**D-Lib Magazine**
**January 2005**

**Volume 11 Number 1**

**ISSN 1082-9873**

# Transparent Format Migration of Preserved Web Content

David S. H. Rosenthal, Thomas Lipkis, Thomas S. Robertson, and Seth Morabito
<dshr@stanford.edu>, <tal@pss.com>, <troberts@stanford.edu>, <sethm@loomcom.com>
Stanford University Libraries

## Abstract

The LOCKSS digital preservation system collects content by crawling the web and preserves it in the format supplied by the publisher. Eventually, browsers will no longer understand that format. A process called format migration converts it to a newer format that the browsers do understand. The LOCKSS program has designed and tested an initial implementation of *format migration* for Web content that is transparent to readers, building on the content negotiation capabilities of HTTP.

## Introduction

Eventually, any format in which digital content might be stored will become obsolete. A format is said to be obsolete when current hardware and software are no longer able to render information represented in it understandable to readers. The design of digital preservation systems must anticipate this obsolescence, and incorporate a strategy by which the content they preserve will still be understood by readers after multiple generations of formats have become obsolete. Two such strategies have been identified:

- **Emulation** - in which the content is both preserved and presented to readers in the original format [1].
- **Migration** - in which the content is presented in a current

> format; it may be preserved in a succession of current formats or in the original format that is transformed on request into the current format for presentation [2].

Some software business models depend on a rapid upgrade cycle. In these areas rapid format change is normal; users who do upgrade produce a format users who have yet to upgrade cannot interpret. This is a powerful motivation for further upgrades, and thus a powerful income generator. But note that rapid format *change* doesn't imply rapid format *obsolescence*. An upgraded application that didn't accept old formats would not be an effective income generator.

We provide an overview of the problem of format obsolescence as applied to Web content and, in this context, examine possible implementations of the two strategies. We identify the practical difficulties that face any implementation of emulation; they led us to choose the migration strategy. We describe the design and implementation of a transparent, on-access format migration capability for the LOCKSS system for preserving Web content. [LOCKSS is a trademark of Stanford University. It stands for Lots Of Copies Keep Stuff Safe.]

Our implementation is capable of transparently presenting content collected in one Web format to readers in another Web format, with no changes needed to browsers. The reader need take no special action to cause this to happen, nor even be aware that it is happening. This appears to be the first time that a production digital preservation system has demonstrated using live Web content that format migration can be performed transparently on behalf of end-users.

## Format Obsolescence of Web Content

A Web format may be said to be obsolescent when widely used browsers are no longer able to present content in that format to their readers.

To the casual observer it may appear that the format in which Web content is supplied is solely determined by the Web server, possibly by the file name extension. In fact, the format is determined by one of a set of mechanisms for *content negotiation* defined in Section 12 of *Hypertext Transfer Protocol—HTTP/1.1* [3] that are capable of negotiating format, language, and encoding. The mechanism for format negotiation uses the **Accept:** header defined in Section 14.1 of *HTTP/1.1* [3]. A browser may try to influence the server's choice of format by sending it this header with a list of acceptable **Mime-Type** values, each with a numeric preference value between 0 and 1. If the server is capable of supplying the requested content in multiple

formats, its choice among them may take into account this list of **Mime-Type** values and corresponding preferences. Whether or not they send an **Accept:** header and whatever use the server made of it, browsers determine the format of the content they receive using the **Mime-Type:** header that accompanies it.

A browser issuing a request for a URL can't predict what Mime-Type: it will get back, or even whether it will be text, audio, video or some other class of object. Browsers therefore typically send a default **Accept:** header on most **GET** requests specifying their preferred **Mime-Type** values, covering all classes of object. These lists typically include a low-preference default ***/*;q=0.1** saying, in effect, "if you can't give me what I want, give me what you have". Because browsers indicate in this way their willingness to receive *any* format, there is some difficulty in determining when obsolescence has occurred.

In brief, the problem of format obsolescence for a system preserving Web content is that of what to do when the system receives a request for some content that was collected in format **F/G**, say **image/gif**, whose **Accept:** header indicates **F/G** is not an acceptable format.

In the light of the ***/*;q=0.1** usage, there are two ways in which this can happen: the browser can explicitly signal it, or the server can be configured to assume it. Although Section 12 of *HTTP/1.1* [3] does not define the semantics of a preference value of zero, it appears that servers treat this as an instruction not to send content in that **Mime-Type**. Thus even a browser that uses ***/*;q=0.1** can flag a format as unacceptable by **F/G;q=0**. Alternatively, a server could be configured not to recognize **F/G** as matching */*.

Fortunately, since Web browsers and their plug-ins are normally free, there are few incentives for rapid format change. No one clamors to remove support for an old Web format; it is valuable so long as there is Web server content that has not yet been migrated to a more current format. There are no good ways to motivate small Web sites to perform this migration, so old Web formats die a very slow death. From the viewpoint of digital preservation, this makes Web content easier to handle; there will be plenty of time to implement format migration.

### Emulation of Obsolete Web Formats

The goal of the *emulation* preservation strategy is to avoid the loss of fidelity that is likely to result from converting content from one format to another. If the content is preserved in its original format and presented to the reader in that format, no conversion is needed. What is needed is the ability of a future reader to run the software the original reader would have run to experience the content. The

emulation strategy seeks to answer that need by preserving the original software as well as the content and providing the future reader with a software emulation of the environment needed to run the original software to interpret the preserved content in its original format. In a suitable context, the emulation strategy is attractive; it is being pursued, for example, by IBM [4] and the Koninklijke Bibliotheek (KB, Dutch National Library) [5] who have collaborated to build a PDF interpreter that runs on a Universal Virtual Computer (UVC), a virtual machine designed to be easy to port to future environments. The terms under which the KB preserves content make this appropriate; they mandate that content be accessed only at the KB, where deployment of the UVC is easy.

In the Web context, emulation means that a future reader wishing to read a preserved Web page that contains some content in an obsolete format must somehow find out the approximate date of the original content and then locate a preserved browser or plug-in of that date and the appropriate emulation needed to allow that preserved browser or plug-in to run in the reader's current computing environment. The reader must then invoke this emulation to run the preserved browser or plug-in to view the Web page.

Since in the emulation strategy all these activities take place in the reader's environment, there is little the preservation system can do to enable them. It has no control over the reader's environment. Indeed, if it is disseminating the preserved content by acting as a Web server, the preservation system will have almost no knowledge of the reader's environment. The effect of a successful emulation strategy would be to prevent the preservation system ever seeing a request with **F/G;q=0** in its **Accept:** header; however, the practical difficulties of implementing the emulation of instruction sets, operating systems, etc. and deploying both the appropriate emulation and preserved browser or plug-in to the appropriate reader are formidable.

### *Migration of Obsolete Web Formats*

*Migration* of Web content from an obsolete format to a current one can take place at any time between the point at which the content is collected and the point at which the reader requests access to it. We examine three migration points that have been implemented, from the earliest to the latest.

### *Migration on Ingest*

The National Archives of Australia (NAA), faced with a requirement to preserve vast volumes of government information in a wide variety of

mostly proprietary formats, chose a strategy of *migration on ingest* [6]. NAA preemptively migrates the content it receives into one of a small number of carefully chosen formats before preserving it. If those choices turn out well, NAA's pragmatic approach has significant advantages:

- It can postpone the need for future migration for a long time, allowing both economies in operation and the use of better, future technology for performing the next migration.
- It can greatly reduce the cost of eventual future migrations by reducing the number of formats to be migrated.

Both these advantages are greatly enhanced if the formats chosen are open standards and are supported by open source software, as they are in the case of NAA. Most of the material NAA handles is not from the Web, and most Web formats would meet their criteria without an initial migration.

The disadvantages of this approach are two-fold. First, *migration on ingest* does not fully satisfy the requirements of archivists, because the content is not preserved in its original form. Some potentially useful information may be lost in the initial migration. Second, *migration on ingest* postpones the format migration problem but does not actually solve it. Even the chosen formats will eventually become obsolete.

### *Batch Migration*

When a format in which some content is being preserved is thought likely to become obsolete, a preemptive *batch migration* process can be performed. The preserved content in the obsolete format is converted to a current format en masse. Some stand-alone tools for doing batch migration have been developed [7] but they have yet to be integrated into a complete digital preservation system.

The DAITSS (Dark Archive In The Sunshine State) system is designed to use a batch migration strategy [8]. As a dark archive, one not intended to be accessed by readers but maintained in a controlled environment, this is an appropriate solution. The archive has total control of the environment and no urgent demands from end-users to satisfy.

### *Migration on Access*

The alternative migration approach, *migration on access*, postpones format migration until the reader actually requests the preserved content. *Migration on access* avoids the disadvantages of the other migration strategies by preserving the content in its original formats.

When a format is thought likely to become obsolete, the digital preservation system is enhanced with the ability to present the reader, upon request, with the requested content in a current format. In effect, the migration tool is integrated into the dissemination pipeline of the preservation system rather than being applied to the preserved content.

This approach requires the ability to convert dynamically from the obsolete to the current format, but it offers significant advantages:

- Content is preserved in its original format, satisfying the archivists' requirements and avoiding the risk of information loss from buggy format converters. Note that NAA actually does preserve both the original and the migrated format but expects that access to the original will be an exception. DAITSS also preserves both the original and the migrated format. The risk of information loss is clearly enough to motivate systems using other migration strategies to hedge their bets by preserving the original format too.
- Preserved content is migrated by the most recent, and presumably best, technology available at the time the reader requests access.
- Preserved content is rarely accessed, thus delaying format migration until it is actually required, and reduces the resource cost of the process by the proportion of content that is never accessed and by the decreasing cost of technology through time.
- Content can be migrated directly from the original to the current format, minimizing the effects of format conversion artifacts.
- The format converters, once developed, can themselves be preserved to document the original format. Note that a converter can be developed *before* the format becomes obsolete and can be preserved against future need if the format's longevity is suspect.
- As with other migration strategies, careful choice of the format to migrate to can greatly reduce the need for and cost of future migrations.

The disadvantages of the *migration on access* strategy are that dynamic format migration may impose significant delays on readers' accesses to preserved material, and that it requires close integration with the dissemination pipeline delivering the digital preservation system's preserved content to its readers.

## Format Migration in the LOCKSS System

The LOCKSS system provides librarians with a simple, low-cost tool they can use to ensure that their communities will have continuing access to material published on the Web [9]. The LOCKSS system is

designed to handle both for-fee subscription e-journals and open access material where copyright is held by the publisher, not by the library's institution. The LOCKSS system is a peer-to-peer network of low-cost PCs running free, open-source software at libraries. Each peer:

- **Collects** the material to be preserved by crawling the publisher's web site (after verifying that the publisher has granted suitable permission).
- **Preserves** the material by cooperating with other peers holding the same material in a mutual audit process by running polls to identify any missing or damaged content and repair it.
- **Disseminates** the preserved material by acting as a proxy cache, intercepting requests from the library's browsers for the original URL from which the material was collected. If the publisher's copy is still available, that copy is delivered. Otherwise the preserved copy is delivered.

The LOCKSS system was released for production use in April 2004 and currently about 80 libraries worldwide use it. Publishers of over 2000 titles have endorsed the system.

**Design**

In the LOCKSS system it is natural to use the migration on access strategy. Preserving content in the original format greatly simplifies the mutual audit and repair process, and the LOCKSS system already implements the complete dissemination pipeline into which the migration process must be integrated.

The LOCKSS system is being enhanced to provide:

- An API for plug-in format converters, by which they can register their input and output **Mime-Type** values, and by which the LOCKSS web proxy code can invoke them to perform on-the-fly conversion.
- A matching process that takes the **Accept:** header of incoming requests and compares it to the original format of the preserved content. If the original format is not acceptable, the matching process searches the table of registered format converters, looking for one that takes the original format as input and for which the output format is acceptable. If a suitable converter is not found, a 406 error is returned as required by Section 10.4.7 of HTTP/1.1 [3].
- A distributed registry of converters, similar to the distributed registry of the plug-ins that adapt the LOCKSS system to particular content. These registries treat Java classes exactly as

other Web content: collecting them by crawling the Web and preserving them by mutual audit and repair.

### Proof-of-Concept Implementation

To confirm the feasibility of this design, a proof-of-concept was implemented and tested. We chose an "obsolete" format widely used in actual content collected by the production LOCKSS system, and a suitable "current" format to replace it. The obsolete format was GIF [10], a format for images that has been in use for a long time. Many open source advocates have deprecated the GIF format for reasons connected with intellectual property restrictions, and they have developed the PNG [11] format as a replacement. This background makes our assessment less artificial, as the format migration in question has been actively solicited. Tools for converting from GIF to PNG are widely available, as would be expected if a widely used Web format were to become obsolete. We did not implement the full **Mime-Type** matching process, but rather a configuration option in the LOCKSS proxy Web server that prevented **image/gif** from matching any **Accept:** header. The mismatch triggered a GIF-to-PNG conversion directly, delivering the content converted to PNG at the original URL but with **Mime-Type=image/png**.

### Assessment

As can be seen from the before (Figure 1) and after (Figure 2) screen-shots, the GIF-to-PNG format migration is not perceptible to the user. Nor does the GIF-to-PNG format migration incur a noticeable delay in accessing the page.

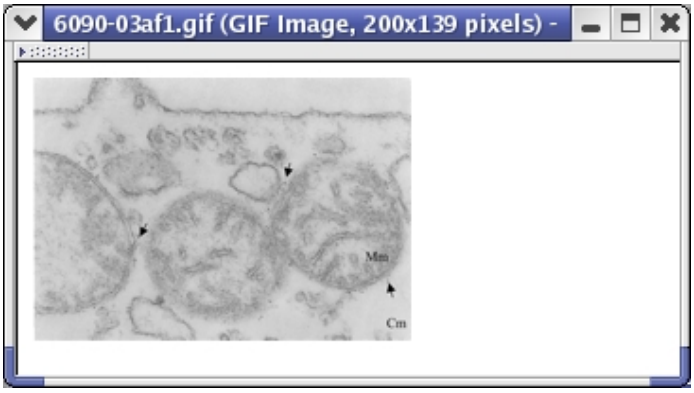


**Figure 1: Firefox browser displaying a GIF image from an article in *Journal of Histochemistry and Cytochemistry* before the simulated obsolescence of GIF. Note the window headline.**

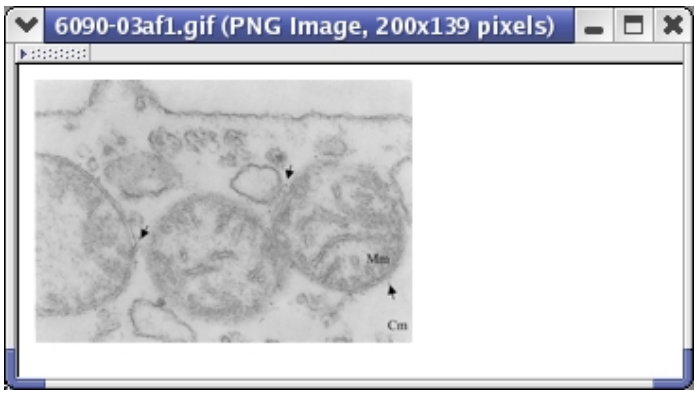


**Figure 2: Firefox displaying the same image after the simulated obsolescence of GIF. The window headline shows that the GIF file has been converted to PNG.**

## Future Work

Our next step is to replace the proof-of-concept implementation by a full implementation of the API for plug-in format converters, and a broader set of converters than just GIF-to-PNG. This implementation will need a more realistic-scale test, and we are arranging to conduct one.

Another approach would be to connect the API to a format migration service such as TOM (Typed Object Model) [2]. We are investigating this possibility.

The development of future format converters will be significantly easier if more—and more reliable—format metadata is available. We are working towards incorporating Harvard's JHOVE [12] format metadata extraction and validation technology to improve the quality of the format metadata in the LOCKSS system.

## Conclusion

We have designed, implemented a proof-of-concept and demonstrated transparent format *migration on access* for the LOCKSS digital preservation system. By doing so we have validated one of the possible format migration strategies and reassured the community of LOCKSS users that, when the time comes, the content they are preserving will remain accessible despite the obsolescence of the formats in which the content was collected.

## Acknowledgments

We are grateful to Claire Griffin and to Vicky Reich, the director of the LOCKSS program, for their help.

This work is supported by the National Science Foundation (Grant No. 9907296) and by the Andrew W. Mellon Foundation. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of these funding agencies.

## References

1. Rothenberg, J., Ensuring the Longevity of Digital Documents. *Scientific American* 272, 1 (1995).

2. Ockerbloom, J., *Mediating Among Diverse Data Formats*. Tech. Rep. CMU-CS-98-102, Carnegie-Mellon University, 1998. <http://tom.library.upenn.edu/pubs/thesis/>.

3. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. *Hypertext Transfer Protocol- HTTP/1.1*. <http://www.w3.org/Protocols/rfc2616/rfc2616.html>, 1999.

4. Lorie, R., Preserving Digital Documents for the Long-Term. In *IS&T Archiving Conference* (San Antonio, TX, USA, 2004), pp. 88-92. <http://www.imaging.org/store/epub.cfm?abstrid=30296>.

5. van Wijngaarden, H., and Oltmans, E. Digital Preservation and Permanent Access: The UVC for Images. In *IS&T Archiving Conference* (San Antionio, TX, USA, 2004), pp. 254-258. <http://www.imaging.org/store/epub.cfm?abstrid=30329>.

6. Heslop, H., Davis, S., and Wilson, A., *National Archives Green Paper: An Approach to the Preservation of Digital Records*. <http://www.naa.gov.au/recordkeeping/er/digital_preservation/Green_Paper.pdf>, 2002.

7. Walker, F., and Thoma, G., A Web-Based Paradigm for File Migration. In *IS&T Archiving Conference* (San Antionio, TX, USA, 2004), pp. 93-97. <http://www.imaging.org/store/epub.cfm?abstrid=30297>.

8. DAITSS Overview. <http://www.fcla.edu/digitalArchive/pdfs/DAITSS.pdf>, 2004.

9. Maniatis, P., Roussopoulos, M., Giuli, T., Rosenthal, D. S. H., Baker, M., and Muliadi, Y., Preserving Peer Replicas By Rate-Limited Sampled Voting. In *Proceedings of the Nineteenth ACM Symposium on*

*Operating Systems Principles*, (Bolton Landing, NY, USA, Oct. 2003), pp. 44-59. <http://www.eecs.harvard.edu/~mema/publications/SOSP2003.pdf>.

10. Graphics Interchange Format version 89a. <http://www.w3.org/Graphics/GIF/spec-gif89a.txt>, 1990.

11. Portable Network Graphics (PNG) Specification (Second Edition). <http://www.w3.org/TR/PNG/>, 2003.

12. JHOVE, Format-Specific Digital Object Validation. <http://hul.harvard.edu/jhove/>, 1999.



---

**Top | Contents**
**Search | Author Index | Title Index | Back Issues**
**Previous Article | Conference Report**
**Home | E-mail the Editor**

---



**doi:10.1045/january2005-rosenthal**