Deliverable D12.4 Storage Technical Architecture

A Strategy and Architecture for Storage

DOCUMENT IDENTIFIER	PS_WP12_BBC_D12-4_Storage_Technical_Architecture
DATE	27.02.2006
ABSTRACT	This document sets out the specific requirements for using storage within the technical work of the PrestoSpace project. It reviews and attempts to complete the architectural specifications and requirements of the project. It adds detail about handling of files and of the key media object of the PrestoSpace data model – the EDOB (Editorial Object).
KEYWORDS	storage architecture data model system
DISSEMINATION	Public
WORKPACKAGE / TASK	WP12 Storage Architecture
AUTHOR, COMPANY	Richard Wright, BBC
INTERNAL REVIEWERS	Ant Miller, BBC

DOCUMENT HISTORY

Release	Date	Reason of change	Status	Distribution
1.0	06.05.2005	First Draft	Living	Confidential
1.1	22.08.2005	First draft updated by Hi-Stor	Living	Confidential
1.2	07.11.2005	Full version	Living	Confidential
1.3	09.12.2005	Final version	Living	Confidential
1.4	27.02.2006	Final version revised	Final	Confidential
2.0	28/02/2008	Dissemination status changed to Public	Closed	Public

Contents Table

1. Document Scope

This document sets out the specific requirements for using storage within the technical work of the PrestoSpace project. The central issue is long-term storage, but the related short-term issues of moving files into and out of long-term storage are also mentioned. It reviews the work already done by the System Architecture work package, and the architecture work done within the MAD and RES area, so that SAM (Storage and Archive Management) need only add anything missing, especially regarding data interchange, external storage and long-term storage.

2. Executive Summary

This document is an overview of technical requirements for the storage component of a PrestoSpace technical architecture. As such it is meant to be relevant to software development in PrestoSpace technical areas. It reviews and attempts to complete the architectural specifications and requirements of the project. It adds detail about handling of files and of the key media object of the PrestoSpace data model – the EDOB (Editorial Object). It is intended as an internal document, though the discussion of "levels of storage" will be developed for the SAM website.

3. Overview

This document sets out the specific requirements for using storage within the technical work of the PrestoSpace project.

Among the key goals for storage are:

- Data integrity,
- Ease of future use and transport or exchange of stored information,
- Longevity of stored information,
- Network and storage efficiency.
- Fast and easy access to metadata
- Bandwidth to share the stored content

In order to achieve these goals, this document also covers the following issues

- In the long term, what is being stored? an EDOB (Editorial Object, the essential working unit of the PrestoSpace data model), or "Material" or "Essence" which can be divided down (to frames, lines, pixels; samples for audio), or multiple EDOBs, or essence and metadata
- Time alignment
- Time-based vs. file-based media, and the general issue of whether files can or should have equivalents to temporal properties (such as the lines and frames of video signals)
- Unique, persistent identifiers
- Protocols for file movement

Files: while files are mentioned at various places in the document, no details are given regarding specific file formats – beyond quoting references to MXF and BWF in section 4.2. For the internal purposes of PrestoSpace, there is a separate document **PS_Data_Exchange_Formats_V2.1.doc** covering file formats.

The relationship between this document and the other architectural work of the project is set out in Section 4.

The storage architecture requirements for ‘keeping everything together’, and keeping it indefinitely, are set out in Sections 5 Levels; 6 Units; and 7 Functionality – and formalised in section 8.

4. Relationship between Storage Architecture and Overall PrestoSpace Architecture

The System Architecture Workpackage defines the overall structure and integration for those technical parts of Presto-Space that need to work together. It is summarised below (4.2). The Metadata Access and Delivery (MAD) work area has done the most detailed modelling, because of their requirement for software integration. That work is summarised in section 4.3.

The overall system architecture shows the work areas, and high-level description of their interconnections, in four areas:

- Metadata
- Essence
- Transactions
- Original Media

The detailed work of MAD also has a high level description of components and interconnections, but adds two layers of detail:

- Interconnection protocols
- A formal data model

In particular, the data model defines a basic working unit (for all processing) the EDOB (editorial object). An EDOB is realised as MATERIAL which requires a SOURCE, which is located in STORAGE.

A storage architecture needs to agree with the above, and catch anything missing. The specific concerns of the storage architecture are :

- Digital storage; analogue media may have a bar-code or identifier that carries over into digital storage, but have no other system architecture issues. [This statement applies specifically to the media; the metadata, such as a catalogue, associated with analogue media has just the same potential system involvement as for digital media metadata.]
- Storage of files; there are ways to put digital information on media without using files (eg audio CDs), but such storage is inaccessible to computer systems.
- Keeping everything together that belongs together, which requires a system-level decision about whether, for long-term storage or for exchange between

storage systems, everything (about one EDOB) is in one file or whether there is a basic unit consisting of:

  - Multiple files
  - Which together represent the file-based equivalent of an analogue film, or video or audio recording (and associated documentation)
- Keeping it indefinitely: condition monitoring, detecting errors or deterioration in the storage data, determining the best migration point, and a strategy to migrate or renew data with minimum total lifecycle cost.

System architecture, the PrestoSpace data model, and the workflow of a preservation factory and of a turnkey system – are all implicated in 'storage architecture'. So the next section attempts to separate these concerns.

4.1. Data Model, Architecture and Workflow

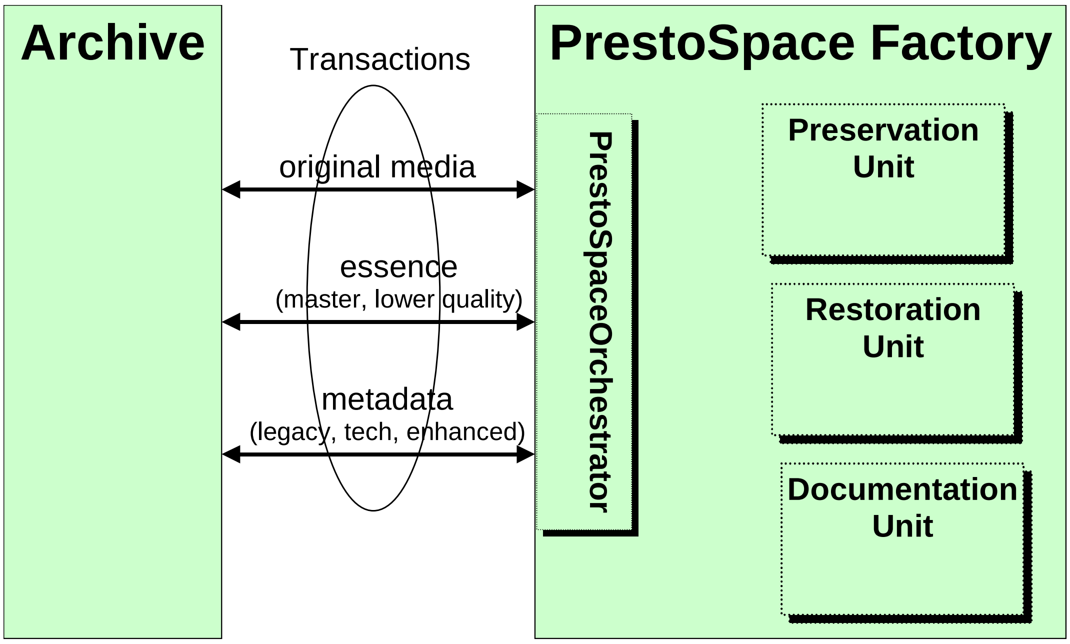
Data Model, Architecture and Workflow (process) are three related issues. A very brief discussion of the essential requirements of a secure process of moving data files into and out of storage is given in this paper (7.2.2). The focus of this paper is on architecture (for storage). However, a Data Model is tied directly to architecture (at least in the MAD work, with Eurix doing the architecture (TF1) and RAI the data model (TF2)).

The next sections summarise the existing PrestoSpace Architecture and Data Model, as it relates to storage.

4.2. PrestoSpace System Architecture

The PrestoSpace system architecture has been summarised in: **PS_D3.2_SAS2_Gene_Deliv_Spec_v5.doc** and
D3.1 SAS1 : Models and Protocols for PrestoSpace Factory Process

As shown in the following diagram, the system architecture moves 'original media' into the Factory, and then moves 'essence' from the various Units of the Factory back to the Archive. Internal Factory movements are not show.



System Constraints:

Some issues are invisible in a system architecture chart. In particular, it is easy to show an arrow, but an arrow indicating a data connecting between systems in the same location has different practical implications than for a data connection between systems in different locations. In the first case (same site) there are many ways to connect storage systems, or share a common storage. In the second (long distance) there are all the practical questions of how best to move data from place to place: wide area connection or physical transport of data (via datatape, removable discs and so forth). While it is messy to introduce such considerations, 'not all arrows are equal' and so practicalities and choices do lie invisibly beneath the above diagram.

As shown in the table below[1], essence moves as a file into and out of MAD and RES factories, probably as BWF (broadcast wave format, for audio) and MXF (for video and digitised film, but can hold audio as well).

[1] D3.1 SAS1 : Models and Protocols for PrestoSpace Factory Process v2.09, Table 8.1, p16

	PRE	RES	MAD
Metadata inputs	PreservationBatch XML DocumentSamples and Schema are given in [PXML]	EOD ([MDF])	EOD [MDF]More precisely requirements are: EDOB ID, Duration and whatever <REALIZATION> node (defining the associated Materials).
Metadata outputs	in EOD ([MDF]) format. MaterialAccessDocument defined in [MDF], the examples and Schema of which are provided in [PXML]Migration MD (Media-Matters MD ([PSAM]), Rich Digitisation MD ([D8.3]), ….), contained in EOD format ([MDF]).	Restoration Documentation (proprietary format) + Defect&Quality description specified in [D8.3], contained in EOD format ([MDF]).	EOD providing the enhanced metadata ([MDF][D16.4])Export format: EOD.
Essence inputs	None	**Audio**: BWF PCM 48kHz or 96kHz, 24bits, up to 2 tracks per file, more files if more tracks **AV**: As PRE output Optional: digital video tape.	**Audio**: BWF PCM 48kHz or 96kHz, 24bits, up to 2 tracks per file, more files if more tracks **AV**: MPEG2 4-8 Mbps
Essence outputs	**Audio**: BWF PCM 48kHz or 96kHz, 24bits, up to 2 tracks per file, more files if more tracks **AV**: One or several of the following • MJPEG2000 lossless (SD, HD, higher…) + BWF files, PCM 48kHz, 16bits • MPEG2 4:2:2 CBR from 15 long GOP to 50I (programme stream, to be confirmed) (audio 48 kHz 16bits uncompressed (to be confirmed) 4 tracks (to be confirmed)) • MP4 AVC • MPEG2 4-8 Mbps for DOC/CA quality Optional: digital video tape.	Same as Essence inputs.	Jpg/png/gif images provided by the content analysis modules (key frame extraction). Optional: video copy at web browsing quality for the Publication Platform
Original media inputs	Tapes, film rolls, disks, in batch boxes.	Not applicable	Not applicable
Original media outputs	Tapes, film rolls, disks, + affixed labels and barcodes, in batch boxes.	Not applicable	Not applicable
Transaction s	The preservationBatch XML Document is defined according to the model given in **Erreur ! Source du renvoi introuvable.** and includes the working options and parameters. Methods exposed 1. importBatch Methods required 2. insertMaterial	To be defined. See RES/MAD integration draft in [RADS], section 4.1.6	XML Document reporting the **profile** for the jobs to be done and the identifier of the EOD to be elaborated Methods exposed registerEDOB insertJob insertMaterial deleteEDOB purgeEDOB getAllMaterials
Workflow (Tracking, Consistency)	Input: Batch identifier or Item identifier. Methods exposed: 1. getBatchStatus 2. getItemStatus 1. search	To be defined. See RES/MAD integration draft in [RADS], section 4.1.6	Input: JOB identifier or EDBO identifier Methods exposed: getJob search

Interfaces between PRE, RES and Documentation Units

So far, the requirement for storage is to hold BWF and MXF files. Further development of PrestoSpace documentation is expected to address the issue of MXF patterns [there are variations on the basic MXF format] because that decision could affect the type of equipment chosen for storage.

If an EDOB can be part of file rather than a complete file (see 6.3 Going Down: Parts of Files), the storage system should provide an efficient method of ‘sub-file access’, normally called a “partial restore function” within the storage industry.

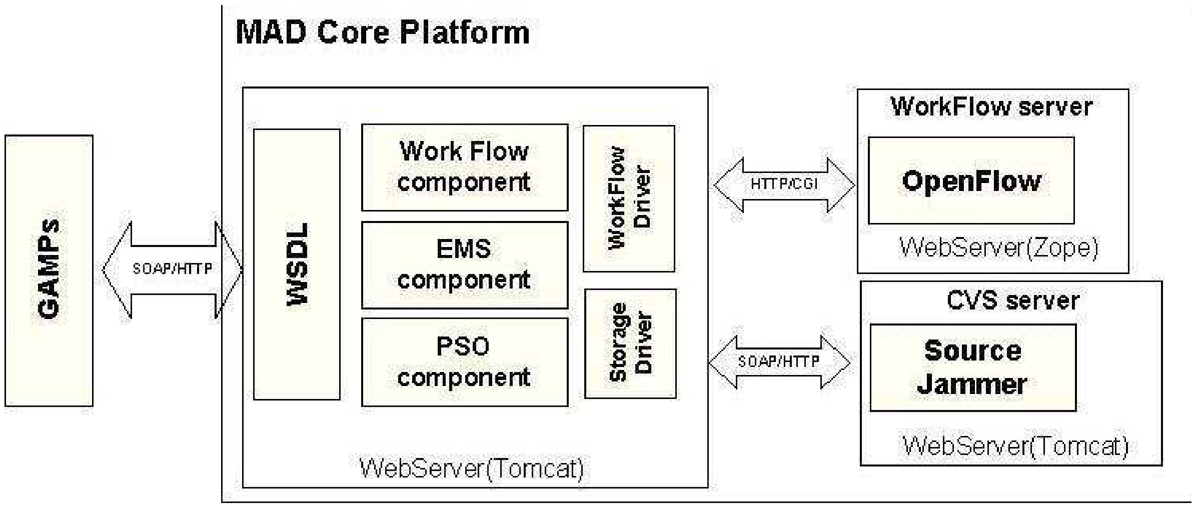
Two other things move: Metadata and Schema. Both will be expressed in XML, and can either be stored in text files, or can be ‘wrapped’ with essence in BWF and MXF files. BWF has a limited capacity for holding metadata (because it is very simple), but MXF has no real limits (and is accordingly dauntingly complex).

Digitisation also creates metadata, as do both MAD and Restoration, but such metadata only add bulk to the storage requirement – it doesn’t add any new types of storage requirement.

4.3. MAD Architecture and Data Model

The MAD architecture is given in **Internal MAD Arch overview.2004-11-10.doc**. As seen in the following figure, the whole ‘core platform’ is assumed to be stored (held) on a computer system, a set of web servers.

The storage part of this architecture is the EMS = *Essence and Metadata Store component.* “which is responsible for managing storage and the concurrent version of metadata exchanged”. This component communicates with physical storage.



So how does it communicate? There are two strands to the answer: the overall communication protocol (the language), and the details of the communication (sentences in the language), which in turn make reference to elements of the MAD data model (**TF2_RFC_2004-06-10.doc**).

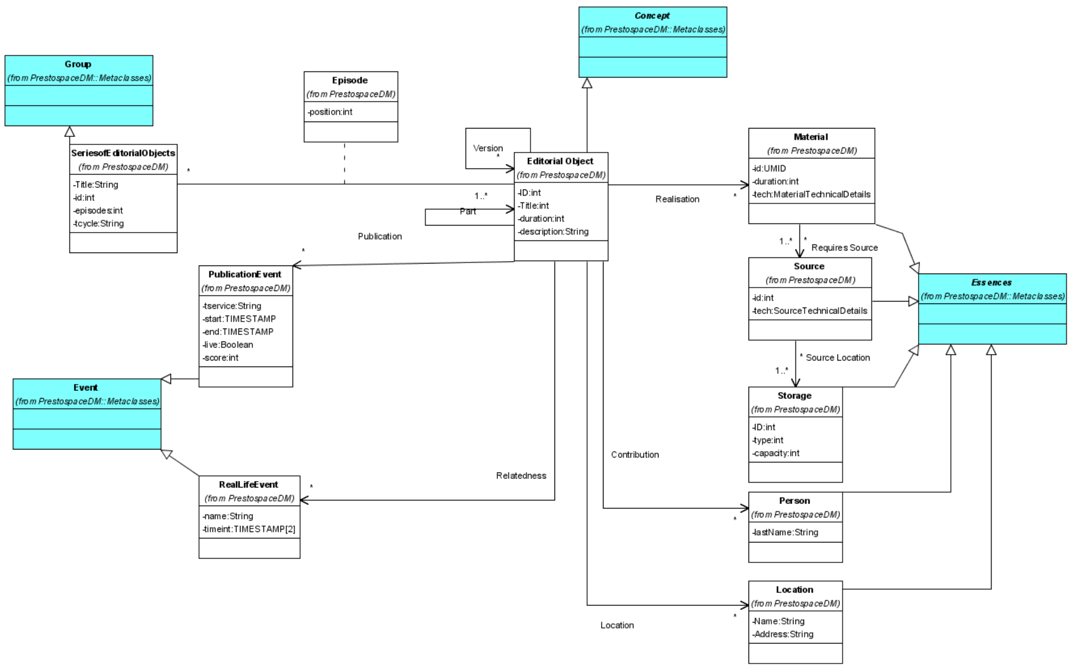
The protocol for communication with the EMS is given in **Eurix internal MAD architecture.2004-11-10.pdf,** where a ‘file server’ component is introduced. EMS is further specifed as (along with everything else) a web service, with four functions:

- getMaterial
- insertMaterial

- checkinEDOB
- checkoutEDOB

EDOB is the editorial object, which is a fundamental unit of processing within MAD (and, by implication, all of PrestoSpace). It is shown in the following diagram, taken from **TF2_RFC_2004-06-10.doc.** An EDOB is related to MATERIAL be being a 'realisation'. The important points are:

- **EDOB**, **MATERIAL**, and **realisation** are formally defined in the data model;
- Checkin and checkout imply locking, to prevent more than one task changing an EDOB, which would lead to corruption of data. So the only functions that actually apply to physical storage are **getMaterial** and **insertMaterial.**



There is a proposal (TF2 Eurix.2004-07-01.doc) to add the following elements (relevant to storage):

- **File**: This element should be a subclass of Storage element in the class diagram hierarchy. We need it for a physical representation of resources.

- **Locator**: This element has the responsibility to provide a physical reference of the resource. We should assume we can have multiple locator elements for a single file.

- **Protocol**: This element has the responsibility to provide the protocol information for accessing resources.

- **Context**: This is an optional element for describing the actual path for accessing the file name in the File Element.

Interpretation: the intended effect of **getMaterial** and **insertMaterial** is to move a file into or out of the **EMS**. An **EDOB** references a **File**, using **Locators.** The implication is that Locators are pointers, so that an EDOB can reference a part of a file. **Protocol** and **Context** provide the method of getting the overall system to ‘call up’ a file.

Further work must be done to give details of these elements. The structure as given may suffice, and no further Storage Architecture may be needed. However the next sections cover issues not explicitly addressed by the above architecture and data model descriptions.

4.4. WA Restoration: System Architecture

Additional requirements and detail have been supplied by the Restoration Work Area, in the document **WA RES Architecture V5.0**.

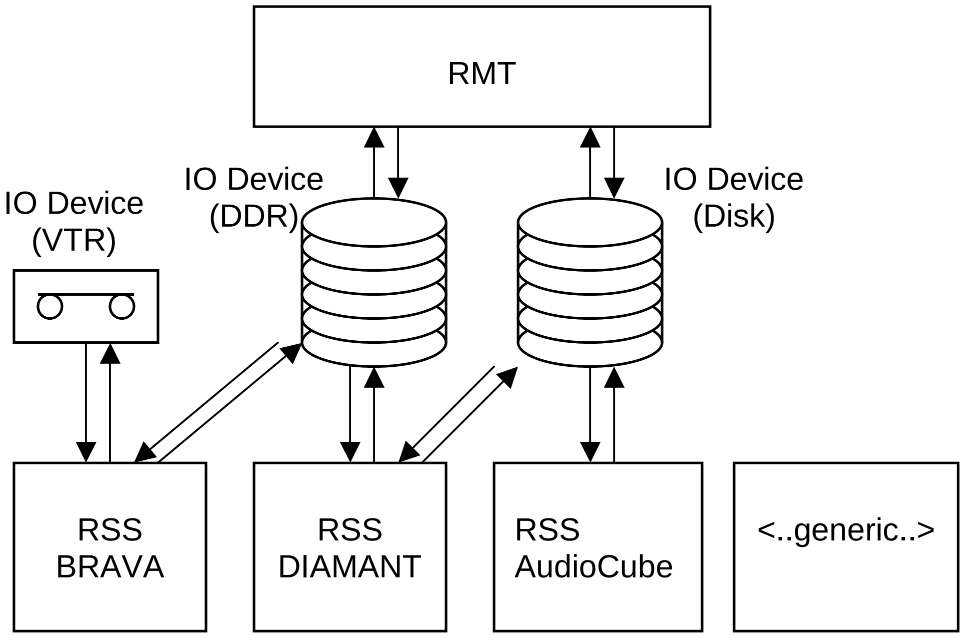
Their general requirements are:

- Data input and output is file based
- Data access (read/write) is possible by standard file system file I/O functionality
- Data Access bandwidth must be scalable by selection of appropriate storage system technology (e.g. for ‘SD (720x576), 25fps, 4:4:4 (3x8 bit), uncompressed, real-time’ the sustained bandwidth requirement is about 32MByte/s, for HD (1920x1080) about 160 MByte/s).

The storage system will be very different depending on the data access bandwidth and number of simultaneous accesses needed..

Integrated HW/SW audiovisual Restoration System with its main dataflow.

RMT – Restoration Management Tool
RSS – Restoration Subsystems
VTR – Videotape Recorder
DDR – Fast computer memory



The Restoration Work Area approach includes defining a *media item*, and a *data track.* However the document says that communication with other work areas will be based on "Import/Export of PrestoSpace Factory objects (EDOB)" (Sect 4.1, p7) which implies that an overall PrestoSpace storage architecture capable of supplying EDOB's to Restoration (and receiving them back) will suffice (though this statement again implicitly raises the issue of how the content is exchanged, by which technology and over what distances and subject to what bandwidth and time constraints.

Finally, a storage architecture for RES needs to support multiple and possibly concurrent users and systems. This is a complex question that has major impact in the workflow. One method of managing access for multiple applications is to use Distributed File Systems (DFS) to share and lock content stored. This adds complexity to the storage system but provides advantages in terms of workflow.

However this document cannot be more specific regarding RES requirements, as RES activity may rely on a local storage buffer (which implements such technology as managing concurrent users) and only use straightforward file access to and from the main 'PrestoSpace Store'. In short, the overall storage architecture may not need to be concerned with details of RES (or of MAD) operations, as they may copy material to local working storage on technology that satisfies their local requirements.

5. Levels of Physical Description

The only specific unit of storage so far described is the file. Formally, a file is a logical unit, which may reside in the active memory of a computer system, but is generally understood to reside in a (semi)permanent state on one (or more!) storage devices.

The following discussion presents a formal structure for describing physical storage as used by computer systems. Throughout, the physical storage will be described in terms of its role in handling files of data.

This material is not essential for the technical storage model needed for PrestoSpace, but it should prove useful as an overall framework for a generic storage architecture description for the SAM website.

5.1. Strategy

A storage *strategy* involves one or more *systems*. The defining property of a *strategy* is that it holds more than one distinctly-addressable copy of a file; a storage *strategy* provides redundancy (and therefore protection against failure). The file replication can be implemented by the following basic methods:

- Mirroring: one or more identical copies of the basic storage *system*
- Backup: a process copies data off the *system* onto something else, which is typically something slower and cheaper
- Alternative copy: when a file goes onto the storage *system,* it also goes onto something else

The important point is: without replication, there is no storage **strategy;** there is only a storage system (or something below the level of system).

5.2. System

A storage system holds a single copy of a file. This feature is essential to provide a unique unambiguous address, but as just discussed it does not provide redundancy – and so multiple systems, or a system plus ‘something else’, are needed for security. The most complex systems have multiple clusters – about to be defined – but a system could default down to the level of a single device (in which case it just complicates things to call it a system).

For significant amounts of storage, this is the basic unit of management. Unfortunately this management is now largely manual (by system administrators), Automation of management of a storage system (including storage area networks and related concepts) is an area of rapid development in the storage industry, because manual management is a severe bottleneck to development of large amounts of disc storage.

Reality check: file systems and storage systems: we all work with computers that have file systems, not storage systems – and we can all save files in multiple places in the file system – with identical names (providing they are in different locations with the typical tree structure of a file system). What this document is defining is a somewhat idealised strict hierarchy, for principled storage management. It requires an unambiguous name and location for an entity (a file) within a **system.** Redundancy is achieved by having more than one system in the strategy.

While we all live and work with file systems, they are in no sense a principled approach to storage and in fact are more than willing to support the most unprincipled processes: multiple files with the same name in different locations within the system; no control over changing locations; no control over entry and exit. Standard computer file systems don't do anything to prevent loss of files – as proven by the unconscionable amounts of time spent hunting for files.

5.3. Cluster

Multiple devices (see next level) working together form a cluster. The most common type is RAID: redundant array of independent discs, which uses multiple devices to form an entity which is seen by the system as a unit, but which has storage integrity, speed and capacity characteristics in excess of anything available from a single storage device.

5.4. Device

A device is a basic unit which the storage industry manufactures, and which is capable of:

- connection to a computer or network
- reading and writing data (onto itself or onto a removable medium)

a hard disc drive, a data tape reader/writer, a DVD reader/writer. Whether a data tape or DVD robot is a medium or a cluster is arguable. It seems sensible to regard it as a cluster only if it has multiple read/write units capable of handling multiple data streams.

A person could argue that one device can have multiple instances of the same data file, and so fulfil the definition of being a storage strategy. Such a person would be a total fool.

5.5. Medium

Storage devices use optical or magnetic media. There is a storage medium built into hard disc drives – namely platters coated with magnetic material. Other devices use removable media: MO-disc, CD, DVD, data tape, floppy disc.

It is generally a convenience for a file to reside on a single medium. For removable media, this is an absolute requirement. For hard drives, especially when organised in a raid array, a file will be spread (striped) across multiple devices and will benefit in the process, because the RAID methodology includes built-in error correction to recover from device failure.

There is a problem with large files, at multiple levels: medium, device, cluster and storage system (and computer operating system) may have a limit on file size that causes a potential restriction for audiovisual files. Gradually systems are getting larger upper limits, but it remains an issue for removable media.

It may be necessary for PrestoSpace to have a systematic way to deal with large files by breaking them into smaller files for removable media, without losing anything. This situation could arise for a small turn-key system using a DVD robot, for instance..

5.6. Formatting

A medium has to be formatted so that the read/write device knows where to find the bits. They do not, in general, find themselves. There are complex standards for things like track size and position on magnetic and optical media – and there are device-specific format rules (which may be computer-system specific), which is why different personal computer systems couldn't reach each other's floppy discs.

The key issue for storage and data longevity is that no assumption can be made about readability of data unless the formatting is known, and a device is available which follows that format. This issue leads to the same problems of obsolescence and forced migration as have been the case with the various formats of videotape.

5.7. Bits

There will be a specific method for altering the magnetic or optical characteristics of the storage media. Some of these methods, like the various kinds of dye used in writeable CDs, produce end results that can all be read by the same generic reader. Other read/write technology is specific, so that a tape written by a certain method requires a matched reader. This situation is part of the reason for multiple incompatible formats of data tape – the other reasons being incompatible formatting, and incompatible sizes of the actual tape cassettes (the media).

6. Units

The basic actual unit of file input/output from storage is the file, as covered in section 4. However the basic conceptual unit is the EDOB, at least for Restoration and MAD. The ability to deal with storage and input/output of an EDOB element means that units both above and below the file will be important.

6.1. File

It may be that an EDOB can be held in a file, but such a file will be a "wrapper" of diverse components: video, audio, metadata. Most internal actions will want to deal with these components, and move them about as files.

6.2. Going Up: Groups of Files

In order to keep such component files coherent, there needs to be a unit of organisation above the file level. If the EDOB can perform that function, then all is well. However the existing data model has a complex connection between EDOB and files input/output. **The architecture needs a unit above the file level that can effectively manage movement of multiple files into and out of storage** (which may or may not be the EDOB)**.**

6.3. Going Down: Parts of Files

The problem with defining an EDOB independently, free from file and storage issues, is that an EDOB can not only be multiple files, **it can also be a part of a file** (when just a bit of 'essence' is identified for processing. This *sub-file EDOB* will be fairly common in restoration.

The situation in practice may well be even more complicated: a sub-file of video may need to be aligned and maintained with a separate sub-file of audio and yet another (or several more) of metadata.

The situation just described amounts to simultaneously moving below and above the file level, and managing storage of the resultant new 'sub-file, multi-file' EDOB (as one of more new whole files, presumably).

The SAM responsibility for storage architecture would happily stop at providing file-based access to storage, but **somewhere in the interface between work areas and actual storage there has to be a mechanism to handle the fact that an EDOB is not a file**.

7. Functionality

This section describes the methods of storage access, the basic access functions (defined extensively in SAM D14.1), and finally gives the basic capacities and transfer capabilities needs for standard audiovisual processes.

7.1. Method of Access

There are three essential requirements for access to stored multimedia content:

a) Unambiguous identification of files, using a persistent identifier
b) Access to portions of essence files, by timecode
c) Access to the structure of the essence:
    - To individual audio tracks, for multi-track audio
    - To fields, frames, lines, and pixels for digital video
    - To individual colour components

a) Unambiguous identification: **A system needs to be established, to run across the entire PrestoSpace workflow, for maintaining file identification, maintaining identification across associated files, and maintaining this identification across non-file elements coming into (or even out of) the system,** specifically video (tape) and audio (disc, tape) recordings (and similar non-file-based media).

b), c) Access to portions of essence files, and to the structure of the essence: File-based access, as discussed in the previous section, will support these requirements ONLY if there is technical (also called structural) metadata to inform the system of the sampling rate, frame rate, lines per frame, samples per line and any other data defining how the material was digitised. Most removable storage system, eg tape drives, will not natively support an efficient way to access portions of files. To provide such function on tape archives, software companies have developed applications, but these require archive management software on top of the storage which adds complexity to the system.

Material coming from the Preservation work area will need to have this technical metadata associated with the essence, either wrapped in one file, or through separate essence and metadata files associated in some way under the overall EDOB concept.

> [As an aside, it is quite possible that a file format that “understood“ the structure of the essence would be more robust than existing file structures, because a read error could then be confined to an individual pixel, video line, field or frame. One approach is to write individual files at some lower level – this approach is used in image process where it is called the ’single-frame per file’ method. It may well be far too inefficient to write anything smaller than a full image per file. However if the formatting of actual storage media had a low-level unit corresponding to, for instance, a video line – then “line-dropout and concealment“ could be implemented on file-based storage. The downside is that this method would lead to (video) format-specific data storage systems, which leads in turn to incompatibility, obsolescence and further cause of data migrations.]

There is one existing use of a file-based approach to storage which does use a media unit. The Cineon format for digital film and digital special effects used a single frame per file, and then it is up to an asset-management application, or a systems managers, to ensure that all these ‘frame files’ stay together, and in the correct order. The “single frame per file method” has some drawback on the storage and access side since a large number of small entities will create, for hard drive systems and for data tape, a significant loss of performance. Mass storage devices, and in particular sequential access devices (data tape) need to be used with large chunks of data to be efficient.

7.2. Access Functions (and Processes)

Storage is not passive: it has to do something (although engineers do make jokes about ‘write-only’ discs)
.

7.2.1. Functions under the control of Processes

Storage is of no value unless files can be written and then read back. Reading and writing are basic *functions* performed by any useful storage device. However reading and writing of files are also the active parts of vital archive or ‘collection management’ *processes* (ingest, update, delete, move). The careful definition and implementation of archive processes is the subject of a separate report (D14.1). The important point to be stressed here is that all basic storage functions must operate within well-defined processes, or chaos (and loss of material) will ensue. All storage devices allow data to be created and destroyed (write and delete functions). Storage management processes are the method for ensuring that data is created or destroyed in a fashion that is consistent with overall content integrity.

If a collection is being managed in a largely manual fashion, without the overall control of an IT system, then it is up to the individuals in charge to ensure that no files are created or deleted in error.

It is very difficult to ensure against manual error in storage system maintenance, which is why it is important to use an overall 'collection management' or 'asset management' IT system if possible.

It is also important to ensure that manual intervention is not used to bypass the processes (manual or automated) governing maintenance of a collection. A process is essentially a set of constraints, and if the constraints are bypassed, the integrity of the collection is at risk.

The major problem with current storage systems is the amount of manual intervention still needed to manage the storage (mount and dismount volumes, schedule backups, delete redundant files) even when supposed automated management is in use. Such manual storage management (at the system supervisor level) has been identified within the storage industry as a major cost, and bottleneck for the growth of storage [Honeycomb; related automation of management projects]. What is not so often mentioned is that manual management of storage (and backups) remains the major source of loss of stored data [ref].

The conclusion is that the following list of basic storage functions represents what storage systems have to do – but these functions MUST be under the control of effective procedures: collection management processes that ensure (so far as it can be ensured) that these functions are used to maintain an archive, not destroy it!

7.2.2. Basic Processes

There are many systems and standards for ensuring storage integrity, and archive/library/collection integrity. D14.1 describes the comprehensive standard for coherent processes in the museum world, which is directly applicable to audiovisual archives.

However for small collections that are not able to fully implement the recommendations of D14.1, and especially for collections that work with manual systems (card catalogue or register; discrete ”media on shelves“), the following list gives the most basic processes for dealing with storage without undue risk of loss. The processes are meant to be equally applicable to files on storage devices and to discrete media such as CDs of shelves.

- **New Item**: this is the least risk, because it doesn't involve deleting anything. It can cause confusion if the new item isn't given a new and hence unique identifier.
  - Check that the proposed identification is not already in use
  - Identify the item
  - Add identification information to the catalogue or other collection documentation
  - Assign physical storage

- **Un-needed Item**
  - Ensure the reason why the item isn't needed is valid. For instance, if the item has been replaced by a new copy on new media – then the new copy must pass all checks before the old copy is deleted.
  - Remove documentation (ideally keep a record of deletions!)
  - Remove from physical storage (ideally to a temporary area before making a permanent deletion)

- **Replacement Item**
  - This is a combination of the New and Un-needed item processes, except the identification need not be new. However the documentation needs to be updated to reflect the fact that an item has been replaced.

7.2.3. Basic Functions

The basic functions (at the file level) of any access to storage are:

2. read a file;
3. write a file, which has two options
   - write a new file (which may need a "create" function)
   - write a new version of existing data (which is usually implemented as 'write new' plus 'delete old')
4. create a new file
5. delete a file
6. copy
   - create, write
7. preserve: this is NOT a standard basic function, but it should be. At the storage strategy level (see section 5.1), preserve means create the same file on a different storage system, read the current file and write to the identical 'clone' on the *preservation* system. Then mark in some way that this operation has taken place. Backup and Mirror would be two variants of Preserve.

7.3. Storage Capacity

Much additional detail is on the SAM website, but here is a brief summary.

Audiovisual requirements

Audio:

Quality	Bits per second	Bytes per hour
24-bit, 96k sampling	4.6 M	2 G
CD: 16-bit, 44.1 k	1.4 M	0.63 G
Broadcast	200 to 400 k	100 to 200 M
Web	10 to 100 k	5 to 50 M

Video (SD):

Quality	Bits per second	Bytes per hour
Uncompressed SD	200 M	100 G
Lossless compression	80 M	40 G
High-end compressed	25 to 50 M	12 to 25 G
Broadcast	4 to 8 M	2 to 4 G
Web	100 k to 1 M	50M to 500 M

Film (HD):

Quality	Bits per second	Bytes per hour
Lossless compression	300 M	150 G
High-end compressed	100 to 200 M	50 to 100 G
Broadcast	8 to 20 M	4 to 10 G
Web	200 k to 2 M	100M to 1 G

Rules of thumb:

- megabits/sec to gigabytes per hour: **divide by two**. The result is a bit too high; subtract 10% to get the exact answer.

- shelf length to gigabytes:
    - **Video**, high quality: a one-meter shelf holds roughly 30 videotapes, which is about 15 hours (for the BBC at least). At 50 M b/s quality, that's 25 GB/hour (see previous rule). So that 375 GB per meter.
        - Lossless compression is higher, say 600 GB per meter.
        - Browse video at 500k b/s is only 4 GB per meter

    - **Audiotape**: 40 hours/meter (assuming half 60-minute and half 30-minute reels) at 0.7 GB/hr = 30 GB per meter

    - **Film:** sticking with 15 hours per shelf, but coding at HD, lossless, implies 160 GB/hr = 2.4 TB per meter.

- film footage to hours
    - 16mm: (40 frames/foot; 24 frames per second)
      so 0.6 feet/second => **36 feet per hour**
    - 35mm: (16 frames/foot; 24 frames per second)
      so 1.5 feet/second => **90 feet per hour**

7.4. Storage Transfer Rate

Transfer rates are important for overall system efficiency, but for systems which handle media as files, there is no absolute transfer rate requirement. It is only at the points when material is captured (digitised), and again when it is played out for viewing and listening, that there is a strict transfer rate requirement.

Audiovisual requirements

Media	Full quality	Browse Quality
Audio	Up to 1.4 Mb/s	30 to 300 K b/s
Video, Standard Definition	Up to 270 M b/s	300 K to 4 M b/s
Video, High Definition	Up to 1 G b/s	500k to 12 M b/s
Film, 16mm (2k sampling)	Up to 2 G b/s	500k to 12 M b/s
Film, 35mm (4k sampling)	Up to 8 G b/s	500k to 20 M b/s

8. Files and Alternatives

There are storage technologies coming to the market that blur the boundaries between system architecture and storage architecture. This document cannot do more than point out the existence of these approaches.

Storage systems have the possibility to work at (roughly) three levels:

- block level: this is the lowest functional level of the storage device; all transfers to and from the device are done as entire blocks of data. Typically the block is a physical reality: there are physical markers written on the media defining the blocks.
- file level: this is a logical level, and is usually the lowest level of access by the user of a computer system. An entire file is written to a device, and read from it.
- something more general than files, such as object-based storage and content-addressable storage:
  - object level: object-based storage devices are being introduced on the market called OSD and promoted by SNIA, the Storage Network Industry Association. The Object-Based Storage Devices (OSD)enable the creation of self-managed, heterogeneous, shared storage for storage networks. OSD will focus on moving low-level storage functions into the storage device itself, accessing the device through a standard object interface. This object level in the device may have some impact in the storage data model, but it may be too early to see how this approach will be accepted by the market[2].
  - content-addressable storage (CAS): as with object-based storage, CAS allows a ‘thing’ (which could be more complex than a single file) to be stored along with a description. The metadata could be computed from the ‘thing’, as in a fingerprint of an image or a hash-code of any digital entity. These are technical areas that rapidly exceeded the sensible bounds of an attempt at a basic architecture[3]. The key issue with both OSD and CAS is that they are alternatives to filing systems, and so can remove a dependency at the storage level – though unless the rest of the system also bypasses files, there is a net increase in complexity as both files and objects/’things’ will be present.

The file level is probably the only practical choice for PrestoSpace as block level is too low, too complicated and above all too device dependent. The object level and associated technology such as content-addressable storage are currently vendor-specific, so again not a useful PrestoSpace choice.

[2] http://www.snia.org/tech_activities/workgroups/osd/ This standard is still being normalised but devices are coming onto the market: e.g. EMC Centera “archives on disk”

[3] There are various products using Content Addressable Storage. Some basic information is here in the wikipedia: http://en.wikipedia.org/wiki/Computer_storage