Midnight Code Documentation

Digital Asset Management Guide

A Taxonomy For Digital Repositories

Classified: PUBLIC

Project Code: “Saturn”

Status: RELEASE
Version: 1.7
Issue Date: July 12, 2020
Document Type: CONTROLLED

Table of Contents

1 Context

Once a business is on to its third terabyte of unstructured data (or an individual starts working with more than a couple of USB drives worth of files), the question inevitably arises: "how do I manage all of these files?" The most intuitive computer interface for unstructured data is the file-system, and the ability to file content away and retrieve it quickly does not require a search engine or even database, so long as a logical taxonomy is articulately defined and easily understood, and naming conventions are clear and consistently followed.

2 Principles

A small number of principles have been used to guide the development of the taxonomy:

- **Create useful containers –**
  - It is common to direct a certain class of infrastructure and/or applications to a certain type of file; for example, digital signage will want a directory full of picture, flash, or video files. The taxonomy and naming conventions should support this type of grouping, such that applications can be associated with one or two directories and don't need to be given the entire file system to find their content of interest.
- **Store handles for, and not classification of, content –**
  - The file-system (directory and file names) should only contain sufficient information to successfully file and retrieve content in a useful way. Applications that consume certain file types (such as iTunes, for audio files) will acquire and store their own meta data, providing additional views of classification. Those views (such as Author, Genre, etc) should not be embedded into the on-disk structure.
  - The *absolute minimum* data required at the file-system layer would therefore be the data required to "look-up" the metadata (aka, the handle). This could mean storing documents named as their invoice number for example. However, the *practical minimum* data required at the file-system layer needs to include consideration of how users are to interact with the files – i.e. a single list of invoice numbered files may not be intuitive, but an issue date and invoice numbered file inside an organisational directory might be.
- **Remove ambiguity –**
  - The resultant taxonomy should not create opportunities for confusion (i.e. should I file this content here or here).
  - Minimising/removing metadata from the taxonomy and naming conventions will aid usability. For example, filing a Lecture in a directory of Lectures is straight-forward. Filing a Lecture in a structure of institutions that is parallel to a structure of lecture topics increases filing complexity and makes access less reliable.
- **Consider performance –**
  - Although the performance of the storage sub-system is not in the scope of the content management guide, poor taxonomic structure can impact storage performance.
  - Broadly speaking file-systems are designed as hash tables. If the application only ever adds/moves/deletes specific files then the directory structure has very little impact on the performance of the application. If the application provides some sort of *browsing* or *scanning* functionality then having millions of files in a single directory will increase load on the storage sub-system and decrease application performance, negatively impacting user experience.
- **Leverage existing standards –**
  - Where common structures and/or naming conventions exist, these will be leveraged.

3 Taxonomy

Per the principles above an initial taxonomy has been developed to provide guidance to people looking to intuitively manage their digital assets.

Some care has been taken to make the taxonomy technology agnostic. Note, for example, that colons are not used in the naming standards. There are no further constraints made by the taxonomy – for example multilingual / Unicode file and directory naming is permitted (encouraged).

Where practical the taxonomy is designed to be as *shallow* as possible, to mimic the common equivalent[1] physical structures:

```
Filing Cabinets
└── Drawers
    └── Folders
        └── Documents
```

In practice, the depth of the implemented directory hierarchy will depend on the number of digital artefacts that need to be filed and found, as well as the intensity of those artefacts (the amount of organisational focus within a given domain). As an organisation, an ISP might have deep *devices* structure, while a Print Media organisation might have none at all, for example.

Furthermore, the larger the organisation the more likely it is to have dedicated information systems (enterprise applications) that have been built to explicitly manage, store and retrieve the assets being discussed – and that blindly implementing the corresponding taxonomy may be a step *backward* for those organisations, circumventing systems and impeding their integrated processes.

Hence this taxonomy is a guide for digital repositories. While it does provide a considered body of *ready-to-roll* data structures, any implementation should be tuned to the specific needs of the target environment, its people and their processes.

This taxonomy is not complete and will continue to be varied over time. Additional structures, such as administration, business/strategic development, customers, marketing, products/services, and providers/suppliers will be detailed in future versions of this guide.

1 https://serverfault.com/a/578322

3.1 apps

The *apps* hierarchy is structured for on-disk application deployment, and execution.

```
apps
├── app1
├── app2
└── appn
    ├── afe
    ├── bin
    ├── bus
    ├── lib
    └── dat
```

As a top level structure, apps is assumed to be common (group) accessed – a community of users rather than by individual users. Where users have their own private apps, this structure can be leveraged in the *personal* hierarchy.

Specification

The sub-directory hierarchy is a directory per *app*.

When anticipating a substantial volume of applications consider additional classification. In an organisation this might be per market vertical, line of business, department or domain of expertise.

The naming convention for apps will be locally relevant, with observed examples including –

Directories	Description
Name	Full application name, for example: Financial Management Information System
Acronym	The full application name as an acronym: FMIS
Acronym Version	The acronym with the version appended: FMIS BECCA, or FMIS 2017

Note that the delimiter is the space character.

3.1.1 apps/app

Under apps, the sub-directory hierarchy is designed to support the live deployment of a multi-tier application.

Directories	Description
afe	The application front-end – whether platform-specific fat-client binaries/packages or a web front-end for the presentation tier.
bin	Binaries structure for the middle/back tier – includes platform specific binaries and platform agnostic scripts (Go, Node.js, Perl, PHP, Python, Ruby, Rust, etc)
bus	On-disk data structures that under-pin the app’s message/service bus.
lib	Libraries structure for the middle/back tier – includes platform specific and platform agnostic libraries.
dat	On-disk data structures that under-pin the app’s database.

Note that the delimiter s the space character.

Example

For a basic example, the Skynet consciousness would be stored as follows:

```
apps/Skynet 1997/dat/consciousness.db
```

Script

The following script has been authored to automate creation of the above skeleton directory hierarchy.

```
#!/bin/bash


if [ ${#} -ne 1 ]
then
  echo "Usage: ${0} <path to project>"
  exit 1
fi

tld="${1}"
if [ ! -d "${tld}" ]
then
  echo "Error: ${0} path \"${tld}\" is not a directory."
  exit 1
fi

(
cat << EOF
  afe
  bin
  bus
  lib
  dat
EOF
) | while read deep_dir
do
  mkdir -p "${tld}/${deep_dir}"
  if [ ! -d "${tld}/${deep_dir}" ]
  then
    echo "Error: failed to create path \"${deep_dir}\"."
    exit 1
  fi
  chmod 755 "${tld}/${deep_dir}"
done
```

3.2 crew

The *crew* hierarchy is structured for filing what would be, for a house-hold, the ***family*** records and what would be for a business it's ***human resources*** records – and should be named accordingly.

The richest taxonomy for this structure is the Changing Minds HR Taxonomy[2] which will be adopted for this structure in a business context, and reduced for the family context. Presented here in life-cycle order –

```
crew
├── definition
├── environment
├── recruitment
├── people
│   ├── person1
│   ├── person2
│   └── personn
│       ├── education
│       ├── health
│       ├── legal
│       ├── performance
│       └── references
├── motivation
├── education
├── management
├── conflict
├── support
├── absence
├── termination
└── compliance
```

As this pertains to local data there is no public metadata mapping available for handles/labelling.

As a top level structure, crew is assumed to be common (group) accessed – such as via an organisation's Human Resources department. Where users have their own private crew papers, this structure can be leveraged in the *personal* hierarchy.

Script

The following script has been authored to automate creation of the above skeleton directory hierarchy.

```
#!/bin/bash


if [ ${#} -ne 1 ]
then
  echo "Usage: ${0} <path to multimedia>"
  exit 1
fi

tld="${1}"
if [ ! -d "${tld}" ]
then
  echo "Error: ${0} path \"${tld}\" is not a directory."
  exit 1
fi

(
cat << EOF
  definition
  environment
  recruitment
  people
  motivation
  education
  management
  conflict
  support
```

2 https://changingminds.org/disciplines/hr/articles/hr_taxonomy.htm

```
  absence
  termination
  compliance
EOF
) | while read deep_dir
do
  mkdir -p "${tld}/${deep_dir}"
  if [ ! -d "${tld}/${deep_dir}" ]
  then
    echo "Error: failed to create path \"${deep_dir}\"."
    exit 1
  fi
  chmod 755 "${tld}/${deep_dir}"
done
```

3.2.1 crew/definition

The *definition* structure is business focused and captures all artefacts related job or role definition. The functions that this encapsulates are: organisation design, job design, job description, contract of employment, pay bands, pay rates (including assessment and negotiation), working time (including standard hours, out of hours, remote/at home, flexible working, standby/on-call, rest breaks), personal agreement, and team design.

3.2.2 crew/environment

The *environment* structure is business focused and captures all artefacts related to the work environment. The functions that this encapsulates are: code of conduct and corporate values/behaviours.

3.2.3 crew/recruitment

The *recruitment* structure is business focused and captures all artefacts related to the acquisition of personnel/talent. The functions that this encapsulates are: advertising, resume/CV filtering, interviewing, assessment centres, selection, and job offers.

3.2.4 crew/people

The *people* structure is both business and house-hold focused. For business, people captures all artefacts related to the personnel record. For the house-hold, people captures all artefacts related to family members.

Specification

Under the device, the sub-directory hierarchy is designed to provide operational support to the platform –

Directories	Description
education	For education and training records, including institution, curriculum, contact details, etc.
health	For medical records, including immunisation, allergies, management/treatment plans, letters from doctors, etc.
legal	For legal artefacts, permission slips, contracts (including service contracts and privacy agreements), copies of identity documents (including passport), etc.
performance	For benchmark results, such as report cards, sports gradings, etc.
references	For milestone and/or remarked achievements – graduations, distinctions, letters of recommendation, etc, as well as biographical material (resume/CV).

Note that the delimiter s the space character.

Script

The following script has been authored to automate creation of the above skeleton directory hierarchy.

```
#!/bin/bash


if [ ${#} -ne 1 ]
then
  echo "Usage: ${0} <path to project>"
  exit 1
fi

tld="${1}"
if [ ! -d "${tld}" ]
then
  echo "Error: ${0} path \"${tld}\" is not a directory."
  exit 1
fi

(
cat << EOF
  education
  health
  legal
  performance
  references
EOF
) | while read deep_dir
do
  mkdir -p "${tld}/${deep_dir}"
  if [ ! -d "${tld}/${deep_dir}" ]
  then
    echo "Error: failed to create path \"${deep_dir}\"."
    exit 1
  fi
  chmod 755 "${tld}/${deep_dir}"
done
```

3.2.5 crew/motivation

The *motivation* structure is business focused and captures all artefacts related to the incentivisation of the workforce. The functions that this encapsulates are: pay (including payroll, deductions and sick pay), bonuses, special recognition, pension, healthcare, holidays, and recreational activities.

3.2.6 crew/education

The *education* structure is business focused and captures all artefacts related to the education, training and awareness of the workforce. The functions that this encapsulates are: organisational development, personal development plan, induction (including day one, corporate, and site/location), and education (including training records, bursaries, standard classes, compliance, external learning, and informal learning).

3.2.7 crew/management

The *management* structure is business focused and captures all artefacts related to leadership in the workforce. The functions that this encapsulates are: strategy, leadership change management and management.

3.2.8 crew/conflict

The *conflict* structure is business focused and captures all artefacts related to conflict management within the workforce. The functions that this encapsulates are: grievances (including equality, bullying, unfair dismissal and more), disciplinary (including substance abuse, IT systems abuse, insubordination, bullying and more), and conflict management (including mediation and arbitration).

3.2.9 crew/support

The *support* structure is business focused and captures all artefacts related to workforce guidance. The functions that this encapsulates are: coaching, mentoring, and general advice.

3.2.10 crew/absence

The *absence* structure is business focused and captures all artefacts related to absence from work. The functions that this encapsulates are: booking and reporting absences, holidays (including standard leave, additional days and leave of absence), and sickness (including sick pay and long-term sickness), bereavement, maternity/paternity, and public duties.

3.2.11 crew/termination

The *termination* structure is business focused and captures all artefacts related to separation from the workforce. The functions that this encapsulates are: retirement, resignation, redundancy/retrenchment, sacking, and redeployment.

3.2.12 crew/compliance

The *compliance* structure is business focused and captures all artefacts related to the regulatory compliance of the workforce and work place. The functions that this encapsulates are: occupational health & safety (OH&S), security, equality, and employment law.

3.3 devices

The *devices* hierarchy is mostly structured for mapping to networked equipment.

This is intended to be a secured structure that enables disk-less devices (whether physical IoT devices, Servers, Workstations, etc, or virtual machines, Docker containers, etc) to map/mount volumes directly from the common storage fabric.

```
devices
├── build
│   ├── images
│   └── scripts
├── device1
├── device2
└── devicen
    ├── backups
    ├── boot
    ├── data
    ├── packages
    ├── patches
    ├── root
    └── scripts
```

Script

The following script has been authored to automate creation of the above skeleton directory hierarchy.

```
#!/bin/bash


if [ ${#} -ne 1 ]
then
  echo "Usage: ${0} <path to multimedia>"
  exit 1
fi

tld="${1}"
if [ ! -d "${tld}" ]
then
  echo "Error: ${0} path \"${tld}\" is not a directory."
  exit 1
fi

(
cat << EOF
  build/images
  build/scripts
EOF
) | while read deep_dir
do
  mkdir -p "${tld}/${deep_dir}"
  if [ ! -d "${tld}/${deep_dir}" ]
  then
    echo "Error: failed to create path \"${deep_dir}\"."
    exit 1
  fi
  chmod 755 "${tld}/${deep_dir}"
done
```

3.3.1 devices/build

The *build* structure holds the assembly components for the devices – such as the build scripts and the raw base OS images –

Directories	Description
images	A repository for the raw (original) OS images, prior to device customisation.
scripts	Scripts to image/configure/harden device builds, from image to deployed boot and root.

Note that the delimiter s the space character.

Example

Filing the Ubuntu 20.04 LTS Desktop image:

```
devices/build/images/ubuntu-20.04-desktop-amd64.iso
```

3.3.2 devices/device

The nearest standard for this structure is the PXELinux configuration file mapping approach[3], which will be adopted.

Specification

Local site preferences will dictate which of the following is most applicable *devicen* naming convention –

Directories	Description
UUID	The Unique ID for the device itself (regardless of its address or location)
MAC Address	The MAC Address of the ethernet adapter that has been used to connect the device to the network (which would include the virtual NIC for Virtual Machines)
Hostname	The hostname of the device (assumed to be unique within the context of the local network / management domain)
FQDN	The hostname of the device, as represented by the fully-qualified domain name[4], which is assured to be globally unique.
IPv4 Address	The hex-encoded IPv4 Address of the device
IPv6 Address	The hex-encoded IPv6 Address of the device

Note that the delimiter is the period character for hostname types and the dash character for non-hostname types.

Under the device, the sub-directory hierarchy is designed to provide operational support to the platform –

Directories	Description
backups	Storage for device backups (whether snapshot, full backup, incremental backup, etc)
boot	Pre-boot images (kernel/firmware) and configuration files for this device
data	Storage for data (including both database and unstructured data)
packages	Primarily designed to store packages required to automate a build to a known-good state
patches	Primarily designed to store patches required to automate a build to a known-good state
root	The root file-system for this device, either as stored images or inflated / raw.

3 https://wiki.syslinux.org/wiki/index.php?title=PXELINUX#Configuration_filename
4 https://en.wikipedia.org/wiki/Fully_qualified_domain_name

Directories	Description
scripts	Scripts for build, run and backup of the device, sufficient to support stateless instances

Note that the delimiter s the space character.

Script

The following script has been authored to automate creation of the above skeleton directory hierarchy.

```
#!/bin/bash


if [ ${#} -ne 1 ]
then
  echo "Usage: ${0} <path to project>"
  exit 1
fi

tld="${1}"
if [ ! -d "${tld}" ]
then
  echo "Error: ${0} path \"${tld}\" is not a directory."
  exit 1
fi

(
cat << EOF
  backups
  boot
  data
  packages
  patches
  root
  scripts
EOF
) | while read deep_dir
do
  mkdir -p "${tld}/${deep_dir}"
  if [ ! -d "${tld}/${deep_dir}" ]
  then
    echo "Error: failed to create path \"${deep_dir}\"."
    exit 1
  fi
  chmod 755 "${tld}/${deep_dir}"
done
```

3.4 financial

The *financial* hierarchy is structured for filing what would otherwise be mostly paper artefacts.

As a top level structure, financial is assumed to be common (group) accessed – such as via an organisation's finance department. Where users have their own private financial papers, this structure can be leveraged in the *personal* hierarchy.

```
financial
├── portfolio
│   ├── cash and currencies
│   ├── commodities
│   ├── other securities
│   └── stocks and bonds
├── budgets
├── expenses
│   ├── bills
│   ├── insurance
│   ├── leases
│   └── receipts
├── revenue
│   ├── recurring
│   ├── service
│   └── transaction
└── tax
```

As this pertains to local data there is no public metadata mapping available for handles/labelling.

Script

The following script has been authored to automate creation of the above skeleton directory hierarchy.

```
#!/bin/bash


if [ ${#} -ne 1 ]
then
  echo "Usage: ${0} <path to multimedia>"
  exit 1
fi

tld="${1}"
if [ ! -d "${tld}" ]
then
  echo "Error: ${0} path \"${tld}\" is not a directory."
  exit 1
fi

(
cat << EOF
  portfolio/cash and currencies
  portfolio/commodities
  portfolio/other securities
  portfolio/stocks and bonds
  budgets
  expenses/bills
  expenses/insurance
  expenses/leases
  expenses/receipts
  revenue/recurring
  revenue/service
  revenue/transaction
  tax
EOF
) | while read deep_dir
do
  mkdir -p "${tld}/${deep_dir}"
  if [ ! -d "${tld}/${deep_dir}" ]
  then
    echo "Error: failed to create path \"${deep_dir}\"."
    exit 1
  fi
  chmod 755 "${tld}/${deep_dir}"
done
```

3.4.1 financial/portfolio

The *portfolio* structure provides a basic repository for account/asset information – account details, property details, periodic statements, etc. To store any artefact required to understand or operate the portfolio[5].

Specification

The sub-directory hierarchy is designed to enable rapid filing/recovery of account and asset records. Within that portfolio directory, the structure is spit amongst publicly and non-publicly traded securities:

Directories	Description
cash and currencies	A sub-directory for all artefacts regarding bank accounts, currency holdings or cash equivalents (bitcoin, etc).
commodities	A sub-directory for all artefacts pertaining to publicly traded commodities (soft and hard).
other securities	A sub-directory for all artefacts regarding non-publicly traded securities like real estate, art, classic cars and other private investments.
stocks and bonds	A sub-directory for all artefacts pertaining to publicly traded securities, including equity and/or creditor stakes.

Note that the delimiter is the space character.

At volume, each sub-directory would typically include further structure for the corresponding holding / source.

Example

For a basic example, filing Bruce Wayne's credit card account statement:

```
financial/portfolio/cash and currencies/AmEx - Gotham National Bank/200807 Statement.pdf
```

3.4.2 financial/budgets

The *budgets* structure retains the short term financial planning material – typically annually.

Specification

For a house-hold the budget may be an evolving, but otherwise static document. Even in the case of versioned budget files, dating these will provide sufficient clarity. So in the case of a house-hold deployment it is recommended that the budget directory not be further structured.

For a business there is likely to be a budget per financial period; in which case it is recommended that the budgets directory be structured further:

Directories	Description
Time Period	The relevant financial reporting period – for financial year 2008 the period would be "2008", while for the second quarter in financial year 2008, the period would be "2008Q2".

Note that the delimiter is the space character.

5 https://www.investopedia.com/terms/p/portfolio.asp

Example

For a personal example, the Gordon family budget might be:

```
financial/budgets/budget.xls
```

For a business example, the quarterly budget for Applied Sciences at Wayne Enterprises, would be named and filed as follows:

```
financial/budgets/2008Q2/Applied Sciences.xls
```

3.4.3 financial/expenses

The *expenses* structure provides a basic repository for expenditure information – bills/invoices, insurance, leases, and a structure for filing receipts.

Specification

A business anticipating a substantial volume of expenses should consider additional classification, for example:

Directories	Description
Various sub-directories	When grouping by expense type, the use of deductible expenses[6] may be the most effective. The following examples could be used as sub-directory values (in lowercase), noting that these will vary by business type and jurisdiction: Advertising (or Marketing), Credit Card Processing, Education (or Training), Legal Fees, License (or Regulatory) Fees, Contractor Wages, Employee Benefits Programs, Equipment Rentals, Insurance, Interest Paid, Office Supplies, Maintenance and Repair, Office Lease, Travel, Utility Expenses, etc.

However, for the house-hold, the structure is intended to be reasonably light-weight – providing directories to help manage document flow:

Directories	Description
bills	A sub-directory for all billed (or invoiced) expenses that are not otherwise classified (i.e. neither insurance, nor leases). This would include periodic bills (utilities for example) and adhoc bills (such as call-out plumbing).
insurance	A sub-directory for all insurance expenses. These are distinct as the invoice and its payment become evidence of a service that may be relied upon.
leases	A sub-directory for all leasing expenses. These would include premises, vehicles, and even cell phone plans. These are distinct as the invoice and its payment become evidence of continued ownership of an asset.
receipts	A sub-directory for all receipts – whether received at Point of Sale (i.e. purchasing via credit card online) or at payment of a bill/invoice. This structure provides a rapid look-up ability, for validating expenditure (an unknown account transaction) or confirming payment (that invoice was paid on time).

Note that the delimiter is the space character.

The above is the most basic template. Consider expanding the above to suit specific situations (if there are routine medical bills, or if splitting utility bills, recurring donations, etc).

At volume, each sub-directory would typically include further structure for the corresponding product or service and its source.

6 https://www.investopedia.com/terms/b/businessexpenses.asp

Example

For a bills example, filing Bruce Wayne’s Bat Mask bill – which could be labelled for the product or the maker, whichever is the most readily recognised:

```
financial/expenses/bills/2005/20050615 – Bat Masks.pdf
```

For a receipts example, filing Bruce Wayne’s Bat Mask payment receipt – labelled per the bill:

```
financial/expenses/receipts/2005/20050621 – Bat Masks.pdf
```

For an insurance example, filing the renewal notice for Bruce Wayne’s Lamborghini:

```
financial/insurance/Murciélago LP 640/2005/Gotham Car Lovers – Comprehensive Renewal.pdf
```

3.4.4 financial/revenue

The *revenue* structure provides a basic repository for earnings information – account details, property details, periodic statements, etc. To store any artefact required to understand or operate the portfolio[7].

Specification

A business anticipating a broad range of revenue types should consider additional classification, for example:

Directories	Description
Various sub-directories	When grouping by revenue type, the alignment to business verticals / taxation classes may be the most effective. The following examples could be used as sub-directory values (in lowercase), noting that these will vary by business type and jurisdiction: Advertising, Arbitrage, Commission (or Brokerage), Dividend, Donation, Fee-for-Service (Sale of Goods), Interest, Lease (or Rent), Licensing. Subscription

However, for the house-hold, the structure is intended to be reasonably light-weight – providing a revenue stream[8] based directory structure to help manage document flow:

Directories	Description
recurring	A sub-directory for all artefacts regarding ongoing revenue, from subscriptions, leases (rentals), interest, brokerage or advertising fees.
service	A sub-directory for all artefacts pertaining to revenue calculated through time based effort, including a salary, consulting, projects and odd-jobs.
transaction	A sub-directory for all artefacts regarding the sale of goods that are typically one-time customer payments, including the sale of an asset, or the proceeds from a garage sale or even a lemonade stand.

Note that the delimiter is the space character.

At volume, each sub-directory would typically include further structure for the corresponding holding / source.

Example

For a full-time income example, filing Luscious Fox’s fortnightly pay slips:

```
financial/revenue/service/Wayne Enterprises/20080714 Payslip.pdf
```

For a project example, filing the invoice for Luscious Fox’s antidote project:

```
financial/revenue/service/Batman/20080721 Antidote Project Invocie.pdf
```

7 https://www.investopedia.com/terms/p/portfolio.asp
8 https://corporatefinanceinstitute.com/resources/knowledge/accounting/revenue-streams/

3.4.5 financial/tax

The *tax* structure is intended to provide the basic skeleton to underpin a repository for all materials related to taxation.

Specification

This specification has been designed with either a personal or a small/medium business in mind. The sub-directory hierarchy is designed to enable rapid filing/recovery of taxation records. The naming convention is the time period combined with the taxation type being reported/lodged, where:

Directories	Description
Time Period	The relevant financial reporting period – for financial year 2008 the period would be “2008”, while for the second quarter in financial year 2008, the period would be “2008Q2”.
Taxation Type	The taxation type that is being reported, i.e: Annual Return, Quarterly Statement, etc.

Note that the delimiter is the space character.

Where this repository is being used in a larger organisation the structure can be further sub-divided to entities; one example may be a family with multiple personal annual returns, requiring a sub-directory per person.

Within that taxation type sub-directory, the structure is spit amongst inputs (sources) and outputs (submission):

Directories	Description
sources	A sub-directory for all artefacts used to develop and/or support/substantiate the submission.
submission	A sub-directory for all artefacts comprising, transacting, and/or acknowledging/receipting the submission.

Note that the delimiter is the space character.

Example

For a personal example, as a source document, filing the personal income statement (Pay As You Go statement, or Group Certificate) for time worked at Wayne Enterprises during financial year 2008:

```
financial/tax/2008 Annual Return/sources/Personal – Income – Wayne Enterprises PAYG.pdf
```

For a group/family example, as a submission document, filing the personal income tax return for Luscious Fox during financial year 2008:

```
financial/tax/2008 Annual Return/Fox, Luscious/submission/Income Tax Return - Signed.pdf
```

For a business example, the quarterly business income statement (Business Activity Statement) for revenue earned in the second quarter by Wayne Enterprises would be named and filed as follows:

```
financial/tax/2008Q2 Quarterly Statement/submission/Activity Statement – Signed.pdf
```

3.5 legal

The *legal* hierarchy is structured for filing what would otherwise be mostly paper artefacts.

As a top level structure, legal is assumed to be common (group) accessed – such as via an organisation's legal department. Where users have their own private legal papers, this structure can be reused in the *personal* hierarchy.

```
legal
├── lawyers
└── matters
```

As this pertains to local data there is no public metadata mapping available for handles/labelling.

3.5.1 legal/lawyers

The *lawyers* structure is intended to provide the basic skeleton for all materials related to a given counselling entity. In the case of personal or business repository each entity would be a firm. In the case of a legal practice this repository may be billable partners, attorneys, barristers, solicitors, etc.

Specification

The sub-directory hierarchy for "usern" is a directory per *user* or *device*. The naming convention will be locally relevant, with observed examples including –

Directories	Description
Identifier	The invoicing/billing identifier or Bar registration number of the entity, such as: 000039
Firm Name	The name of the legal firm: Lee, Peck and Associates
Attorney Name	Full personal name, where examples include: "Finch, Atticus", "Atticus.Finch", "Atticus Finch"

Note that the delimiter varies based on the scheme chosen.

It is recommended that the naming convention selected is one that can be most automated within the target environment, with the least additional administrative overhead.

3.5.2 legal/matters

The *matters* structure is intended to provide the basic skeleton for document-based matter management[9] - a repository for all materials related to a given matter.

Specification

Unless anticipating a substantial volumes of matters, the structure is intended to be flat (matters listed without further classification). However, at volume, additional classification could be incorporated by matter type to create additional grouping.

Directories	Description
Various sub-directories	To create grouping by matter type[10] the following example legal areas could be used as sub-directory values (in lowercase), noting that in a commercial practice these could be re-grouped to partner domains, or market/advertised verticals: Administrative, Admiralty, Adoption, Agency, Alcohol, Alternative dispute resolution, Animal, Antitrust (or Competition), Art (or Art and Culture), Aviation,

9 https://en.wikipedia.org/wiki/Legal_matter_management
10 https://en.wikipedia.org/wiki/List_of_areas_of_law

Directories	Description
	Banking, Bankruptcy (Creditor Debtor Rights or Insolvency and Reorganisation), Bioethics, Business (or Commercial); Commercial Litigation, Business Organisations (or Companies), Canon, Civil or Common, Class Action Litigation or Mass Tort Litigation, Communications, Computer, Competition, Conflict of Law (or Private International), Constitutional, Construction, Consumer, Contract, Copyright, Corporate (or Company), Criminal, Cryptography, Cultural Property, Custom, Cyber, Defamation, Drug Control, Elder, Employment, Energy, Entertainment, Environmental, Family, Financial Services, Firearm, Food, Gaming, Health, Health and Safety, Housing, Immigration, Insurance, Intellectual property, International, International Human Rights, International Humanitarian, International Trade and Finance, Internet, Juvenile, Labour (or Labor), Landlord-Tenant, Litigation, Martial, Media, Medical, Military, Mining, Mortgage, Music, Nationality, Obscenity, Parliamentary, Patent, Poverty, Privacy, Procedural, Property, Public Health, Public International Law, Real Estate, Securities / Capital Markets, Space, Sports, Statutory, Tax, Technology, Tort, Trademark, Transport / Transportation, Trusts and Estates, Water.

Whether flat or categorised, per the above, the remaining sub-directory hierarchy is simply defined in order to reflect a list of distinct matters. The naming convention will be a local preference, with observed examples including –

Directories	Description
Serial Number	An increasing matter number, for example: 008264 Note for business legal services, this could equally be a Billing Number.
Date	Calendar Year & Month & Day of initiation the matter, such as: 18030211
Matter Type	The area of law that frames the matter: Constitutional
Matter Name	Full name of the matter: Marbury v. Madison
Serial Number and Matter Name	The recommended business implementation: The serial or billing number, with the full event name: 008264 – Marbury v. Madison
Date and Matter Name	The recommended personal implementation: The date, with the full matter name: 18030211 – Marbury v. Madison Or, the date to month, with full matter name: 180302 – Marbury v. Madison
Date, Matter Type and Matter Name	The recommended small/medium business implementation: The date (to the day, or the month), with the matter type and full matter name: 180302 – Constitutional, Marbury v. Madison

Note that the delimiter is the space character.

Example

For a personal example, the matter of Marbury v. Madison[11] could be named and filed as follows:

```
legal/matters/180302 – Marbury v. Madison
```

For a business example, the matter of Marbury v. Madison would be named and filed as follows:

```
legal/matters/180302 – Constitutional, Marbury v. Madison
```

For a counsel example, the matter of Marbury v. Madison would be named and filed as follows:

```
legal/matters/constitutional/008264 - Marbury v. Madison
```

11 https://en.wikipedia.org/wiki/Marbury_v._Madison

3.6 multimedia

The *multimedia* hierarchy is structured for both simple filing, and to maximise metadata source to high-level path mapping. Some depth has been provided to allow for infrastructure alignment by technical capability.

As a top level structure, multimedia is assumed to be common (group) accessed – via devices in communal environments rather than by individual users. Where users have their own private multimedia, this structure can be reused in the *personal* hierarchy.

```
multimedia
├── books
├── dph
├── games
│   └── roms
├── images
│   ├── backgrounds
│   ├── photos
│   └── pictures
├── movies
│   ├── 3d
│   ├── 4k
│   ├── 8k
│   ├── hd
│   ├── iso
│   └── sd
├── music
│   ├── audio
│   └── video
└── tv
    ├── education
    │   ├── applied science
    │   ├── arts and recreation
    │   ├── general
    │   ├── geography and history
    │   ├── language
    │   ├── literature
    │   ├── natural science
    │   ├── philosophy and psychology
    │   ├── religion
    │   └── social science
    ├── exercise
    │   ├── endurance
    │   ├── flexibility
    │   └── strength
    └── series
```

Naming conventions will be content specific in order to maximise the principle of storing handles.

Script

The following script has been authored to automate creation of the above skeleton directory hierarchy.

```
#!/bin/bash


if [ ${#} -ne 1 ]
then
  echo "Usage: ${0} <path to multimedia>"
  exit 1
fi

tld="${1}"
if [ ! -d "${tld}" ]
then
  echo "Error: ${0} path \"${tld}\" is not a directory."
  exit 1
fi

(
cat << EOF
  books
  dph
  games/roms
  images/backgrounds
  images/photos
  images/pictures
  movies/3d
  movies/4k
  movies/8k
  movies/hd
  movies/iso
  movies/sd
  music/audio
  music/video
  tv/education/applied science
  tv/education/arts and recreation
  tv/education/general
  tv/education/geography and history
  tv/education/language
  tv/education/literature
  tv/education/natural science
  tv/education/philosophy and psychology
  tv/education/religion
  tv/education/social science
  tv/exercise/endurance
  tv/exercise/flexibility
  tv/exercise/strength
  tv/series
EOF
) | while read deep_dir
do
  mkdir -p "${tld}/${deep_dir}"
  if [ ! -d "${tld}/${deep_dir}" ]
  then
    echo "Error: failed to create path \"${deep_dir}\"."
    exit 1
  fi
  chmod 755 "${tld}/${deep_dir}"
done
```

3.6.1 multimedia/books

The books hierarchy is under review and is yet to be fully defined.

3.6.2 multimedia/dph

This hierarchy has been structured around the concept of the Digital Photo Hub[12]. All multimedia (photos, videos, etc) that have been captured for a set of events can be stored in this structure – creating a central and structured location for locally created visual media.

Specification

A sub-directory hierarchy is simply defined in order to reflect the most common use case – photography, and video recordings of a particular event – holiday, sports match, birthday, etc. The naming convention will be a local preference, with observed examples including –

Directories	Description
Serial Number	An increasing event number, for example: 000682 Note for business photography, this could equally be a Billing Number.
Date	Calendar Year & Month & Day of initiation the event, such as: 20191231
Event Name	Full name of the event: NYE, Welcome to 2020
Serial Number	The recommended business implementation:

12 https://zapier.com/blog/photographs-workflow/

Directories	Description
and Event Name	The serial or billing number, with the full event name: 000682 – NYE, Welcome to 2020
Date and Event Name	The recommended personal implementation: The date, with the full event name: 20191231 – NYE, Welcome to 2020 Or, the date to month, with full event name: 201912 – Christmas at Whistler

Note that the delimiter is the space character.

Within that event directory structure, images and videos can be organised to preference. Give some thought to the outputs for an Event; for example if 5% of those images are going to be printed for Grandma, and 10% are going to be published to Facebook, then perhaps additional “grandma” and “facebook” sub-directories might be useful within the given event directory.

3.6.3 multimedia/games

At this time the structure only hosts roms, though it is designed to be expanded in the future.

Example sub-structures might potentially include iso, win32, win64, etc.

3.6.3.1 multimedia/games/roms

To preserve our digital heritage, in 2013 [13], the Internet Archive released[14] a collection of ROM images into the Public Domain, where each of those ROM images could be played with software emulators.

Specification

Those software emulators expect the ROM images to be in a *platform* based hierarchy in order to intuit the appropriate emulation software and ROM metadata.

Directories	Description
Various sub-directories	Per the supported platforms in the available configuration [15] for the representative emulation software, the included are the only known roms/ sub-directory values: 3do, ags, amiga, amstradcpc, apple2, arcade, arcadia, atari2600, atari5200, atari7800, atari800, atarijaguar, atarilynx, atarist, c64, channelf, coco, coleco, crvision, daphne, dragon32, dreamcast, fba, fds, fm7, gameandwatch, gamegear, gb, gba, gbc, gc, intellivision, love, macintosh, mame-advmame, mame-libretro, mame-mame4all, mastersystem, megadrive, moto, msx, n64, nds, neogeo, nes, ngp, ngpc, np2pi, oric, pc, pc88, pc98, pcengine, pcfx, pokemini, ports, ps2, psp, psx, samcoupe, saturn, scummvm, sega32x, segacd, sg-1000, snes, solarus, ti99, trs-80, vectrex, videopac, virtualboy, wii, wonderswan, wonderswancolor, x1, x68000, zmachine, zx81, zxspectrum

Within those sub-directories, each ROM is stored as a single zip file. The recommended naming convention, in order to support the handle principle, is follows.

Field	Description	Values
Full Name	The full name of the game as it was released, optionally in a pre-sorted structure.	Examples: • “The Three Stooges” • “Three Stooges, The”
Region	The comma separated regions	Examples:

13 https://games.slashdot.org/story/13/12/28/0351206/archiveorg-hosts-massive-collection-of-mame-roms
14 https://archive.org/details/MAME_0.151_ROMs
15 https://github.com/RetroPie/RetroPie-Setup/blob/master/platforms.cfg

Field	Description	Values
	of the game as it was released, in brackets.	• “(USA)” • “(USA, Europe)”
Version	Bracketed version-specific information.	Examples: • “(Beta)” • “(Enhanced)”
Extension	The three letter file extension	As all ROMs are zip files the extension will be “zip”

Note that the delimiter is a space character, except for the period delimiting the file extension.

Example

For a simple example, the Archived[16] game “3stooges.zip” would be named and filed as follows:

```
multimedia/games/roms/nes/Three Stooges, The (USA) (Beta).zip
```

3.6.4 multimedia/images

The images hierarchy is under review and is yet to be fully defined.

Specification

A sub-directory hierarchy has been defined to allow data mapping based on typical use cases.

Directories	Description
backgrounds	Content that is intended for use as computer desktop backgrounds
photos	Content that has been captured from camera/s, of the physical environment.
pictures	Other content, such as rendered, scanned or otherwise synthetically generated.

3.6.5 multimedia/movies

There are a number of existing standards [17] [18] [19] for the naming of movies. These will be used as the basis of this specification, but additional detail is recommended for file-system based (rather than library based) access.

Specification

A sub-directory hierarchy has been defined to allow data mapping based on device capability.

Directories	Description
8k	8K Ultra High Definition (8KUHD) – any content that is 2161 to 4320 pixels high
4k	4K Ultra High Definition (4KUHD) – any content that is 1081 to 2160 pixels high
hd	Full High Definition (FHD) – any content that is 577 to 1080 pixels high
sd	Standard Definition (SD) – any content up to 576 pixels high
iso	Any content stored as an ISO/IEC 13346 file-system (UDF[20])
3d	Three Dimensional – Content that includes depth rendering information

16 https://archive.org/download/MAME_0.151_ROMs/MAME_0.151_ROMs.zip/MAME%200.151%20ROMs%2F3stooges.zip
17 https://support.plex.tv/articles/naming-and-organizing-your-movie-media-files/
18 https://support.emby.media/support/solutions/articles/44001159102-movie-naming
19 https://kodi.wiki/view/Naming_video_files/Movies
20 https://en.wikipedia.org/wiki/Universal_Disk_Format

Working from these standards and adding some usability at the file system level, the recommended convention is as follows.

Field	Description	Values
Full Name	The full name of the movie as it was released, where spaces are replaced with the delimiter.	Examples: • "Sintel" • "Big.Buck.Bunny"
Release Year	The year that the movie was released, in brackets.	Four-digit year. e.g. "(2008)"
Video Format	The display height and render flag	Valid display heights are 480, 576, 720, 1080, 2160, 4320. The render flags are either "i" for interlaced, or "p" for progressive. e.g. "1080p"
3D Format	If 3D then "3D" and the two-eye encoding method.	Either horizontal (side-by-side) or vertical (top-and-bottom) encoding. e.g. "3D.H-SBS", "3D.V-TAB"
Source	The origin of the file content	Valid sources are: • BluRay – raw disc content • BRRip – transcoded disc content • HDDVD – raw disc content • HDRip – transcoded disc content • HDTV – raw TV content • Web – raw web downloaded content • WebRip – transcoded web downloaded content
Audio Encoding	The audio encoding / protocol	Valid encoding schemes are AAC, AC3, DTS, DTS-HDMA, EAC3, MP2, MP3, TrueHD.
Audio Channels	The number of audio channels	Valid numbers are 1 (mono) to 9 (spatial), and presented as "NCHn". e.g. 5.1 would be "NCH6"
Video Encoding	The video encoding / protocol	Valid encoding schemes are MP2, MP4, x264, x265, XviD
Stacking Value	If multi-part, then note this part	If this file is only part of the same movie then denote the part type and part number. Valid stacking terms are cd, disc, disk, dvd, part, pt. Valid part numbers are 1 – 9. e.g. "disc2"
Extension	The three letter file extension	Typical file extensions are: avi, iso, mkv, mpg, mp4, wmv Note that the extension is required to be an accurate representative handle for the media container [21] [22]

Note that the delimiter is the period character.

Example

For a simple example, the movie Sintel in 1080p[23] would be named and filed as follows:

```
multimedia/movies/hd/Sintel.(2010).1080p.Web.AC3.NCH6.x264.mkv
```

For an example where the name of the movie also contains spaces, the movie Big Buck Bunny in 1080p[24] would be as follows:

```
multimedia/movies/hd/Big.Buck.Bunny.(2008).1080p.Web.MP3.NCH2.x264.mp4
```

21 https://en.wikipedia.org/wiki/Comparison_of_video_container_formats
22 https://www.encoding.com/blog/2014/01/13/whats-difference-codecs-containers/
23 https://durian.blender.org/ | https://download.blender.org/durian/movies/Sintel.2010.1080p.mkv
24 https://peach.blender.org/ | http://distribution.bbb3d.renderfarming.net/video/mp4/bbb_sunflower_1080p_30fps_normal.mp4

For a lower definition example, the movie Elephants Dream in 640x360 [25] would be named and filed as follows:

```
multimedia/movies/sd/Elephants.Dream.(2006).480p.Web.AAC.NCH2.x264.mp4
```

For a 3D example, the movie Big Buck Bunny in HD Stereoscopic 3D [26] would be named and filed as follows:

```
multimedia/movies/3d/Big.Buck.Bunny.(2008).1080p.3D.V-TAB.Web.MP3.NCH2.x264.mp4
```

Script

The following script has been authored to help automate file naming consistently with this specification.

```
#!/bin/bash


function use_mplayer {
  # mplayer -vo null -ao null -identify -frames 0 [file]
  # unable to detect interlacing
  video_interlace="p"
  "${1}" -vo null -ao null -identify -frames 0 "${2}" 2>/dev/null |
  {
    while read ENTRY
    do
      # echo $ENTRY
      if [ -z "${video_format}" -a "${ENTRY:0:15}" == "ID_VIDEO_FORMAT" ]
      then
        case "${ENTRY:16:32}" in
          "1x0000002")
            video_format="MP2" ;;
          "DX50")
            video_format="MP4" ;;
          "H264")
            video_format="x264" ;;
          "HEVC")
            video_format="x265" ;;
          "VC-1")
            video_format="x264" ;;
          "WVC1")
            video_format="x264" ;;
          "XVID")
            video_format="XviD" ;;
          *)
            video_format="unknown"
            ;;
        esac
      fi
      if [ -z "${video_height}" -a "${ENTRY:0:15}" == "ID_VIDEO_HEIGHT" ]
      then
        source_height=$(( ${ENTRY:16:32} + 0 ))
        if [ ${source_height} -gt 2160 ]
        then
          video_height="4320"
          video_path="8k"
        elif [ ${source_height} -gt 1080 ]
        then
          video_height="2160"
          video_path="4k"
        elif [ ${source_height} -gt 720 ]
        then
          video_height="1080"
          video_path="hd"
        elif [ ${source_height} -gt 576 ]
        then
          video_height="720"
          video_path="hd"
        elif [ ${source_height} -gt 480 ]
        then
          video_height="576"
          video_path="sd"
```

25 https://orange.blender.org/ | https://archive.org/download/ElephantsDream/ed_hd.mp4
26 http://distribution.bbb3d.renderfarming.net/video/mp4/bbb_sunflower_1080p_30fps_stereo_abl.mp4

```
        else
          video_height="480"
          video_path="sd"
        fi
      fi
      if [ -z "${audio_format}" -a "${ENTRY:0:15}" == "ID_AUDIO_FORMAT" ]
      then
        case "${ENTRY:16:32}" in
          "MP4A")
            audio_format="AAC" ;;
          "TRHD")
            audio_format="TrueHD" ;;
          "EAC3")
            audio_format="EAC3" ;;
          "354")
            audio_format="WMA" ;;
          "8192")
            audio_format="AC3" ;;
          "8193")
            audio_format="DTS" ;;
          "80")
            audio_format="MP2" ;;
          "85")
            audio_format="MP3" ;;
          *)
            audio_format="unknown" ;;
        esac
      fi
      if [ -z "${audio_channels}" -a "${ENTRY:0:12}" == "ID_AUDIO_NCH" ]
      then
        case "${ENTRY:13:32}" in
          "9")
            audio_channels=".NCH9" ;;
          "8")
            audio_channels=".NCH8" ;;
          "7")
            audio_channels=".NCH7" ;;
          "6")
            audio_channels=".NCH6" ;;
          "5")
            audio_channels=".NCH5" ;;
          "4")
            audio_channels=".NCH4" ;;
          "3")
            audio_channels=".NCH3" ;;
          "2")
            audio_channels=".NCH2" ;;
          "1")
            audio_channels=".NCH1" ;;
          "0")
            audio_channels="" ;;
          *)
            audio_channels="" ;;
        esac
      fi
    done
    echo "${video_path}/${video_height}${video_interlace}.BluRay.${audio_format}$
{audio_channels}.${video_format}"
  }

  return 0
}

function use_ffprobe {
  # ffprobe -show_streams -select_streams v:0 [file]
  # ffprobe -show_streams -select_streams a:0 [file]
  video_interlace="p"
  "${1}" -show_streams -select_streams v:0 "${2}" 2>/dev/null |
  {
    while read ENTRY
    do
      # echo $ENTRY
      if [ -z "${video_format}" -a "${ENTRY:0:10}" == "codec_name" ]
      then
        case "${ENTRY:11:32}" in
          "mpeg2video")
            video_format="MP2" ;;
          "dx50")
            video_format="MP4" ;;
          "h264")
            video_format="x264" ;;
          "hevc")
            video_format="x265" ;;
          "vc1")
            video_format="x264" ;;
          "xvid")
            video_format="XviD" ;;
          *)
            video_format="unknown"
            ;;
        esac
      fi
      if [ -z "${video_height}" -a "${ENTRY:0:6}" == "height" ]
      then
        source_height=$(( ${ENTRY:7:32} + 0 ))
        if [ ${source_height} -gt 2160 ]
        then
          video_height="4320"
          video_path="8k"
        elif [ ${source_height} -gt 1080 ]
        then
          video_height="2160"
          video_path="4k"
        elif [ ${source_height} -gt 720 ]
        then
          video_height="1080"
          video_path="hd"
        elif [ ${source_height} -gt 576 ]
        then
          video_height="720"
          video_path="hd"
        elif [ ${source_height} -gt 480 ]
        then
          video_height="576"
          video_path="sd"
        else
          video_height="480"
          video_path="sd"
        fi
      fi
    done
    "${1}" -show_streams -select_streams a:0 "${2}" 2>/dev/null |
    {
      while read ENTRY
      do
        if [ -z "${audio_format}" -a "${ENTRY:0:10}" == "codec_name" ]
        then
          case "${ENTRY:11:32}" in
            "ac3")
              audio_format="AC3" ;;
            "eac3")
              audio_format="EAC3" ;;
            "dca")
              audio_format="DTS" ;;
            "mp2")
              audio_format="MP2" ;;
            "mp3")
              audio_format="MP3" ;;
            "mp4")
              audio_format="AAC" ;;
            "truehd")
              audio_format="TrueHD" ;;
            "wmapro")
              audio_format="WMA" ;;
            *)
              audio_format="unknown" ;;
          esac
        fi
        if [ -z "${audio_channels}" -a "${ENTRY:0:8}" == "channels" ]
        then
          case "${ENTRY:9:32}" in
            "9")
              audio_channels=".NCH9" ;;
            "8")
              audio_channels=".NCH8" ;;
            "7")
              audio_channels=".NCH7" ;;
            "6")
              audio_channels=".NCH6" ;;
            "5")
              audio_channels=".NCH5" ;;
            "4")
              audio_channels=".NCH4" ;;
            "3")
              audio_channels=".NCH3" ;;
            "2")
              audio_channels=".NCH2" ;;
            "1")
              audio_channels=".NCH1" ;;
            "0")
              audio_channels="" ;;
            *)
              audio_channels="" ;;
          esac
        fi
      done
      echo "${video_path}/${video_height}${video_interlace}.BluRay.${audio_format}$
{audio_channels}.${video_format}"
    }
  }

  return 0
}


##
## MAIN
##
if [ ${#} -ne 1 ]
then
  echo "NFO [main] Try \"${0} <path/file>\"."
  exit 1
fi

tld="${1}"
if [ ! -f "${tld}" ]
then
  echo "Error: Unable to find file \"${tld}\"."
  exit 1
fi

binary_path=`which mplayer`
if [ "${?}x" == "0x" ]
then
  use_mplayer "${binary_path}" "${tld}"
  exit 0
fi

binary_path=`which ffprobe`
if [ "${?}x" == "0x" ]
then
  use_ffprobe "${binary_path}" "${tld}"
  exit 0
fi

echo "Error: Unable to find video interrogation tool in current search path."
```

```
exit 1
```

Note that this script is only able to identify/classify sd, hd, 4k and 8k videos.

Note also that both the mplayer and ffmpeg versions are using the first video and audio stream to determine the content quality, which may not be valid for all media files.

3.6.6 multimedia/music

The music hierarchy is under review and is yet to be fully defined.

Specification

A sub-directory hierarchy has been defined to allow data mapping based on device capability.

Directories	Description
audio	Content that only has sound, and no visual track – stored in audio-only file types
video	Content that is both audible and visual – stored in video file types

The impending structure within the above directories, is likely to two-fold, depending on the track context:

- artists/artist name/album name/track files
- compilations/album name/track files

3.6.7 multimedia/tv

The notion of what is and isn't "Television" is being blurred by Internet accessible content, such as Youtube. This structure is therefore defined as "episodic" content, as opposed to movies which are works of one entire but singular narrative each.

Specification

A sub-directory hierarchy has been defined to allow better metadata alignment.

Directories	Description
education	Content to facilitate learning – the acquisition of knowledge, skills, values and beliefs
exercise	Content with instruction for the performance of physical activity – for the purpose of enhancing or maintaining physical health and well-being
series	Content for entertainment – each subset being a narrative that is divided into seasons / series of component episodes

3.6.7.1 multimedia/tv/education

The education structure exists to capture all types of learning material. As this material is typically multi-sourced and as there is no universal meta-data repository for this type of content, it is typically stored as "unstructured data". Hence, this specification needs to classify the content at the file-system layer, despite the second principle.

The problem with creating a taxonomy for the reference/storage of all human knowledge is that it is notoriously difficult[27]. Despite some oddities, the recommended structure is the Dewey Decimal Classification system (in its 23rd edition[28], since the paper edition[29] was first published in 1873).

27 http://emo.world/2018/05/01/on-creating-a-taxonomy-of-all-human-knowledge-for-haha-academy/
28 https://www.oclc.org/content/dam/oclc/dewey/ddc23-summaries.pdf
29 https://archive.org/details/decimal19v1dewe

If you’re not familiar with it, the Dewey Decimal Classification organises content by discipline. The primary / top level structure includes (for example) philosophy, social sciences, natural sciences, applied sciences and history.

The scheme comprises ten classes, each divided into ten divisions, each having ten sections. The system's notation uses Arabic numbers, with three whole numbers making up the main classes, sub-classes and decimals designating further divisions. The classification structure is hierarchical and the notation follows the same hierarchy.

Where the full level of detail of the classification is not required, the right-most decimal digits can be trimmed from the class number to obtain more general classifications. For example:

```
500 Natural sciences and mathematics
└── 510 Mathematics
    └── 516 Geometry
        └── 516.3 Analytic geometries
            └── 516.37 Metric differential geometries
                └── 516.375 Finsler geometry
```

While this specification has no interest in the numbering scheme, the nesting of the DDC inherently lends itself to a directory based file-system structure, and can be made as deep or as shallow as either the volume of content, or depth of interest, demands.

Specification

This sub-directory hierarchy has been defined to allow for deep storage of otherwise unstructured media.

Directories	Description
general	*Information Management (000)* – Computer science, knowledge & systems; Bibliographies; Library & information sciences; Encyclopedias & books of facts; Magazines, journals & serials; Associations, organisations & museums; News media, journalism & publishing; Quotations; Manuscripts & rare books
philosophy and psychology	*Who am I? (100)* – Philosophy; Metaphysics; Epistemology; Parapsychology & occultism; Philosophical schools of thought; Psychology; Philosophical logic; Ethics; Ancient, medieval & eastern philosophy; Modern western philosophy
religion	*Who made me? (200)* – Religion; Philosophy & theory of religion; The Bible; Christianity; Christian practice & observance; Christian pastoral practice & religious orders; Christian organisation, social work & worship; History of Christianity; Christian denominations; Other religions
social science	*Who’s the guy in the next cave? (300)* – Social sciences, sociology & anthropology; Statistics; Political science; Economics; Law; Public administration & military science; Social problems & social services; Education; Commerce, communications & transportation; Customs, etiquette & folklore
language	*How do I talk to that guy? (400)* – Language; Linguistics; English & Old English languages; German & related languages; French & related languages; Italian, Romanian & related languages; Spanish, Portuguese, Galician; Latin & Italic languages; Classical & modern Greek languages; Other languages
natural science	*The world we can see (500)* – Science; Mathematics; Astronomy; Physics; Chemistry; Earth sciences & geology; Fossils & prehistoric life; Biology; Plants (Botany); Animals (Zoology)
applied science	*Making stuff out of what we can see (600)* – Technology; Medicine & health; Engineering; Agriculture; Home & family management; Management & public relations; Chemical engineering; Manufacturing; Manufacture for specific uses; Construction of buildings
arts and recreation	*What we do for fun (700)* – Arts; Area planning & landscape architecture; Architecture; Sculpture, ceramics & metalwork; Graphic arts & decorative arts; Painting; Printmaking & prints; Photography, computer art, film, video; Music; Sports, games & entertainment
literature	*Tell our children how wonderful we are (800)* – Literature, rhetoric & criticism; American

Directories	Description
	literature in English; English & Old English literatures; German & related literatures; French & related literatures; Italian, Romanian & related literatures; Spanish, Portuguese, Galician literatures; Latin & Italic literatures; Classical & modern Greek literatures; Other literatures
geography and history	*Tell future children how wonderful we were (900)* – History; Geography & travel; Biography & genealogy; History of ancient world (to ca. 499); History of Europe; History of Asia; History of Africa; History of North America; History of South America; History of other areas

Note that the delimiter is the space character.

Note that under the Dewey Decimal Classification system, computer science is not an *applied science*, it is an information system classified under *general*. By way of example, at the third tier of the DDC, under *Computer science, knowledge & systems* are: Computer science, information general works; Knowledge; The book; Systems; Computer science; Computer programming, programs, data, security; Special computer methods. For a sense of sanity, it is recommended that *computer science*, be created under *applied science*.

Additional classification detail can be found online[30].

Unlike other structures in the taxonomy, education is highly likely to have both specific end-point (deep domain) educational material (*network security*, for example) and cross discipline educational material (lecturers who span *mathematics*, *physics* and *chemistry*, for example). It is therefore recommended that cross discipline material be placed at the appropriate point in the directory tree, and not to attempt to create a deep structure that approximates or averages the content's educational domains.

Individual content stores will be named in various ways. The following are examples that have been seen before.

Directories	Description
Last, First	Where the content is acquired from a multiple sources, the directory can be named after an expert in the field, such as Feynman,.Richard and where further sub-directories group lecture series, i.e.: 1964 – The Character of Physical Law Example: Feynman, Richard/1964 – The Character of Physical Law/
Channel Name	Where the content is acquired from a single source, the directory can be named after the account-holder or channel from that source, such as KhanAcademy and where further sub-directories are play-lists, i.e. Developmental Maths 3 Example: Khan Academy/Developmental Maths 3/

Note that the delimiter is the space character.

3.6.7.2 multimedia/tv/exercise

The exercise structure exists to capture all types of physical training. The top level classification has been defined from the US NIH's National Institute on Aging[31] and the Wellness Helper[32].

Specification

A sub-directory hierarchy has been defined to allow for meaningful navigation, based on physical property.

Directories	Description
endurance	Aerobic activities including walking/jogging (including the treadmill), cycling, jumping rope, swimming, kickboxing, aerobic dancing, etc.
flexibility	Stretching and balancing activities including static (Pilates, Tai chi, Yoga etc), ballistic

30 https://en.wikipedia.org/wiki/List_of_Dewey_Decimal_classes
31 https://www.nia.nih.gov/health/four-types-exercise-can-improve-your-health-and-physical-ability
32 https://www.wellness-helper.com/types-of-exercise.html

Directories	Description
	(power training), and proprioceptive neuromuscular facilitation (PNF) exercises.
strength	Resistance activities including self/body weight, free and machine weights (including kettle bells), and resistance bands.

Note that the delimiter is the space character.

Individual content stores will be named in various ways. The following are examples that have been seen before.

Directories	Description
Last, First	Where the content is acquired from a multiple sources, the directory can be named after an expert in the field, such as Schwarzenegger, Arnold and where further sub-directories group exercise sets, i.e.: 1974 – Powerliftng Example: Schwarzenegger, Arnold/1974 – Powerliftng/
Channel Name	Where the content is acquired from a single source, the directory can be named after the account-holder or channel from that source, such as UK NHS and where further sub-directories are play-lists, i.e. Couch to 5K Example: UK NHS/Couch to 5K/

Note that the delimiter is the space character.

3.6.7.3 multimedia/tv/series

There are a number of existing standards [33] [34] [35] for the naming of television shows. These will be used as the basis of this specification.

Specification

The sub-directory hierarchy is a directory per *show* which is key to the metadata handle relationship.

Directories	Description
Full Name (Date)	The per-show sub-directory naming convention is full name and release date, for example, the TV show "The Beverly Hillbillies"[36] released in 1962: The Beverly Hillbillies (1962)

Note that the delimiter is the space character.

Within each sub-directory there are further sub-directories for the seasons/series;

Directories	Description
Season 00	The sub-directory *Season 00* is for "Special" episodes
Season nn	All other episodes are stored in the corresponding season/series number from Season 01 to Season 99

Note that the delimiter is the space character.

Working from the standards and adding some usability at the file system level, the recommended convention for episodes is as follows.

Field	Description	Values
Full Name	The full name of the show as it was released, where spaces are	For example: The.Beverly.Hillbillies

33 https://support.plex.tv/articles/naming-and-organizing-your-tv-show-files/
34 https://support.emby.media/support/solutions/articles/44001159110-tv-naming
35 https://kodi.wiki/view/Naming_video_files/TV_shows
36 https://en.wikipedia.org/wiki/The_Beverly_Hillbillies#Home_media_and_legal_status

Field	Description	Values
	replaced with the delimiter.	
Release Year	The year that the show was released, in brackets.	Four-digit year. e.g: (1962)
Dash	Delimiter to separate the season/episode reference	Literal: -
[Either] Season Episode	For most shows (with numbered episodes) use the season and episode reference for this episode. See the appropriate metadata source for details.	Typically denoted by “S” followed by the two digit season number and “E” followed by the two digit episode number, for example: S01E01 In case of double-episodes, the naming convention is as above, plus “-E” followed by the second episode’s two digit episode number, for example: S01E01-E02
[Or] Published Date	For other shows (which aren’t numbered) use the appropriate publishing date for this episode	Dates are in the format of YYYY.MM.DD, for example 1962.09.26
Dash	Delimiter to separate the season/episode reference	Literal: -
Full Title	The fill title of the episode, where spaces are replaced with the delimeter.	For example: The.Clampetts.Strike.Oil
Stacking Value	If multi-part, then note this part	If this file is only part of the same movie then denote the part type and part number. Valid stacking terms are cd, disc, disk, dvd, part, pt. Valid part numbers are 1 – 9. e.g. “disc2”
Extension	The three letter file extension	Typical file extensions are: avi, mkv, mpg, mp4, wmv Note that the extension is required to be an accurate representative handle for the media container [37] [38]

Note that the delimiter is the period character.

Example

For a simple example, the first episode of The Beverly Hillbillies[39] would be named and filed as follows:

```
multimedia/tv/series/The Beverly Hillbillies (1962)/Season 01/The.Beverly.Hillbillies.
(1962).-.S01E01.-.The.Clampetts.Strike.Oil.mp4
```

For a an example with punctuation in its title, the third episode of The Beverly Hillbillies[40] would be named and filed as follows:

```
multimedia/tv/series/The Beverly Hillbillies (1962)/Season 01/The.Beverly.Hillbillies.
(1962).-.S01E03.-.Meanwhile,.Back.at.the.Cabin.mp4
```

Script

The following script has been authored to aid the filing of episodes according to the above convention.

37 https://en.wikipedia.org/wiki/Comparison_of_video_container_formats
38 https://www.encoding.com/blog/2014/01/13/whats-difference-codecs-containers/
39 https://archive.org/download/TheBeverlyHillbillies/The%20Beverly%20Hillbillies/Season%201/The%20Beverly%20Hillbillies%20-%20S01E01%20-%20The%20Clampetts%20Strike%20Oil.mp4
40 https://archive.org/download/TheBeverlyHillbillies/The%20Beverly%20Hillbillies/Season%201/The%20Beverly%20Hillbillies%20-%20S01E03%20-%20Meanwhile%2C%20Back%20at%20the%20Cabin.mp4

```
#!/bin/bash


usage() {
  echo
  echo "Usage: ${0} \"Show Name (year)\" SS EE \"/p/file.ext\" [\"title\"] [-n]"
  echo
  echo
  echo "Example:"
  echo "    ${0} \"The Beverly Hillbillies (1962)\" 01 01 /tmp/mbatc.mp4"
  echo
  echo "Alternatively:"
  echo "    ${0} \\"
  echo "        \"The Beverly Hillbillies (1962)\" \\"
  echo "        01 01 /tmp/mbatc.mp4 \\"
  echo "        \"Meanwhile, Back at the Cabin\""
  echo
  echo "No Action:"
  echo "    ${0} \"The Beverly Hillbillies (1962)\" 01 01 /tmp/mbatc.mp4" -n
  echo
  exit 1
}


if [ ${#} -lt 4 -o ${#} -gt 6 ]
then
  usage
elif [ ${#2} -ne 2 ]
then
  echo "Error: the season needs to be two digits."
  usage
elif [ ${#3} -ne 2 ]
then
  echo "Error: the episode needs to be two digits."
  usage
elif [ ! -f "${4}" ]
then
  echo "Error: the source file does not exist."
  usage
elif [ ${#} -eq 6 -a "${6}" != "-n" ]
then
  usage
fi

show_name="${1}"
season="${2}"
episode="${3}"
source_file="${4}"
no_action="false"

dot_show=`echo "${show_name}" | sed "s/\ /\./g"`
dot_seep="-.S${season}E${episode}.-"
dot_ext="${source_file##*.}"
dot_name="unknown"
if [ ${#} -eq 5 ]
then
  if [ "${5}" == "-n" ]
  then
    no_action="true"
  else
    dot_name=`echo "${5}" | sed "s/\ /\./g"`
  fi
elif [ ${#} -eq 6 ]
then
  dot_name=`echo "${5}" | sed "s/\ /\./g"`
  no_action="true"
fi
dot_full="${dot_show}.${dot_seep}.${dot_name}.${dot_ext}"

if [ "${no_action}" == "true" ]
then
  echo
  echo "Not performing the following actions:"
else
  echo
  echo "Filing episode: ${dot_full}"
fi

if [ ! -d "${show_name}" ]
then
  if [ "${no_action}" == "true" ]
  then
    echo "  mkdir \"${show_name}\""
    echo "  chmod 755 \"${show_name}\""
  else
    mkdir "${show_name}"
    chmod 755 "${show_name}"
  fi
fi
if [ ! -d "${show_name}/Season ${season}" ]
then
  if [ "${no_action}" == "true" ]
  then
    echo "  mkdir \"${show_name}/Season ${season}\""
    echo "  chmod 755 \"${show_name}/Season ${season}\""
  else
    mkdir "${show_name}/Season ${season}"
    chmod 755 "${show_name}/Season ${season}"
  fi
fi
if [ -f "${show_name}/Season ${season}/${dot_full}" ]
then
  if [ "${no_action}" == "true" ]
  then
    echo "  No action would be taken, a file already exists at \"${show_name}/Season $
{season}/${dot_full}\""
  else
    echo "Warning: episode already exists, ignoring ${dot_full}"
  fi
else
  if [ "${no_action}" == "true" ]
  then
    echo "  cp \"${source_file}\" \"${show_name}/Season ${season}/${dot_full}\""
    echo "  chmod 644 \"${show_name}/Season ${season}/${dot_full}\""
  else
    cp "${source_file}" "${show_name}/Season ${season}/${dot_full}"
    chmod 644 "${show_name}/Season ${season}/${dot_full}"
  fi
fi
```

3.7 personal

The *personal* hierarchy is defined to capture the users' home directories (Unix) or home drives (Windows) – a common place for all users to have their personal records stored.

These directories are intended to be accessed by their individual users, only.

```
personal
├── user1
├── user2
└── usern
    ├── financial
    ├── multimedia
    ├── projects
    └── work
```

Other than the appropriate sub-structure specific mappings, there is no public metadata mapping available for handles/labelling in this structure.

Specification

The sub-directory hierarchy for "usern" is a directory per *user* or *device*, or colloquially "person". The naming convention will be locally relevant, with observed examples including –

Directories	Description
User ID	The users' logon identifier, where examples include: jdaniel1, j3di, u41092
Name	Full personal name, where examples include: "Jackson, Daniel", "Daniel.Jackson", "Daniel Jackson"
Email	Full email address, for example: daniel.jackson@localhost.localdomain

Note that the delimiter varies based on the scheme chosen.

It is recommended that the naming convention selected is one that can be most automated within the target environment, with the least additional administrative overhead.

3.7.1 personal/person

The remainder of the per-user sub-directory hierarchy is the re-used taxonomic elements that are defined elsewhere (see section 4.1 Top-Level File-System Layout, below, for details).

3.8 projects

The *projects* hierarchy is structured exclusively for rapid filing and access – a robust on-disk projects hierarchy will increase productivity.

As a top level structure, projects is assumed to be common (group) accessed – a community of users rather than by individual users. Where users have their own private projects, this structure can be reused in the *personal* hierarchy.

```
projects
├── project1
├── project2
└── projectn
    ├── 00 Administration
    ├── 01 Analysis
    ├── 02 Design
    ├── 03 Implementation
    ├── 04 Testing
    └── 05 Review
```

Specification

The sub-directory hierarchy is a directory per *project*.

When anticipating a substantial volume of projects consider additional classification. In an organisation this might be per market vertical, line of business, department or domain of expertise, or program of work. For the house-hold it might be helpful to group these projects by hobby – i.e. Programming, Gardening, Renovation.

The naming convention for projects will be locally relevant, with observed examples including –

Directories	Description
Name	Full project name, for example: Warp Speed
Date - Name	Financial year, or Calendar Year & Month & Day of initiation and full project name: 20200523 – Warp Speed
Customer - Name	Customer name and full project name: US Gov – Warp Speed
WBS	The Work Break-down Structure or Billing Code for the project: 10857.20
WBS - Name	Work Break-down Structure or Billing Code and full project name: 10857.20 – Warp Speed

Note that the delimiter is the space character.

3.8.1 projects/project

Under the project, the sub-directory hierarchy is designed to provide ready access to artefacts at each stage of the project life-cycle.

Directories	Description
00 Administration	Project administration such as project planning (project charter / project management plan, steering / working group terms of reference, etc), project finances (budget, rate cards, time-sheets, invoices, reimbursements, etc), project management (resource plans, RAID logs, status reports, PSG/PWG minutes, etc).
01 Analysis	Problem analysis such as audit or risk data, interview notes, research and ultimately scoping documents (scope, requirements, acceptance criteria).
02 Design	Solution design, including business process, application, infrastructure, security and service layers and corresponding artefacts (diagrams, supporting documentation such as quotes, commercial models, etc).

Directories	Description
03 Implementation	Solution build, including product orders and service provider engagement (RFS/RFQ) documents, communications / organisational change plans, etc.
04 Testing	Solution testing artefacts including the test plan and results documents (regression, performance, HA/DR, UA, etc)
05 Review	Project review artefacts, including post implementation review (PIR), audit, etc.

Note that the delimiter s the space character.

Example

For a simple example, of a finance document filed in the Rosie Replacement project:

```
projects/Rosie Replacement/00 Administration/Robot Maid Financing.xls
```

For a complex example, of a finance document filed in the NX5 project:

```
projects/Automation Program/NX5 Uplift Project/00 Administration/FY2035 Budget.xls
```

Script

The following script has been authored to automate creation of the above skeleton directory hierarchy.

```
#!/bin/bash


if [ ${#} -ne 1 ]
then
  echo "Usage: ${0} <path to project>"
  exit 1
fi

tld="${1}"
if [ ! -d "${tld}" ]
then
  echo "Error: ${0} path \"${tld}\" is not a directory."
  exit 1
fi

(
cat << EOF
  00 Administration
  01 Analysis
  02 Design
  03 Implementation
  04 Testing
  05 Review
EOF
) | while read deep_dir
do
  mkdir -p "${tld}/${deep_dir}"
  if [ ! -d "${tld}/${deep_dir}" ]
  then
    echo "Error: failed to create path \"${deep_dir}\"."
    exit 1
  fi
  chmod 755 "${tld}/${deep_dir}"
done
```

4 Implementation

Whether you’re using Box / Dropbox, Google Drive, iCloud, One Drive, S3, or a local file server, anywhere that multiple people are expected to rapidly file and retrieve digital assets, then it is recommended that this guide be put at the top level (root) directory of that shared storage, to help establish a consistent set of patterns and processes.

4.1 Top-Level File-System Layout

The top level structure will depend on the consuming entity.

By way of example, a business model would be well served with a top-level file-system layout containing finances, personal and projects.

```
/
├── devices
├── financial
├── human resources
├── legal
├── personal
│   └── multimedia
└── projects
```

Alternatively, a residential model would be well served with a top-level file-system layout containing multimedia and personal.

```
/
├── devices
├── multimedia
└── personal
    ├── family
    ├── financial
    ├── legal
    └── projects
```

5 Licensing

The following sections detail the licensing of the Saturn and the components that comprise Saturn.

5.1 Midnight Code Trademark

The term Midnight Code and the two half-moon mnemonic are registered trademarks of Ian Latter.

5.2 The Midnight Code Applications and libMidnightCode Library

All Midnight Code source code, including the libMidnightCode library and the Midnight Code applications are released with the following copyright, trademark and licensing terms.

```
/*

                                       _JNJ`
                                   .JNMH`
                                  JMMF`       `;.
                                .NMM)          `MN.
                                MMM)            (MML
                               (MMM`             MMML
                         M  I  D  N  I  G  H  T    C  o  D  E
                               (NMMF             MHNH
                                NMML            .MMM
                                 NMML          .NMH
                                  4MMNL       .#F
                                   `4HNNL_   `
                                       `"""`

                              Copyright (C) 2004-2020
           "Ian (Larry) Latter" <ian dot latter at midnightcode dot org>

        Midnight Code is a registered trademark of Ian Latter.

        This program is free software; you can redistribute it and/or modify
        it under the terms of the GNU General Public License as published by
        the Free Software Foundation, and mirrored at the Midnight Code web
        site; as at version 2 of the License only.

        This program is distributed in the hope that it will be useful,
        but WITHOUT ANY WARRANTY; without even the implied warranty of
        MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
        General Public License for more details.

        You should have received a copy of the GNU General Public License
        (version 2) along with this program; if not, write to the Free
        Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
        02111-1307 USA, or see http://midnightcode.org/gplv2.txt

*/
```

5.3 This Document

This document is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.