STATSBIBLIOTEKET DET KONGELIGE BIBLIOTEK

Archival Data Format Requirements

July 2004

The Royal Library, Copenhagen, Denmark
The State and University Library, Århus, Denmark

Main author:
Steen S. Christensen
The Royal Library
Postbox 2149
1016 København K
Denmark
Email: ssc@kb.dk

Table of Contents

1. Introduction

The general goal of Netarkivet is to collect and preserve Danica (Danish material) on the Internet. The preservation strategy distinguishes between bit-preservation and logical preservation. The purpose of bit-preservation is to ensure that the collected data are preserved. The purpose of logical preservation is to ensure that is possible to "understand" the preserved bits. The archival storage format must accommodate the storage requirements of bit- and logical- preservation. It is not sufficient just to store collected bits, but also necessary to store additional metadata in order to ensure the long term preservation of the data collection.

This document describes the requirements identified by Netarkivet for

- an archival storage format suited to preserve collections of internet data
- metadata needed to preserve collections of internet data

These requirements are a result of experiments and literature studies performed as part of the Netarchive phase 2 study [1] combined with input from the IIPC framework working group[2].

The requirements for the archival storage format are used to evaluate some common approaches used to archive Internet data. Based on this analysis it is recommended how to proceed in the Netarchive project to establish a concrete proposal for an archival storage format.

In chapter 2.1 the requirements for an archival storage format are presented. In chapter 3 different common storage formats are evaluated with respect to the presented requirements and an recommendation for a long-term storage format is made. In chapter 4 requirements for metadata information are listed. Finally the results are summarized in chapter 5.

2. Requirements

Various formats are already used to store internet data – examples are provided in the next chapter. Experiences have shown that not all formats are equally well suited for long-term preservation. This chapter lists provides a list of basic requirements a storage format must accommodate if it may be used for long term preservation.

2.1. The format must be suited for long-term storage

It is hard to define exactly what makes a storage format suite for long-term storage. Some of the known properties such as verification of data integrity and simple backup procedures have been stated as separate requirements. It is however hard to predict the exact usage of the data in the future. In order to simplify future usage of the stored data the following principles are proposed:

- The format should be simple to describe, understand and implement
- The format should not depend on specific hardware
- The format should not depend on specific operating systems
- The format should not depend on proprietary software
- The format should be robust against single points of failure

2.2. OAIS compatible

The storage format should match the requirements of the OAIS model [3].
The following requirements related to the storage format have been extracted from the OAIS model. Please refer to the OAIS model for a definition of terms, abbreviations and details on each requirement.

1. Documented procedures and policies for preserving AIPs ([3] section 3.2.5)
2. AIPs must be visible to Designated Communities ([3] section 3.2.6)
3. The format must support error checking, disaster recovery and media independence for Content Information and PDO ([3] section 4.1.1.3)
4. It must be possible to retrieve stored AIPs ([3] section 4.1.1.7)
5. It must be possible to separate stored Content Information into Content Data Object and Representation Information ([3] section 4.2.1.4.1)
6. The format must support storage of Preservation Description Information, reference information, context information, provenance and fixity information ([3] section 4.2.1.4.2).
7. The OAIS should avoid holding PDI or Content Information only in naming conventions of directory or file name structures ([3] section 4.2.1.4.3)
8. The format must support digital migration ([3] section 5.1) in general. More specifically the following migration types must be supported: Refreshment, Replication, Repackaging, Transformation ([3] section 5.1.3)
9. It must be possible to distinguish AIP versions, editions and derived AIPs ([3] section 5.1.4)

The following sections address most of the above issues.

2.3. The format must support all important Internet protocols

A protocol is supported if it is possible to store and retrieve all relevant information transferred by the protocol. For the HTTP protocol this means that the HTTP headers should be stored and that both HTTP requests and responses should be stored. Current Internet protocols that must be supported are FTP, HTTP, NNTP, SMTP. Handling of streamed data like RealVideo is mainly a question of obtaining and storing the correct bit stream.

2.4. The format must support meta-data

It is crucial that it is possible to enrich the stored bit-stream with additional metadata information. Examples of metadata information are: harvest description, access restrictions, and site categorization. The format should provide a mechanism that allows adding extra metadata after the harvested data have been added to the archive.

2.5. Data integrity must be easy to verify and maintain

Current storage media degrade over time. It is thus essential that it is possible to detect bit errors in archived data. It is obviously equally important that errors once detected are easily corrected. A simple scheme would be fingerprinting of the data and a method to restore from an additional backup of the data. Loss of data due to multiple bit-errors should only affect part of the archived data and not propagate to otherwise unaffected parts of the archive. Errors must not propagate to other copies of the data.

2.6. Data backup must be simple

A simple and reliable backup procedure must exist for the selected format.

When evaluating the complexity of a backup procedure the following should be considered:

- It must be possible to do a complete backup of all data
- It must be possible to backup only a subset of the data
- It must be possible to verify the success of a backup
- It must be possible to restore all data
- It must be possible to restore only a subset of the data

2.7. The format must support recording of access limitations

Some of the data in the archive may have access limitations. Examples of limitations are: only access for researchers, only access from specific terminals, no access at all, or only access for registered users. The information that determines access limitations may only be accessible at collection time, and the format must support recording of such information available at collection time or at a later point. Note that the format is **not** required to support administration of user rights and profiles.

2.8. The format must support authenticity information

The authenticity of data in the archive is of crucial importance in usage scenarios such as research or legal matters. It must be possible to reliably identify the person responsible for adding a collection of data to the archive.

2.9. It must be possible to retrieve the original bit-stream

It must be possible to store and retrieve the original bit-stream retrieved from the web. It must also be possible to store and retrieve the original request bit-stream.

2.10. It must be possible to delete material from the archive

During normal operation of the archive deletion of material should not occur. Exceptional situations may appear where the archive needs to delete specific items from the collection. An example might be a court order to delete illegal material. Deletion of material is not required to be an efficient process and an approach where data are rewritten without the parts that must be deleted may suffice. It must however be possible to record that material has been deleted and it must be possible to verify that only the targeted material has been deleted.

2.11. It must be easy to locate archived data

The success of an archive is highly dependent upon its usage. It is thus essential that data stored in the archive can be retrieved reliably and easily. References to data in the archive should be similar to the original references used when the data were originally collected combined with a timestamp.

2.12. Support format transformations

A wide variety of data formats are present on the Internet. A number of these formats will however not be supported by future applications. It is possible to address this problem in several ways, of which the most promising are:

- store the applications necessary to interpret the data and develop emulators
- convert the data to equivalent supported formats

It is not obvious which (if any) of these approaches are the best long term solution. The storage format should however provide appropriate support for both approaches.
The following general transformation requirements must be supported:

- Original data must still be accessible after transformation
- It must be possible to retrieve transformed data
- It must be possible to perform successive transformations of the same object
- It must be possible to store meta-data about the transformation
- It must be possible to retrieve the relationship information between the original data and the transformed versions of the original.

2.13. Support data compression

The cost of archiving data increases with the amount of raw bits that must be preserved. In order to reduce the cost of storage media it may be necessary to compress data. The data format must support an optional data compression of the stored data. It should be noted that the usage of compression should only be used after a careful consideration of the associated long term preservation risks.

2.14. Support duplicate reduction

If a website is harvested more frequently than it is updated the majority of the pages on the site are not expected to change. A lot of duplicated information will be stored in this scenario if everything needs to be stored for each visit. We generally want to reduce the total amount of data stored in order to reduce the total cost of storage. This leads to the following requirement.

If an object has not changed since last harvest:

- It is not necessary to store a new copy of the object data
- It must always be possible to record that the object was visited
- It must always be possible to record metadata

2.15. The format should be efficient

The success of any system depends not only on the functionality of the system but also on its performance and efficiency. The following scenario outlines some of the expected operations that the storage format should support: *During collection of data large amount of information are processed and stored. Once data have been collected a number of analyses such as search term indexing are performed on the complete dataset. A backup (and/or) copy of the collected data is made. Subsequent browsing of archived data retrieves a number of individual objects from the archive.* This leads to the following requirements:
Evaluation of the efficiency of the storage format must consider the following factors:

- Storing.
- Administration (backup, copying)
- Retrieval of individual objects
- Bulk access - large amounts of data are accessed

3. Analysis

This chapter evaluates some common approaches used to archive internet data with respect to the requirement list.

3.1. File + Folder

Each harvested URL-object is stored in a separate file. The path and filename is created from the URL of the object.

Example:
`URL: http://www.kb.dk/yd/ros/index.htm`
`File: /harvest/2003/`**`www.kb.dk/yd/ros/index.htm`**

The format is simple, easy to implement and describe.

The usage of folder and filenames as identification of individual URL-objects violates the OAIS recommendation that PDI and Content Information should not be stored in naming conventions of directory and file name structures (section 4.2.1.4.3).

There is a problem with using the object reference as the file name. Object references can be quite long, resulting in equally long file names. Some operating systems (even modern ones) do not handle long file names well. The huge amount of files produced by this scheme is also known to stress current operating systems.

This format can accommodate Internet Protocols based on URIs only, it does not handle storage of multiple versions of the same URI well as all versions would map to the same file name. This complicates handling of http post requests as the URI may remain unchanged while the post data changes. It is possible to store metadata with this scheme, simply by creating an additional file with the .metadata extension.

Procedures to verify, backup and restore files are already available. The large number of files will however stress these systems.

It is easy to locate and retrieve the original bit stream.
It is straightforward to implement an incremental storage scheme.

This model is very efficient for the retrieval of individual objects. The large number of files that must be handled by the operating system during writing, copying and streaming operations will on most current operating systems result in rather poor performance.

3.2. File + Index

This format is a variation of the File + Folder format. Each URL object is still stored in a separate file. The naming of the file does however not depend on the URL of the object. A separate index file is instead used to record the correspondence between files and URLs.

This format is in better agreement with the OAIS recommendations, it does however introduce a single point of failure as the loss of the index potentially results in all files associated with the index becoming unusable. The implementation is more complex than for file + folder as software is needed to handle the index. The usage of an index has the advantage that the problem with long references disappears and it becomes possible to handle multiple files with the same URI reference. The problems with the large number of files however remain.

3.3. ARC

The ARC format is used by Internet Archive to store data [4]. The format consists of a sequence of document records. Each record starts with a header line containing some bare bones metadata. The header is followed by the actual data returned by the web-protocol. The main fields of the header line are the URI of the document, the timestamp of when the data were acquired and the size of the data block following the header line.

The complexity of this format is comparable to the complexity of the file + index format. It is rather straightforward to implement support for the reading and writing of the format. The ARC format does not have the problems with long filenames and many files that the two previous formats have. ARC supports storage of bit streams.

It is possible to store multiple documents with the same URI originating from different http post request. The format does however not support the storage of the actual post request.

The metadata support of the ARC format is limited to the barebones metadata stored in the document headers. The lack of metadata support poses some problems:

- it is hard to record access restriction information at collection time
- recording of document deletion is only done indirectly
- it is possible to store the results of format transformations but there is no convenient place to store information about the transformation
- the format does not define a place to record information needed to handle duplicate reduction (note that an interpretation of some header fields has been used by IA for duplicate reduction purposes)

It is easy to check the integrity of an ARC based archive using available tools.
The smaller number of files in an ARC based archive simplifies backup.
Internet Archive have demonstrated retrieval of data from an ARC based archive. Notice that efficient access to data requires that an external index is created.

One of the main advantages of the format is its efficiency. Internet Archive have demonstrated that it is possible to collect, store and retrieve in excess of 300 TBytes of data in this format.

3.4. XML

A popular technology for data description is XML and one might consider using XML as the underlying technology of a file format.

It is possible to use XML both in a file + folder/file + index like scheme resulting in a large number of files or in an ARC like scheme where many URL-objects are represented in one XML file. If XML is used in the file like schemes we get the disadvantages of many files. If many URL-objects are stored in a single XML file we get some performance related problems. In order to interpret an XML file it is generally necessary to parse and load the complete file. This is not efficient when individual objects are retrieved. A scheme where an XML file is processed in an ARC-like style might alleviate this problem. In this scheme an external index is created that marks the byte offset into the XML file of the individual URI-objects. Although access to individual objects is improved in this scheme it has the disadvantage that XML is used in a non-standard way.

Embedding of binary data in XML files is associated with an overhead as a 64 bits encoding of data is required. It is unclear what the overhead associated with the creation of a XML file is during collection of data.

3.5. Databases

Another alternative to the file based storage schemes is to use a database system to store data retrieved from the Internet. The usage of databases has the advantage that just about any kind of data and metadata can be stored.

One major disadvantage however is that database systems aren't suited for long-term storage. Database systems are under continuous development and improvement and normally only the most recent versions are supported. The archive becomes crucially dependent on the selected database system and the dependencies of this system. The consequences of reliance on one database system are comparable to the consequences of reliance on one operating system.

As this is considered a showstopper no detailed analysis has been made to determine how a database relates to the other requirements.

3.6. Summary

None of the considered formats seem to provide a perfect match for the list of requirements. ARC is the most efficient format but it lacks meta-data capabilities. XML has very strong meta-data capabilities but a number of efficiency problems. The file + folder approach is simple but the problems with long file names makes this scheme unsuitable for usage in an archive. The file + index scheme is simple and robust and is probably the best choice for small to medium scale usage. The large number of files however makes the file + index scheme less ideal for large-scale usage. The properties of the different formats are briefly summarized in the following table, where

✖ - marks a requirement that is not supported or very hard to support using the format

(✓)- marks a requirement that although not an integral part of the format, may be supported.

✓ - marks a requirement that is easily supported by the format

format requirements	File + Folder	File + Index	ARC	XML	Databases
Suited for long-term storage	✓	(✓)	✓	✓	✖
OAIS compatible	✖	✓	✓	✓	✓
Support all important Internet protocols	✖	(✓)	(✓)	✓	✓
Support meta-data	✓	✓	(✓)	✓	✓
Data integrity must be easy to verify and maintain	(✓)	(✓)	✓	✓	✓
Data backup must be simple	✓	✓	✓	✓	✓
Support recording of access limitations	(✓)	(✓)	(✓)	✓	✓
Support authenticity information	(✓)	(✓)	(✓)	✓	✓
Possible to retrieve the original bitstream	✓	✓	✓	(✓)	✓
Possible to delete material from the archive	✓	✓	✓	✓	✓
Easy to locate archived data	✓	✓	✓	✓	✓
Support format transformations	(✓)	(✓)	(✓)	✓	✓
Support data compression	✓	✓	✓	✓	✓
Support duplicate reduction	✓	✓	(✓)	✓	✓
Efficient	✖	✖	✓	✖	?

The main problems thus seem to be:

- The large number of URL-objects expected makes it impractical to store each object in a separate file.
- The ARC format has demonstrated that it handles many URL-objects well but it lacks sufficient powerful meta-data capabilities.
- XML handles meta-data well but is inefficient

- Databases are not a viable long term solution

The most obvious options now are:

- Extend the ARC format to incorporate better meta-data capabilities
- Improve the efficiency of the XML format
- Create a new format

The efficiency problems of XML are intrinsic and not easily corrected. It seems however to be possible to add better meta-data capabilities to the ARC format. As it is preferable to use an existing format, the recommendation becomes:

- It is recommended to extend the ARC format to accommodate all requirements.

4. Metadata Requirements

This chapter summarizes the requirements for the metadata format.

4.1. Minimize resource-demanding workflows

It is an overall requirement that metadata creation must be automatic as far as possible. The collection of data from the Internet is expected to produce large amounts of data. Manual processes used to create metadata should thus be kept to an absolute minimum for practical and economic reasons.

4.2. The format must be extendable

It must be possible to extend the format with additional information. It is not possible to predict future metadata requirements. The long term viability of any metadata format thus becomes crucially dependent upon its adaptability to new technologies and/or requirements.

4.3. Harvester setup

It is anticipated that the majority of the internet data in the archive will be collected using a web harvester application. The individual setup parameters differ significantly between different harvester applications. The following requirements describe a minimum set of information that is required to describe the most common harvester setups:

1. A unique reference to an URL object describing the harvester setup
2. A detailed textual description of the purpose of the harvesting and limitations in the collected data
3. A list of sites included in the harvest
4. A list of specific URL objects selected for harvesting
5. A list of seed URLs (or file/folder paths)
6. A classification of the harvest type (cross-section, selective, event, deposit)
7. Identity of harvest creator: institution, name, email, digital signature[1]
8. Reference to harvester source code
9. Harvester link following limits: depth limit, width limit
10. Harvester format limits: PDF, TIFF etc. inclusion or exclusion
11. Harvester handling of robots.txt limits
12. Harvester handling of site access restrictions (login, password) and encryption
13. References to provider agreements[2]

4.4. Harvest statistics

It must be possible to record the following information once a harvest is completed:

1. Start time of the harvest

[1] In order to ensure authenticity, it is very important to record the creator of the harvester setup and harvesting e.g. the harvesting institution, person and the digital signature, especially if some of the harvested URL objects should later be used in legal matters.

[2] In some cases a special agreement has been made with one or more data providers. It must be possible to specify a reference to such agreements.

2. Completion time of the harvest
3. Status of the harvest (succeeded, failed, canceled)
4. Total number of URL objects harvested
5. Total amount of bytes harvested
6. Reference to additional harvesting statistics

4.5. Information for individual URI objects

It must be possible to record various kinds of information for each URI object harvested. It should be noted that some of the information is only available at harvest time whereas other kinds of information may only be available after the original URI object has been stored in the archive.

It must be possible to record the following information for each URI:

1. reference to the harvester setup
2. the reason for not harvesting a URI
3. the reason for harvesting a URI
4. did the URI lookup succeed or return an error
5. the URI request data
6. the URL requested
7. basic access restrictions
8. date and time the metadata record was created
9. mime type information (returned by http protocol)
10. format version (example MS Word v. 5.0)
11. format verification [3]
12. size of the object
13. fingerprint
14. unique key – persistent identifier
15. transformation information:
    a) from format
    b) to format
    c) transformation program used
    d) date
    e) status
    f) comments
16. references to additional standalone meta-data[4]
17. links extracted by the harvester at harvest time
18. digital signature - to document origin and authenticity.
19. virus check – marking an object as virus infected
20. access illegal – marks all access to the object being illegal
21. duplication elimination information – this is an original or a duplicate

[3] Wrong format types provided by the object producers are not uncommon. It must be possible to indicate the format is verified - either during or after harvesting.

[4] Some types of deposits and selective harvestings provides previous created metadata about an URL object in stand alone files – either on a URL object level or on a folder level. Other URL objects embed previous created metadata e.g. html, pdf , word or tiff objects. In such cases will the reference be a self reference.

It must be possible to encrypt the URL request when it contains passwords or other strictly confidential information.

4.6. Changes and additions to already recorded metadata

It is anticipated that an URI-object is added to the archive with an initial set of associated metadata. Subsequent processing may however result in additional metadata that must be added to the object or the original information may need to be corrected. Archived data should never be changed from an integrity point of view.

Examples:

- After adding an object to the archive a new antivirus scanner marks the record as virus infected. A new metadata entry is added to the archive that augments the existing metadata information in the archive with the additional information that the object contains a virus.
- An object is added to the archive with the metadata information that it is a PDF object. A later analysis determines that the object is in fact not a PDF object. A new metadata record is added to the archive with the information that the mime-type information in the original record must be changed.

This results in the following set of requirements:

1. It must be possible to add new metadata to already archived metadata sections
2. It must be possible to record changes to already written metadata information without, changing/deleting the original archived information

5. Conclusion

A set of requirements for an archival storage format suited for storage of material collected from the internet has been presented. Based on these requirements common storage formats have been investigated. None of the investigated formats match the requirements perfectly and it is recommended to extend the ARC format to accommodate all requirements. Finally a list of metadata requirements has been presented.

6. References

[1] Netarkivet: http://www.netarchive.dk

[2] IIPC framework group: http://www.netpreserve.org/about/framework.php

[3] OAIS blue-book, Reference Model for an Open Archival Information System (OAIS) CCSDS 650.0-B-1, January 2002,CCSDS. http://www.ccsds.org/CCSDS/documents/650x0b1.pdf

[4] Internet archive ARC format, http://www.archive.org/web/researcher/ArcFileFormat.php