catherine; Woodcock, Jim; & Conway, Esther. (2009). Towards a Methodology for Software Preservation. *California Digital Library*. UC Office of the President: California Digital Library. Retrieved from:  
<https://escholarship.org/uc/item/8089m1v1>
- McDonough, Jerome P., Robert Olendorf, Matthew Kirschenbaum, Kari Kraus, Doug Reside, Rachel Donahue, Andrew Phelps, Christopher Egert, Henry Lowood, and Susan Rojo. *Preserving Virtual Worlds Final Report*. University of Illinois, 20 Sept. 2010. Web. 10 Sept. 2015.  
<https://www.ideals.illinois.edu/handle/2142/17097>
- Owens, Trevor, and Doug White. "Life-Saving: The National Software Reference Library." *The Signal: Digital Preservation*. Library of Congress, 4 May 2012. Web. 20 Sept. 2015.

<http://blogs.loc.gov/digitalpreservation/2012/05/life-saving-the-national-software-reference-library/>

Owens, Trevor, Henry Lowood, Matthew Kirschenbaum, Alice Allen, Doug White, and Michael Mansfield. *Final Report for Preserving.exe: Toward a National Strategy for Preserving Software*. Proc. of Preserving.exe: Toward a National Strategy for Preserving Software, Library of Congress, Washington, DC. Library of Congress, 2013. Web. 2015.

<http://www.digitalpreservation.gov/meetings/preservingsoftware2013.html>

Stodden, V., D. H. Bailey, J. Borwein, R. J. LeVeque, W. Rider, and W. Stein. (2013) Setting the Default to Reproducible: Reproducibility in Computational and Experimental Mathematics. *ICERM*. Proc. of ICERM Workshop on Reproducibility in Computational and Experimental Mathematics, Brown University, Providence, RI. Web. [https://icerm.brown.edu/tw12-5-rcem/icerm\\_report.pdf](https://icerm.brown.edu/tw12-5-rcem/icerm_report.pdf)

*TechFocusIII: Caring for Software-Based Art*. Solomon R. Guggenheim Museum, New York. AIC, 30 Sept. 2015. Web. 20 Oct. 2015. <http://resources.conservation-us.org/techfocus/techfocus-iii-caring-for-computer-based-art-software-tw/>

Webster, Keith, and Euan Cochrane. *Software Curation as a Digital Preservation Service*. Coalition for Networked Information, 7 Apr. 2015. Web. 20 Sept. 2015. <https://www.cni.org/topics/digital-preservation/software-curation-as-a-digital-preservation-service>

Wilson, Scott. "Preserving and Curating Software." *OSS Watch*. N.p., 5 Nov. 2014. Web. 20 Sept. 2015. <http://oss-watch.ac.uk/resources/preservation>

Yahaya, Jamaiah H., Aziz Deraman, and Zuriani Hayati Abdullah. "Evergreen Software Preservation: The Anti-Ageing Model." *Proceedings of the International Conference on Internet of Things and Cloud Computing*. New York: ACM, NY. Print.