BACK TO BASICS: THE MINIMUM PRESERVATION TOOL

Maureen Pennock
The British Library
United Kingdom
maureen.pennock@bl.uk
https://orcid.org/0000-0002-7521-8536

John Beaman
The British Library
United Kingdom
john.beaman@bl.uk
https://orcid.org/0000-0002-0357-1154

Peter May
The British Library
United Kingdom
peter.may@bl.uk
https://orcid.org/0000-0001-8625-9176

Kevin Davies
The British Library
United Kingdom
kevin.davies@bl.uk
https://orcid.org/0000-0001-6522-9568

Abstract – This paper presents the Minimum Preservation Tool (MPT), designed and developed by the British Library to provide a basic and local technical digital preservation environment for collections awaiting ingest to a more formal digital preservation repository. The MPT can satisfy fundamental preservation storage requirements that are not typically otherwise supported in a standard corporate technical environment more focused on cyber-security. Replication, checksum generation and validation, and regular reporting are all key features of the MPT, written as a set of Python Utilities and freely available on Github. MPT is an entry-level tool that lowers the bar for early participation in preservation endeavors, in contrast with larger scale and more expensive, complex end-to-end technical solutions.

Keywords – minimum preservation tool, checksum, integrity, replication, accessibility, open source, file preservation, risk reduction, assurance

Conference Topics – Building the Capacity & Capability

I. INTRODUCTION

Digital preservation is an ambitious discipline: we seek to maintain long-term access to authentic digital content that is inherently intangible and otherwise prone to damage or loss from a multitude of different sources. Standards such as ISO 16363 clearly infer that 'trustworthy' and reliable digital preservation requires a fully functioning digital repository system. [1] Yet repositories are only one part of a larger picture within the discipline and across the digital preservation community. Our experience over the past two decades is there is often a delay between initial creation of content and ingest of content into a preservation repository system. Content files may be damaged or lost if not maintained properly during this time, with varying consequences to the content's demonstrable provenance and integrity (which may or may not be noticed prior to ingest into a costly preservation system). Moreover, as Langley noted at the iPres conference in 2017, whilst many national libraries and archives have built up their capacity and proficiency for managing and preserving digital collections, smaller organisations, particularly those situated outside of memory institution contexts or those in developing countries, are often still 'struggling with the basics of managing their digital materials'. [2] Fully functioning digital preservation repositories in those contexts may still be many years off.

This paper introduces the Minimum Preservation Tool as a response to these challenges, developed at the British Library to go back to basics with a bit-level solution for locally safeguarding and ensuring the

17th International Conference on Digital Preservation
iPRES 2021, Beijing, China.

DOI: 10.1145/nnnnnnn.nnnnnnn

technical integrity of content files prior to more substantial preservation activities.[1]

II. A (BRIEF) LITERATURE REVIEW

What is minimum preservation, and what does it comprise? A search through the literature does not reveal widespread use of the term, though the underlying concept (i.e. that there is a basic yet adequate way to support preservation) is evident in some notable works. The Jisc co-funded LIFE project of the 2000's provides one such example. [3] The project identified two different preservation stages of the lifecycle: bit-stream preservation, including such activities as backup, storage, and fixity audits, and content preservation, which focused on more complex processes such as preservation planning, preservation action, and preservation watch. The former, with a focus on bit-level file maintenance, is arguably a minimal level of preservation when compared to the latter's objective to ensure access to the intellectual object over time.[2] Another example is the concept of Parsimonious Preservation. [4] This aims to ensure economy of action and intervention by avoiding expenditure of effort on threats considered unlikely to manifest within the current generation of IT systems. Parsimonious Preservation relies on 'the measures already taken by a good IT services department' to manage and support bit-level storage, and argues that preservation activities should focus primarily on 'knowing your collection'; this requires such things as an inventory of content, metadata on file formats and file modification dates, and fixity information.

Tiers of maturity models offer another glimpse into what might be considered 'minimum preservation'. The lowest tier of the NDSA 'Levels of Digital Preservation', for example, requires at least two copies of files in at least two locations, alongside integrity information, control processes to limit access and alternations, an inventory of the content with some metadata, and documentation of file formats and essential content characteristics. [5] Similarly to Parsimonious Preservation, this lower tier is also defined as 'knowing' your content; protecting and monitoring it requires additional activities. The Minimum Viable Preservation concept on the other hand, discussed in 'apres-ipres 2018' and subsequently explored on the Digital Preservation Coalition blog by Matthew Addis, takes a different approach and includes not just knowing your content but also having access to an independent means of rendering the content. [6] There is no precise consensus across these sources on what exactly minimum preservation entails, but multiple copies and fixity data are common characteristics in all of these examples. Evidence on the fundamental nature of these for preservation (as well as optimized strategies for implementation) can be found in the excellent paper by Micah Altman and Richard Landau (Massachusetts Institute of Technology) presented at IDCC 2019, ''Selecting Efficient and Reliable Preservation Strategies'. [7]

The value of 'minimum' is also evident in the concept of the 'Minimal Effort Ingest' project promoted by and implemented at the Royal Danish Library. [8] This pushes certain widely-accepted yet time-consuming ingest activities such as format validation into a post-ingest stage of repository data management; as a result, incoming content files can be ingested more quickly and a base level of preservation more swiftly achieved for a greater number of items.[3]

What then does the solutions landscape look like for minimum preservation? The preservation goal for many organizations is a comprehensive, 'content-level' (to paraphrase the LIFE project term) digital preservation repository system, licensed or otherwise supported by a commercial vendor. These certainly offer more than minimum preservation, though arguably most can be configured to do as little as a customer wishes. Nonetheless, the overhead associated with commercial repository licensing, implementation and management, can still be prohibitively high. Whilst repository solutions both commercial and open source certainly support a minimum level of preservation, they are typically designed to support additional use cases beyond replication and fixity management, such as storage of object metadata or end user access. This is the case across the board, including with the highly regarded LOCKSS (Lots of Copies Keeps Stuff Safe) framework. [9]

Outside of such a commercial repository setting, a plethora of different tools exists to support different preservation functions. The Community Owned digital Preservation Tool Registry COPTR, for example, identifies over 500 different tools that support a range of preservation functions from format identification,

[1] An earlier brief introduction to the MPT can also be found at https://www.dpconline.org/blog/minimum-preservation-tool-mpt

[2] It should be noted that bit-level preservation in the LIFE project model was nonetheless considered as a post-ingest, repository level function, not as a lifecycle stage independent of a repository system.

[3] This submission won the Best Poster Award of the conference, indicating a good level of acceptance for the concept.

validation, migration, and disc imaging, to costing, de-duplication, file re-naming and version control. [10] Within the context of minimum preservation functions, Bagit and AVP Fixity, for example, are two widely used tools that support checksum generation and validation. [11] [12] Content replication for multiple identical copies is supported by other tools and utilities, such as rsync and Robocopy. [13] [14] Yet we found little within the tools landscape that simply, clearly and primarily targeted the core functions of replication and fixity checking within a single tool. It seems almost to be a case of 'repository or bust'.

III. Requirements for a Minimum Preservation Solution

Our literature and tool review indicated that none of the existing tools we surveyed satisfactorily matched with all of our requirements or in the way in which we sought to deploy and use a minimum preservation solution. MPT was therefore developed to address this gap and provide an internal, bit-level, minimum preservation environment for collections awaiting processing or content preservation in an enhanced preservation repository system.

Requirements for the MPT were purposefully few and minimal, in line with the overall goal of the initiative:

- The solution must be economical;
- The solution must be realistically achievable and maintainable with limited effort and budget;
- The solution must support at least three synchronized copies of each item, stored in at least two different physical locations;
- The solution must carry out checksum generation and validation with reports on a regular basis, ideally on all copies, but at the very least on two active (readily accessible) copies;
- There must be a fully-tested, robust and reliable recovery process in place which can restore known good copies of corrupted files from one of the data store copies, and can be invoked when file fixity issues are detected via the checksum validation process;
- The solution architecture should focus primarily on preservation of small to medium sized collections (i.e. <20TB each).[4]

These requirements are open to a certain amount of interpretation. For example, avoidance of classic terms like 'preservation masters' and 'backups' and using instead the term 'synchronized copies' allowed for different potential designs in the storage architecture. Requiring 'regular' checksumming, rather than (for example) rolling or quarterly fixity checking, provisioned a similar flexibility that could be tailored as the design took shape and our experience developed. The tool that was ultimately developed reflects this.

IV. Design & Development

The Library's digital preservation team structure includes flexible R&D resource so that it has some capacity to respond to issues and challenges as they arise; this was used to provide the staff time and effort needed to develop the concept further and build the MPT as a set of Python Utilities. The resulting toolset makes use of pre-existing network storage, read/write protocols, and compute resources already available at the Library, and simply provides additional functionality to turn existing network space into a basic preservation-acceptable environment.

The tool supports four main functions:

Staging: The MPT Staging function collects files from a pre-defined temporary holding location and moves them into designated preservation storage areas, preserving their directory structure. Additionally, the process also creates a 'tree' of checksum files that matches the original directory structure and which is transferred into each designated preservation storage area, alongside the files it represents. The default algorithm is sha256, though other algorithms are also supported. Optionally, the process can also create or update a manifest file that contains checksums for all files in the storage location.[5] If staging of a file fails for any of its destinations, then staging is aborted for that file and its failure is logged. Results are summarized in an email distributed once the staging process for a given collection is complete.

Creation: The MPT Creation function can be used to create and save checksums for files that may already be held in a preservation storage location, without going through the staging process. Checksums are saved in a checksum tree that mirrors the directory structure of the original files and optionally in a manifest file. A summary of activities is generated and sent by email once the Creation process for a given collection is complete.

[4] This requirement was defined primarily in line with our expected internal use of the tool.

[5] A number of different checksum algorithms are supported and can be selected from during instalation. The default algorithm is sha256.

Validation: The MPT Validation function checks the fixity of all files in a storage location by comparing their current checksum value to one previously calculated. The stored checksum value can be read from the checksum tree or manifest file. As with other functions, results are summarized in an email distributed once the process has completed for a given collection.

Comparison: The MPT Comparison function compares one set of checksum values against one or more other sets in different locations. The set can be a checksum tree or a manifest file. Any discrepancies are highlighted and again included in a summary distributed via email at the end of the comparison process.

The recovery process for corrupted files is currently manual and has not been automated. Whilst it remains a valid requirement, we have not to date experienced sufficient corruption to justify allocation of development effort to this task.

V. The Process

Files can be added to the MPT either via a one-off batch upload from an existing network share, or via a dedicated 'staging area' that can be created in advance. Our experience is that staging areas are particularly appropriate for organizations wishing to add content to the MPT on a regular basis, as any new content subsequently copied into the staging folder by the user is transferred automatically into the MPT by a staging process that runs at regular intervals.

Once a job is initiated, the MPT scripts generate checksums for content held in the upload location or staging area and stores them either in a manifest file or in a checksum tree. If checksums are already available, these can added to the upload location or staging area prior to job initiation so they can be re-used so long as they are structured consistently with the way MPT expects. Content is subsequently replicated across the designated MPT storage nodes, after which the MPT 'Validate' and 'Compare' functions are used. These validate files against the checksum values stored in either the checksum tree or checksum manifest file, and compare the checksum tree or checksum manifest across the storage nodes to demonstrate that they are in sync (a significantly faster process than comparing the data files themselves). Re-validation of checksums can take place whenever required and a report emailed to designated recipients detailing any discrepancies.

VI. Deployment

The MPT is freely available on Github under an Apache license v2.0, with full instructions to support installation and configuration. [15] The interface is primarily command line so usage therefore requires a level of technical competency, though a colleague at the Library (Andrew Jackson of the UK Web Archive) also produced and shared an experimental graphical user interface for running MPT on World Digital Preservation Day 2020.[6] [16]

The MPT works best when deployed within a Virtual Environment and the implementation can be tailored to suit the size of a collection. For example, although the MPT was designed primarily with small collections in mind, larger collections may benefit from their own virtual machines (VMs) to facilitate processing. Any one MPT instance can also be parallelized within a VM to make use of multi-cores.

The upper threshold for the MPT has yet to be identified, as actual thresholds are dependent on a range of factors. Our deployment at the Library has to date successfully supported in excess of 6 million files and over 16TB of data, dispersed across five distinct collections. We have configured two main storage locations with a VM running on each. Each location is backed up and backups are retained for 30 days; our checksum validation process is scheduled to run once a month within this window so that content can be recovered from a backup should both of our main copies become corrupted within this short period.[7] A member of our team generates assurance reports in Power BI, using data from the MPT's emailed results summary.

The tool relies upon the administrator to define the number and location of storage areas that will be used. This allows each deployment to vary according to organizational requirements. We acknowledge that for external users, this may mean that some deployments do not satisfy our internal requirement for at least three copies of files and therefore not achieve our definition of minimum preservation. Our installation notes on Github will soon be updated to provide some guidance on this so that potential users take this into consideration.

VII. Conclusion

The MPT is a pragmatic way to safeguard content and ensure file-level integrity in the absence of a fully-

[6] The GUI is still in an experimental stage so has not yet been incorporated into the main MPT codebase.

[7] Note that backups are separate from MPT function and part of our standard IT infrastructure processes when provisioning network storage.

fledged repository system. It fills a clear gap between an all-or-nothing approach in technical digital preservation storage management solutions. We encourage re-use of the code, and welcome feedback, engagement, and questions from the wider community about the MPT's use and development going forwards. We observe also that the MPT has potential to generate sharable and directly comparable data around storage arrangements and integrity checking results that will help grow the evidence base for future best practice.

The lasting value of the MPT comes from its simplicity and re-usability, and its availability as a new technical tool for the digital preservation community toolkit. Moreover it serves as a reminder that whilst preservation is often focused on large scale solutions, basic safeguarding actions can be taken without need of a monolithic (and potentially expensive) digital preservation system.

VIII. ACKNOWLEDGEMENTS

Our thanks go to Simon Whibley and Akiko Kimura for MPT coordination and reports generation in our live environment, David Russo for technical development and deployment support, and to Andy Jackson for his support of the MPT and production of the experimental GUI.

REFERENCES

[1] International Standards Organization, 'ISO 16363:2010 Space data and information transfer systems — Audit and certification of trustworthy digital repositories', 2012 https://www.iso.org/standard/56510.html

[2] Langley, Somaya, et al, 'Operational Pragmatism in Digital Preservation: establishing context-aware minimum viable baselines', in the Proceedings of the iPres 2017 Conference, Kyoto, Japan, 2017 http://www-archive.cseas.kyoto-u.ac.jp/ipres2017.jp/wp-content/uploads/69Somaya-Langley.pdf

[3] Ayris, Paul et al, 'The LIFE2 final project report', 2008 https://www.researchgate.net/publication/32894761_The_LIFE2_final_project_report

[4] Gollins, Tim 'Parsimonious Preservation', in the 'Online Information 2009 Conference proceedings', 2009, https://www.nationalarchives.gov.uk/documents/information-management/parsimonious-preservation.pdf

[5] NDSA Levels of Digital Preservation website, first published in 2013 and revised in 2019, https://ndsa.org/publications/levels-of-digital-preservation/

[6] Addis, Matthew 'Minimum Viable Preservation', DPC Blog, November 2018, https://www.dpconline.org/blog/minimum-viable-preservation

[7] Altman, Micah & Landau, Richard, 'Selecting efficient and reliable preservation strategies: modeling long-term information integrity using large-scale hierarchical discrete event simulation', International Digital Curation Conference 2019, Dublin, https://arxiv.org/ftp/arxiv/papers/1912/1912.07908.pdf

[8] Ammitzbøll Jurik, Bolette; Blekinge, Asger Askov & Christiansen, Kåre Fiedler 'Minimal Effort Ingest', in the Proceedings of the iPres 2015 Conference, Chapel Hill, USA, https://en.statsbiblioteket.dk/about-the-library/projects-1/MinEffortIngest_iPRES2015.pdf

[9] Lots of Copies Keeps Stuff Safe (LOCKSS), https://www.lockss.org/

[10] Community Owned digital Preservation Tool Registry COPTR, https://coptr.digipres.org/index.php/Main_Page

[11] Bagit, https://metacpan.org/pod/Archive::BagIt

[12] AVP Fixity, https://www.weareavp.com/fixity/

[13] rsync https://linux.die.net/man/1/rsync

[14] Robocopy https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/robocopy

[15] Davies, Kevin; May, Peter; & Russo, David, Minimum Preservation Tool MPT on Github https://github.com/britishlibrary/mpt

[16] Jackson, Andrew, Experimental MPT User Interface on Github https://github.com/anjackson/mpt/releases/tag/v1.1.6-UI