Archival of Digital Assets.

John Burns, Archive Analytics

Summary:

We discuss the principles of archiving, best practice in both preserving the raw bits and the utility of those bits, and assert that bit-level and functional archiving should not be treated the same way in terms of technology, organizational processes or urgency. We include a set of key questions to ask when considering an archival system.

Introduction.

Archiving is a purposeful activity for preserving the assets of an organization, with the explicit intention that they be useful for some defined uses in the future. It is distinct from backup, which is primarily there to protect against mishaps and is intended to recover ‘lost’ files for a relatively short period. The intention of archives is that archive assets remain “usable” , for a potentially indefinite period, and for any of a variety of reasons, including legal, for re-use, or for some other considered reason.

An archival system needs to recognize that that there are two distinct processes going on – with very different modes of operation. On the one hand there is the ongoing *bit preservation* processes that ensure that the deposited assets can be returned, with provable guarantees as to their integrity. On the other hand there is a much broader and tolerant set of processes that are intended to guarantee the content can be used … i.e. that the *functionality* of the bag of bits will be preserved, and adapted as future generations of users call on it.

The bit-preservation layer is the core, and has simple, clearly defined capabilities, but in an operational archive there are also always going to be a complex set of archival applications that impose additional requirements on the underlying set of preserved assets – for instance there will input conditioning services that ensure that the input is well-formed and complete, there will be approval work-flows, there will be compliance applications that verify that the assets are managed according to defined practices, both generic and domain specific. It is important to understand that these services are an essential part of an *operational* archive, but that they should be distinguished from the bit preservation itself, i.e. they are modules that may not always be required, and will vary from archive to archive and from domain to domain.

In addition there are other functions that are typically bundled with an archival system, including searching, indexing, web-portals, conversion, and compute-services. The decision to include those in a system is an operational decision, and there are benefits to be derived from tight integration of the various components, provided the essential and core function of the archive itself is not compromised. *Conversely, unless it is very clear what benefits are to be achieved from such integration, it is strongly suggested that a bit-preservation layer be selected and deployed with urgency, since losing the bits is irreversible,* whereas format migration and other functional preservation systems typically have lead times of years to decades, and can be revisited as necessary.

The remainder of this document will discuss in more detail the functions of the bit-preservation layer, the function preservation (often referred to as migration) and typical applications.

Bit preservation

Bit-preservation is mostly self-explanatory … but in practice it requires immediate, unrelenting and meticulous care of the bits, without very much regard to internal structure of the object. [The understanding being that the internal structure, its preservation, migration and "curation" are specific to the type of content, its use, its value to the organization and its stage in its life-cycle. The bit preservation system expects well-formed submissions, will confirm the integrity of the submission, and then manage the content according to fairly generic policy obligations. Those obligations are handled internally by the archive to ensure that content is never lost to hardware failure, media degradation or obsolescence, operator error, software failures, malicious interference or natural disasters. Moreover, the bit preservation system should apply sound audit and tracking principles so that it can demonstrate that the objects in its custody are the same, bit for bit, as the ones originally deposited, plus any approved additions or changes, typically by the use of the signed audit trails and cryptographic check-sums.

There are architectural design principles that can facilitate such reliability; and they must be holistic design principles, accounting for technical measures, awareness of the typical behaviours of organizations, the regulatory and business environment and the typical behaviour of organizations towards archives. [i.e. they are typically *not* the highest priority, until briefly they are, and periodic neglect is to be expected]

The core design principles are all precautionary – i.e. work on the assumption that the worst scenario will come to pass, and make the archive survive in spite of that. Given that archives are all about very long-term preservation it is likely that some of the eventualities will come to pass.
Such precautionary principles include

- ♣ Require that the operation of the archive is totally transparent … it does exactly what it is asked to do.
- ♣ Assume that the software, and possibly even documentation will not be available, so that the archive must be reconstructed just from backup media,

by inspection of the content.

- ♣ Assume that entire sites will be lost.
- ♣ Assume that there will be both incompetent and malicious operators.
- ♣ Assume that the providers of your storage systems will go out of business.
- ♣ Assume that standards will change.
- ♣ Assume that any encryption keys will be lost, whether they be held by you or by an escrow service.
- ♣ Assume that submissions will be made in error and will over-write existing data.

How do these translate into design decisions? A good operating principle is to work on the idea an average person should be able to dissect the archive storage and reconstruct it – i.e. it should be evident how the archive structure "on-disk" relates to perceived structure as seen from the outside. So, for instance, assume that no encryption is best principle, and if it is required then make sure that there are physically secured, plain text versions somewhere. If the archive is organized as a set of folders then the on disk storage should reflect the same structure with the same naming conventions.

As a checklist for sound assessment use the following checklist.

1. 1. Who 'owns' the archived objects? Is authorization role based?

   The archive is an organizational function, so the idea of personal ownership is at odds with that concept. A user should have a role, and it is the role to which permissions and capabilities are assigned. The user per-se should be identified in audit trails, for obvious reasons, but the otherwise there should be no concept of user ownership or permissions.

1. 2. Can the archive issue notifications to external services?

   As note above, the archive per-se should not be responsible for complex applications. It is not a programming or execution environment, it is not possible nor prudent to support for all and any executable, and as a general principle there should only be trusted and certified code inside 'the vault'. Therefore, in order to support complex requirements, workflows and procedures the archive should be able issue notifications that can be captured and responded to by external procedures that can then modify the archive appropriately and with the standard authentication and authorization.

1. 3. Is the archive using encryption?

   If the archive software is encrypting the content, that is bad. If it is using encryption even for off-line storage, then that is really bad. There is a distinct, almost inevitable likelihood that at some stage in its life the archive will be decommissioned. And the keys *will* be lost. So there will be no chance of reconstructing the archive. There is also a distinct possibility that keys will be lost even with an operational archive.

1. 4. What is the granularity of the access controls?

An Archive is intended to be long lived, so at any time there will be a variety of assets, at various points in their life cycle, within the archive. It should therefore be possible for access to be controlled at the level of directory trees and individual objects.

1. 5. How are users authenticated?

Ideally users should be authenticated using the same mechanism as for other applications in their organization. If the archive has its own authentication system *it will degenerate* to having a common username and password shared by 'trusted' users.

1. 6. Does the Archive really delete objects? Or just hide them? Does it support versions?

Archive best practice dictates that objects should never really be deleted, though for pragmatic reasons that may ultimately be necessary. In normal operations deleting an object should never occur, it should probably be separated from current objects. Likewise, overwriting should never occur when an object is updated, so versions should be supported, and it is better if the archive per-se supports them than that every application has to devise an naming scheme.

1. 7. Does the archive support composite objects? Does the archive enforce dependencies?

A composite object is an application convenience, and there is no single view on how a object with distinct sub-parts should be presented. That being said, almost any system is better than none at all, and any system should provide explicit dependencies, wherein (a) updating of one or more sub-parts causes or requires the creation of a new version of the composite object and (b) attempt to delete or update a dependent object will be refused and will trigger an event which has to be resolved .. probably using the notification service to initiate an external work-flow.

1. 8. Does the archive store audit logs using the same protections as the archived objects? Are the logs tamper-resistant?

One of the perennial issues around archives is provenance and authenticity, with integrity often being part of the same discussion. It is advisable that the audit logs be stored using the same storage infrastructure as the archive itself. Thus provenance is subject to the same care as the objects themselves. The audit trail should include the integrity checksums on creation, and on periodic integrity checks.

1. 9. Does the archive permit user code inside the vault?

This should raise alarm bells – user code has no place inside the archive – period. It's an archive – its prime raison d'être is integrity. If it is allowing user code execution then it is probably really something other than an archive. Use of the mechanisms in note 2 are likely sufficient for archival purposes. The one argument for internal code execution is that it allows you to implement pre-triggers, i.e. you intercept the request *and do something different.* This is contrary to one of the chief design tenets of archives – which is that its operation should be transparent … and anything that causes something other than the submitted content to end up in the archive is not advisable.

1. 10. The 100 year principle … can the archive be rebuilt from just the content?

This assumes that there is no state held in the software itself, and that the archive is completely represented by the storage.   Moreover, the storage is accessible in that an intelligent user to infer the structure of the content. This precludes encryption, tricky storage allocation schemes, or any other obfuscation.

1. 11. Can the archive survive loss of entire sites? Can it operate during and recover from network partitioning?

   If a site is lost, can the archive be reconstructed? Or better, can it heal itself? If the archive loses network connectivity and effectively ends up as two or more unconnected networks, to what extent can it continue operating in each of those segments? When connection is re-established, will it restore itself to a consistent state, or be able to detect and ask for resolution of inconsistencies.

Functional Preservation.

Functional preservation tries to ensure the continued usefulness of archive contents. In that sense it is the opposite of bit-preservation, in that it has to recognize that eventually a different set of bits will be needed, because the original binary object will no longer be compatible with extant software. For example, the Lotus-123 spreadsheet from 1990 may still be around, but there is unlikely to be any software to read it.   Or, it may be that the utility of the content has changed … what was a research paper in 1960 may now be a source of `historical' terminology or observational data may now be used for calibrating modern measurements.

Again, in contrast to bit-preservation, there is not the same unrelenting need for operational attentiveness.   It will become apparent that the functionality of a class of objects is at risk long before that happens. There is lots of time to assess, test and action a preservation process, which may include file conversion (e.g. Lotus 123 to Excel), data extraction (e.g. to a text file), capture (e.g. print to an 'image' format), or whatever the organization decides is an appropriate way of preserving those aspects of the data objects that it considers valuable. Moreover, provided that the bit-preservation has been done correctly, if it should arise at some far future time that there is some valuable but unanticipated aspect of the original that needs to be recovered, then *in principle* it should be possible to create a converter de-novo.

Functional preservation is domain and application specific, and hence it is not easy to be as clear and precise about what constitutes good practice, since it depends on who is going to be using which objects for what purpose… Even so, there are some precautionary principles that can be enumerated.

1. Where possible the content should self-contained, i.e. it should not rely on external facilities, services or commonly distributed data, no matter how

currently ubiquitous… for example, interpretation should not rely on external fonts, links to web-pages, or other potentially ephemeral data.

2. If there is need for visually exact preservation then a frozen format such as PDF-A, or even TIFF/PNG should be used.
3. For image data a lossless format should be used, and in general any compression should be lossless.
4. Obscure applications and formats should be avoided … where possible convert to a mainstream format. Consider including abstracted data alongside the application specific format, and if possible include documentation in the archive that describes the format in sufficient detail to allow future deconstruction.