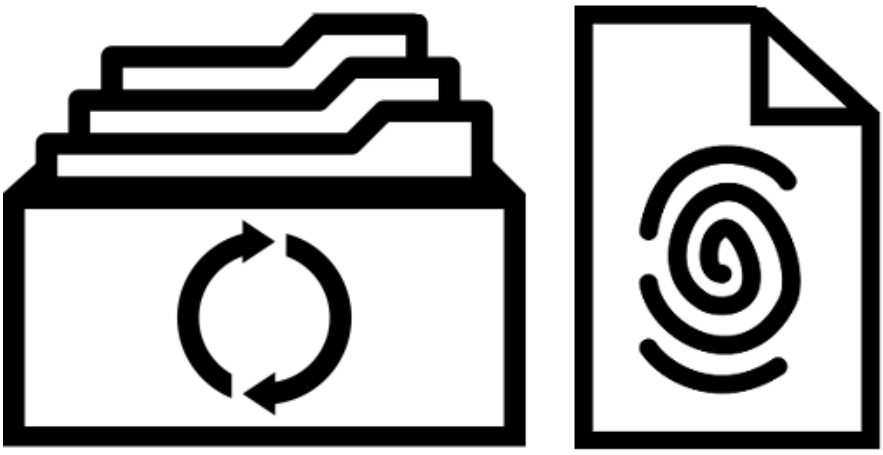
CHECKING YOUR DIGITAL CONTENT

AN NDSA PUBLICATION



Icons for Archive and Checksum, Designed by Iconathon Los Angeles, California, US 2013. Public Domain

2014

What is Fixity, and When Should I be Checking It?

Authors

- Paula De Stefano, New York University
- Carl Fleischhauer, Library of Congress
- Andrea Goethals, Harvard University
- Michael Kjörling, Independent Researcher
- Nick Krabbenhoeft, Educopia Institute
- Chris Lacinak, AV Preserve
- Jane Mandelbaum, Library of Congress
- Kevin McCarthy, National Archives and Records Administration
- Kate Murray, Library of Congress
- Vivek Navale, National Archives and Records Administration
- Dave Rice, City University of New York
- Robin Ruggaber, University of Virginia
- Trevor Owens, Library of Congress
- Kate Zwaard, Library of Congress

A joint project of the National Digital Stewardship Alliance Standards & Practices and Infrastructure working groups.

Persistent URL: http://hdl.loc.gov/loc.gdc/lcpub.2013655117.1

How do I verify that a file/object has not changed over time or during transfer processes? This fundamental goal of digital preservation is attained by establishing and checking the "fixity" or stability of digital content. At this point, many in the preservation community know they should be checking the fixity of their content, but how, when and how often? The need for guidance on fixity and audit and repair strategies was specifically noted in The National Agenda for Digital Stewardship (http://www.digitalpreservation.gov/ndsa/nationalagenda/). This document aims to help stewards answer these questions in a way that makes sense for their organization based on their needs and resources.

About The National Digital Stewardship Alliance

Founded in 2010, the National Digital Stewardship Alliance (NDSA) is a consortium of institutions that are committed to the long-term preservation of digital information. NDSA's mission is to establish, maintain, and advance the capacity to preserve our nation's digital resources for the benefit of present and future generations. The NDSA comprises over 160 participating institutional members. These members come from 45 states and include universities, consortia, professional societies, commercial businesses, professional associations, and government agencies at the federal, state, and local level. NDSA organizations have proven themselves committed to long-term preservation of digital information. To learn more about the NDSA: http://www.ndsa.org.

Defining Fixity and Fixity Information

Fixity in this context is the property of a digital file or object being fixed or unchanged. In this sense, it is synonymous with bit-level integrity. Fixity information offers evidence that one set of bits is identical to another. The PREMIS data dictionary defines fixity information as "information used to verify whether an object has been altered in an undocumented or unauthorized way." The most widely used tools for creating fixity information are checksums (like CRCs) and cryptographic hashes (like MD5 and various SHA algorithms), but there are other methods such as expected file size and file count that provide basic fixity information. The instruments for creating fixity information are explained later in this document, but before getting to those, it is worth pausing to discuss some of the limitations of fixity information and explain the diverse range of reasons to collect, check, maintain, and verify fixity information.

Limits of Fixity and the Uses of Fixity Information

Generating and logging fixity information to support the repair of digital objects is necessary but not sufficient for ensuring long-term access to digital information. The information must be put to use, in the form of scheduled audits of the objects against the fixity information. Additionally, replacement or repair processes must be in place. Ideally these will have been tested before being needed. All of this is critical for bit-level preservation, but ensuring fixity does not mean that the object is or will be understandable. Long term access is also contingent on one's ability to make sense of and use the contents of the file in the future.

Reasons to Collect, Check, Maintain, and Verify Fixity Information

Fixity information helps answer three primary questions:

- Have you received the files you expected? When fixity information is provided with objects upfront, it can be used to validate that you have received what was intended for the collection.

- Is the data corrupted or altered from what you expected? Once you have generated baseline fixity information for files or objects, comparing that information with future fixity check information will tell you if a file has changed or been corrupted.
- Can you prove you have the data/files you expected and they are not corrupt or altered? By providing fixity information alongside content, you enable your users to verify that what they have is identical to what you say it should be. This supports assertions about the authenticity and trustworthiness of digital objects.

Fixity information has other uses and benefits as well.

- **Support the repair of corrupt or otherwise altered digital files or objects:** If you have multiple copies of digital objects, you can discern which copy is correct and then replace the corrupted digital files or objects.
- **Monitor hardware degradation:** If checks against fixity information for a set of objects begin fail at high rates, it can be an indication of media failure. This should be considered a supplement to hardware-specific monitoring of storage components.
- **Assure confidence when providing someone with a copy of an item, or a segment or portion of an item, that the file or object is unchanged:** By comparing fixity information on a file or object against recorded fixity information you can be sure that what you are providing is exactly what you assert it should be.
- **Permit an update to a portion of a content file or object while being able to determine that the "other" portion is unchanged:** If you maintain very granular fixity information you can use comparisons and revisions of that information to be sure that other parts of it are not affected.
- **Meet requirements or best practice guidelines** such as ISO 16363/TRAC and the NDSA Levels of Digital Preservation.
- **Support the monitoring of production or digitization processes:** Generating and checking fixity information across complex processes provides a means to monitor content integrity as content is moved and custody is transferred.
- **Document provenance and history:** By maintaining and logging fixity information, you can provide evidence of the integrity of content across the time objects have been under stewardship.
- **Help diagnose possible systemic or human error in the lifecycle management of preserved content:** Regularly computing fixity information and comparing it with initial fixity information provides continual documentation of changes or damage to files. As such, the process of checking fixity works to help surface issues related to operator error or system problems.

General Approaches to Fixity Check Frequency

The following list describes a wide range of approaches to checking the fixity of digital content, including those that are built into storage systems, those that can be automated through scripts and applications, and those that might involve manual workflows.

- **Generating/Checking Fixity Information on Ingest:** It is important to check the fixity of content transferred to you when you bring it under your stewardship. Whenever possible, it's ideal to encourage content providers or producers to submit fixity information along with content objects. You can only provide assurance about the fixity of content overtime once you have initial fixity values, thus it is imperative to document fixity information as soon as possible. If fixity information isn't provided as part of the transfer, you should create fixity information once you have received the materials.

- **Checking Fixity Information on Transfer:** Transferring data from one storage system to another is a potential point at which your digital content could be damaged. As such, it is critical to check the fixity of your content whenever it is moved. Assuming you have additional copies of your data, you can use any failures as an opportunity to recover or repair by checking any files or objects that show up as not matching their fixity values against the fixity values for other copies. Note, some file copy applications, like Robo-copy, perform check fixity or can be configured to do so automatically as part of transfer workflows.

- **Checking Fixity at Regular Intervals:** In addition to checking fixity before and after transfer, collections of digital files and objects should be checked on a regular basis. There are a range of systems and approaches focused on checking the fixity values of all objects at regular intervals. This could be monthly, quarterly, or yearly for example. The more often you check, the more likely you are to detect and repair errors.

- **Building Fixity Checking into Storage Systems:** Some storage systems have fixity built into the system so that data is regularly checked. Some systems support per-file checksums on tape and, with new tape technologies, per-block checksums can be validated on tape systems without reading the data back to the host. Some file systems (such as ZFS) compute block level fixity information on a regular basis. However, file system checksumming will not log or help to fix issues that occur through the file system; s. As a result, even with such file systems, separate fixity data will still be needed to fully ensure changes to content (including individual file deletion) can be detected. File system level checksums can, however, be used as one indicator to detect (and, with redundancy available, automatically recover from) storage media degradation.

- **Fixity Checking for Process Monitoring:** For certain classes of content -- there are notable examples in the realm of digital audio and video -- fixity information supports process or production monitoring. For example, checksums on individual data blocks can enable stewards to know how extensive any digital damage is and where it is located. This information can inform further attempts to get a proper read from a medium under adjusted circumstances in hopes of producing a more accurate data transfer.

- **Fixity Checking on a Segment or Portion of a File** when that segment is to be provided to an end user or when other portions of the file are to be changed. Examples of this implementation include checksumming the encoded audio data within an audio file or individual frame-level checksums for video files.

Considerations for Fixity Check Frequency

You might now ask yourself, why doesn't everybody just run fixity checks and compare fixity information for all their content at fixed intervals? For answers to that question, we have to examine resource constraints and, in some cases, lack of fixity support.

- **Storage Media:** Doing fixity checks typically increases the usage of the media and of the mechanical devices that read and handle the media. For some media, usage may be a factor contributing to the projected failure rate of the media.
- **Throughput:** Your rate of fixity checking is going to be dependent on how quickly you can run the checks, the complexity of your chosen fixity instrument, and how much of your resources (e.g., CPU, memory, bandwidth) can be used for this operation. This can become a choke point as the amount of digital content increases but the infrastructure to perform the checks stays the same. In a situation like this, the fixity checking activities can adversely affect other important functions like delivery of the content to users.
- **Number and Size of Files or Objects:** Different resource requirements emerge as the scale of digital files and objects increases both in number and size.
- **Redundancy Level in Content and Process:** Depending on your system design, you may want to have different practices for checking fixity on redundant copies. For example, if the fixity is already being checked for the primary and secondary copies on a regular basis, you might decide that you don't need to check the tertiary copies as often. Similarly, it often makes sense to have different practices for files that serve as preservation masters than for files that serve as derivatives or access copies.
- **Assurance from Third Party, like a Cloud Storage Provider:** Instead of running your own checks, you may be in a situation to trust the claims of a third party about their persistent checking of the copies they maintain in their system. It is important to understand the details of what a cloud provider is supporting and how often and how detailed fixity checks are done.
- **Covered at the File System or Object System Level:** If the file or object system itself performs frequent checks at the block level you may not need to be as concerned with bit rot as a threat to fixity. For example, file systems like ZFS maintain and check block level fixity information which can be automatically used to support the repair of data that is damaged through silent data corruption (assuming there is only one hardware corruption event at a time). Importantly, maintaining and checking fixity information separate from this automated process is invaluable in offering further confidence and assurance of fixity.

Characteristics of Common Fixity Instruments

The table below presents some basic information on a range of instruments that are commonly considered for creating fixity information. Each instrument takes different levels of effort and resources to use and results in varying degrees of detail and quality of the fixity information they generate.

Fixity Instrument	Definition	Level of Effort and Return on Investment
Expected File Size	File size that differs from the expected can be an indicator of problems, for example [by highlighting] zero byte files	**Low level of effort and low level detail.** File size is auto-generated technical metadata that can be viewed in Windows Explorer or other common tools.
Expected File Count	File count that differs from the expected can be an indicator that files are either added or dropped from the package.	**Low level of effort and low level detail.** File count is auto-generated technical metadata that can be viewed in Windows Explorer or other common tools.
CRC	Error detection code, typically used during network transfers.	**Low level of effort and moderate level of detail.** CRC function values, which are variable but typically 32 or 64 bit, are relatively easy to implement and analyze.
MD5	Cryptographic hash function	**Moderate level of effort and high level of detail.** CPU and processing requirements to compute the hash values are low to moderate depending on the size of the file. The output size of this hash value is the lowest of the cryptographic hash values at 128 bits.
SHA1	Cryptographic hash function	**Moderate level of effort, high level of detail, and added security assurance.** Due to its higher 160-bit output hash value, SHA-1 requires more relative time to compute for a given number of processing cycles CPU and processing time than MD5.
SHA256	More secure cryptographic hash function	**High level of effort, very high level of detail, and added security assurance.** With an output hash value of 256 bits, SHA-256 requires more relative time to compute for a given number of processing cycles CPU and processing time than SHA-1.

Verifying auto-generated metadata such as expected file counts and file size can provide a simple accessible first line of inquiry into the health of the digital objects, both as individual units and an aggregate bundle, and can be used in all situations in combination with other instruments.

Network protocols such as ethernet, SAS, fiber channel, and others have low-level CRCs built into the hardware protocol. The ANSI T10 PI standard is used in addition to the standard channel CRCs for an added level of protection and detection in hardware interfaces. CRCs are also used often on the intra-file level.

For a given fixity instrument, the harder it is to find two objects that result in the same fixity information, the more "collision resistant" that instrument is. This is important mostly for preventing the concealment of intentional changes to objects. For example, expected file size and expected file count are extremely vulnerable to collision: it is very easy for someone to replace an object with one that matches in file size. It's also possible (although unlikely) for an unintentional change (such as corruption or human error) to result in an object with the same fixity information for instruments that have low collision resistance. Of the fixity instruments described above, the cryptographic hash functions (MD-5, SHA-1, and SHA-256) are the most collision resistant; SHA-256 is recommended for applications where security is important. However, performing fixity checking and replacing damaged objects is critical for any preservation system, and using any fixity instrument is much better than none at all. Note that as the level of security of the hash function increases, so do the time and resources needed to compute.

Where to Store and Reference Fixity Information

Where to store fixity information depends on the situation. Here are some examples of places you might store fixity data to accomplish different objectives.

- **In object metadata records:** In many cases, you will want to record some file or object fixity information wherever you store and manage the metadata records. These metadata records are actually stored as discrete files or in databases. This is particularly useful for maintaining originally submitted or generated fixity information as part of the long-term object metadata.
- **In databases and logs:** For checks you run at given intervals you may not want to be constantly adding to your object metadata records. In this case, it makes sense to keep running fixity information in databases and logs that you can return to when needed.
- **Alongside content:** It's often ideal to have fixity information right alongside the content itself. That way, if you have problems with other layers in your system, or want to transfer some set of objects, you still have a record of previous fixity values alongside your content. For example, the BagIt specification includes a requirement for a hash value for the bagged content alongside the content. Similarly, some workflows involve creating *.md5 files, which are simply text files with the md5 hash, named identically to the file it refers to, but with an additional .md5 extension.
- **In the files themselves:** When a checksum is for a portion of a file, it may make sense to store the information directly in the file. Note that this only makes sense when storing sub-file fixity information within a file. Adding fixity information for an entire file to the file itself changes the file and therefore changes its fixity value.

Further Reading

- Hashing Out Digital Trust. Kate Zwaard. http://blogs.loc.gov/digitalpreservation/2011/11/hashing-out-digital-trust/
- Fixity Checks: Checksums, Message Digests and Digital Signatures. Audrey Novak. http://www.library.yale.edu/iac/DPC/AN_DPC_FixityChecksFinal11.pdf
- The NDSA Levels of Digital Preservation: An Explanation and Uses. Megan Phillips, Jefferson Bailey, Andrea Goethals, and Trevor Owens. http://www.digitalpreservation.gov/ndsa/working_groups/documents/NDSA_Levels_Archiving_2013.pdf
- Authenticity of Electronic Federal Government Publications. Kate Zwaard and Lisa LaPlant. http://www.gpo.gov/pdfs/authentication/authenticationwhitepaper2011.pdf
- Reconsidering the Checksum for Audiovisual Preservation. Dave Rice. http://dericed.com/papers/reconsidering-the-checksum-for-audiovisual- preservation/
- What's the Real Impact of SHA-256?. A Comparison of Checksum Algorithms. Alex Duryee. http://www.avpreserve.com/papers-and-presentations/whats-the-real-impact-of-sha-256/

Information About Relevant Standards and Specifications

- PREMIS Data Dictionary: http://www.loc.gov/standards/premis/v2/premis-dd- 2-2.pdf
- PREMIS Conformance: http://www.loc.gov/standards/premis/premisConformance_v4.pdf
- BagIt Specification: http://www.digitalpreservation.gov/documents/bagitspec.pdf
- MD5: http://en.wikipedia.org/wiki/MD5; https://tools.ietf.org/html/rfc1321
- SHA-1: http://en.wikipedia.org/wiki/SHA-1; http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf
- SHA-256: http://en.wikipedia.org/wiki/SHA-2; http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf
- Cyclic redundancy check (CRC): http://en.wikipedia.org/wiki/Cyclic_redundancy_check
- Standards for SCSI interfaces: http://www.t10.org/
- SMART monitoring for storage drives: http://en.wikipedia.org/wiki/S.M.A.R.T.