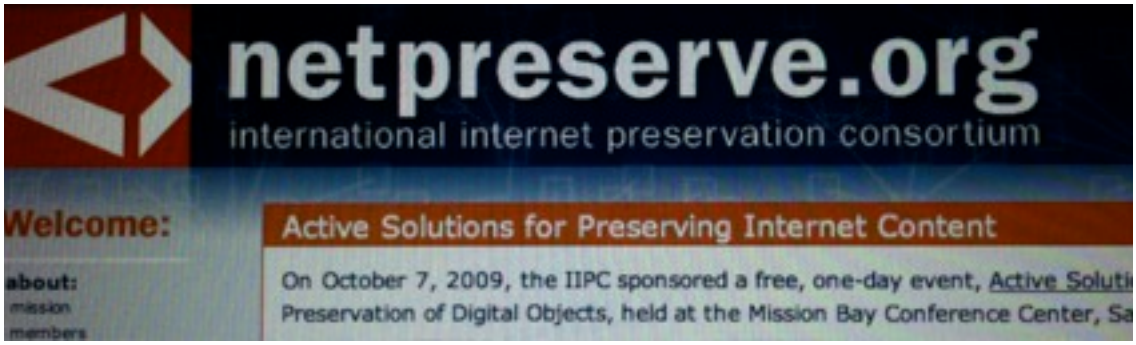
HANZO ARCHIVES




64 Clifton Street, London,
EC2A 4HB, U.K

+44 20 8816 8226

www.hanzoarchives.com
contact@hanzoarchives.com

# WARC Tools Phase III Functional Requirements Specification

**Prepared for:** IIPC
**Prepared by:** Hanzo

**Delivered:** Tuesday, 20 October 2009

**Abstract & Summation:** This Functional Requirements Specifications (FRS) is a base-line statement of functional requirements for the WARC Tools Phase III project. Also includes high-level estimates of effort and schedule.

HANZO ARCHIVES

# Table of **Contents**

64 Clifton Street, London,
EC2A 4HB, U.K

+44 20 8816 8226

www.hanzoarchives.com
contact@hanzoarchives.com

**Introduction** 1

Background ..... 1

WARC Tools Phase I and Phase II ..... 1

WARC Tools Phase III ..... 1

Outline Approach ..... 1

**Specification** 3

Default Behaviours ..... 3

Migration Application ..... 3

Migration Tool ..... 4

Validation Tool ..... 6

Repackaging Tool ..... 6

Reporting Application ..... 7

WARC Summary Tool ..... 7

WARC Browser Integration ..... 8

Quality Assurance Tool ..... 8

WARC Comparator ..... 8

Enhanced WARC Browser ..... 8

64 Clifton Street, London,
EC2A 4HB, U.K

+44 20 8816 8226

www.hanzoarchives.com
contact@hanzoarchives.com

# Introduction

## Background

**WARC Tools Phase I and Phase II**

The main goal of the WARC Tools project is to facilitate and promote the adoption of the WARC file format for storing web archives by the mainstream web development community by providing an open source software library, a set of command line tools, web server plug-ins and technical documentation for manipulation and management of web archive files, or WARC files.

This project has delivered a core software library called "libwarc" and a set of end user command line tools, extensions to existing tools, and simple web applications for accessing WARC content. In addition all the libraries have APIs and dynamic language bindings. The library and tools are scriptable (command lines in shell scripts, dynamic language bindings to the library), and programmable (dynamic language bindings, Java packages, and the C library itself).

Together, these deliverables are known as "WARC Tools" and are available as free software, here: http://code.google.com/p/warc-tools/

In parallel, Hanzo have developed an extension to WARC Tools that provides full-text and metadata search of WARC files, known as "Search Tools", which is also available as free software, here: http://code.google.com/p/search-tools/

Together, these projects provide a compelling implementation of the WARC standard, and provide a robust engineering foundation for further development of tools and applications centred around WARC files and their usage.

**WARC Tools Phase III**

Following the development of libwarc and associated WARC Tools in phases I and II, Hanzo will implement a third development phase, WARC Tools Phase III. This phase will build upon the original libwarc, extending the collection of WARC Tools and implement a full migration application. Phase III will include community participation in the specification of the tools and applications, these will come from a number of International Internet Preservation Consortium (IIPC) member institutions, and similarly for testing.

Phase III implementation will follow the original philosophy of providing powerful tools to enable crawl engineers, web archivists, researchers and other WARC users to easily manipulate and explore collections of web archive content without needing to write complex low-level code.

## Outline Approach

WARC Tools Phase III will specify, build, test and deploy the following:

1. Migration Application
2. Repackaging Tool
3. Reporting Application

4. Quality Assurance Tool
5. Enhanced WARC Browser

In addition, Hanzo will carry out the following tasks as part of the project:

1. Collaborative requirements gathering and requirements management
2. Libwarc maintenance
3. Project management and meetings
4. Deployment within participating institutions

The project will be organised around a collaborative engagement with the web archiving community, inviting IIPC member institutions to collaborate and contribute functional requirements and deployment and testing of the tools and applications in their own institutions.

Specifically, Hanzo plan to involve IIPC member institutions to work on creating and/or reviewing requirements during the inception phase of the project, which includes a questionnaire prepared by BnF. During the transition phase, when working towards project closure, IIPC member institutions are invited to collaborate on acceptance testing of the tools within their "real world" settings; involving deployment of the tools and applications within participating institutions.

Participating institutions are:

- Denmark (Netarchive.dk) — will contribute at the testing level, will use a copy of some of their data as test data. Happy with the specification to date, but wish to be kept informed of changes and developments.
- Norway (NL) — will contribute through testing and collaborating, based on their own plans to convert their web-archive to WARC (approximately 2,000,000,000 URLs).
- UK (BL) — will contribute both at the requirement specification and testing stages of the project, possibly using their selective archive as test data.
- France (BnF) — will contribute at specification stage and at testing stage also. Will provide ARC files as test data. Will facilitate circulation and sharing of all useful information between this project and other ongoing relevant work on WARC usage run by the IIPC Preservation WG and the IIPC Technical Committee.
- Sweden (NL) — form of contribution to be confirmed.
- New Zealand — will contribute in the specification and the migration process.

This approach will ensure the community will have first-hand use and knowledge of the tools in their real-life environment, serving as further acceptance and verification of the toolset as a whole.

This Functional Requirements Specifications (FRS) is a base-line statement of functional requirements for the project as a whole, including high-level estimates of effort and schedule, based on the above approach and assumptions and best-practice estimation. Following review of the FRS and questionnaire additional estimation and scoping exercises will be carried out to ensure the project remains within the base-line scope. Any institutional requirements that cause the project to exceed this scope will be flagged and the institution may either de-scope the requirement or provide additional financial contributions to the project, thereby keeping the requirement in scope.

Also listed in this document are Non-Functional Requirements (NFR), which are requireents reflecting higher level goals.

# Specification

### Default Behaviours

The following features of WARC Tools commands are to be provided by default. All other modules of WARC Tools, such as migration tool, repackaging, reporting, etc., will inherit these capabilities.

NFR 1 — The tools shall be able to process a set of multiple WARC files at the same time. For this purpose, the tools will allow you to choose/group them by:

- explicit naming
- wildcard name matching
- size limitation
- number of items

NFR 2 — The tools shall be able to scale to process large collections using distributed processing and data transport (see FRS 10 and 11)

NFR 3 — The tools shall offer best possible performance to correctly process large collections (I/O bound)

NFR 4 — The tools shall be able to run on multiple machines **but will not provide functionality for dealing with hardware failures (out of scope)**

NFR 5 — Implementation of the tools shall avoid unnecessary technology dependencies and shall not include development of partner-specific integration technologies

NFR 6 — The tools shall be compliant with Java development environments, by using web services, RESTful API's, etc.

NFR 7 — The tools shall provide logging facilities, for example, to follow command progression, i.e. process duration, output levels, etc.

NFR 8 — The tools shall provide enhanced usability by providing easy to adapt shell script wrappers for the major commands

NFR 9 — It shall be possible to operate the tools in modes that restrict their effects to selected collection subsets. For example, select by name (see NFR1), random sampling, etc.

### Migration Application

The migration application provides workflow to support and manage the arc → warc migration process and verification functionality to check and validate by content and metadata (time, URL, etc.).

NFR 10 — The workflow system shall have a configurable management strategy for migration from ARCs to WARCs.

Additionally, the migration application would provide checks and balances throughout the migration process to ensure it is verifiable and working properly. This application will be designed to support large-scale migration to WARC, in which we expect to migrate millions of ARC files to WARC. It will be important to take real-life requirements from IIPC member institutions into account.

The migration application will consist of four components:

1. A workflow/configuration application to set up the migration configuration. A web user interface shall be available to set up the migration. End point is a migration configuration that can be used in a migration job. Configuration shall persist as a configuration file or database.
2. Migration tool to migrate content from a collection of ARC files to WARC files

3. Validation command-line tool to validate content in a collection of WARC files against the original ARC files content.
4. A console to report on progress of active migration processes (such as the Heritrix console), manage the process, view logs, etc.

Non-functional requirements concerning scale of migration will need to be developed alongside the functional requirements. It should be noted that libwarc and WARC Tools in general have been implemented to deal with very large scale operations by ensuring all functionality is performed with minimal memory usage and performed atomically by simple tools in the style of UNIX.

NFR 11 — The migration workflow system will be driven by command line tools and scripts

NFR 11.1 – Migration configurations can be built with a Web User Interface

NFR 11.2 – Active migrations can be monitored, paused, and restarted (a console application) with a Web User Interface

NFR 12 — Pre and post conversion actions will be provided for at each level of the migration, e.g. record, WARC, job

The migration and validation tools are described below.

**Migration Tool**

A tool will be created to migrate content from a collection of ARC files to a collection of WARC files.

```
$ arc_warc_migrate <ARC_FILES> <CONFIG> [options]
```

The command arc_warc_migrate uses a list of ARC files and will migrate them based on the configuration file CONFIG. Each field should describe how it will be translated into WARC records. If some fields are missing, errors will be generated that require operator intervention.

FR 1 — The migration workflow shall provide a clear Application Programming Interface (API) to handle the migration process and default configuration.

FR 2 — The migration API will require inclusion of only one header file.

FR 3 — A configuration must be provided explicitly, errors should be generated in the absence of a configuration.

FR 4 — The user shall be able to provide metadata related to the conversion that will be stored in the converted files (e.g., institution, context, crawler, collection name, ...)

FR 5 — It shall be possible to request that automatically generated migration metadata be stored in the converted files (e.g., OS/Kernel type, original ARC name, ARC size, ARC digests, conversion timestamp, ARC record offset, ...)

NFR 13 — IIPC members should provide us with the default METADATA they want to be included for migration.

Example configuration file directives:

- *undefined_mime*: by default, “unknown_mime” will be applied when no MIME type is found
- *ignore_bad_record*: if any error occurs during the conversion, skip the record and log the problem.
- *segment_filter_record*: the ARC record will be split into multiple WARC Continuation records each one not longer than N bytes. The last record maybe less than N bytes; note that no record will end in such a way as to split a multi-byte character.
- *metadata_record*: a user-supplied metadata record that will be added one to each WARC file.

FR 6 — The migration workflow shall be able to call external tools and services (e.g., database queries, shell commands, web service).

The migration tool will permit use of external tools or services to generate preservation and migration metadata to be stored in the WARC records. As such the arc2warc command will become one out of several processes to be used for migration.

Examples of external tools and services are:

- tools to create persistent identifiers for WARC-record-ID
- file identification services (DROID, FILE, JHOVE) to ascertain the format of a file and write it as WARC-identified-payload.
- tools to scan files for malware and viruses and identify them as such in the WARC file

FR 7 — The migration process shall use persistent, opaque, unique, and global identifiers for records access.

For example:

- Noid: http://search.cpan.org/~jak/Noid-0.424
- ARK: http://www.cdlib.org/inside/diglib/ark
- UUID: http://en.wikipedia.org/wiki/Globally_Unique_Identifier

FR 8 — A (default) external tool (e.g., ClamAV) will be used to scan files before conversion (pre-conversion step).

FR 9 — The API should be flexible enough to allow external tools such as JHOVE, DROID to be used for file format identification.

Such external tools or services will be identified in the strategy configuration file. The manager should be able to follow the progress of a migration process (eg.,number of migrated records/files, time spent and time remaining, etc.), whether it is running on one machine or is distributed across machines.

FR 10 — The ARC to WARC migration should be able to run on multiple machines that will be easy to deploy. This may be achieved by using a simple messaging infrastructure.

FR 11 —Logging during the migration may be turned on/off at any time .

FR 12 — Software checkpoints (such as start-trans, end-trans) may be added during processing for managing atomic transactions (e.g., operations on a file, these must start and complete to be accepted in the migration), except where operating system limitations would prohibit it.

Note that it should be possible to avoid having to specify any network code or learn a foreign API, which accounts for considerable complexity, for example, in map-reduce solutions (e.g., Hadoop, Google Map/Reduce). The process migration will take care of these issues at runtime. To start more clients in parallel, this code would be called multiple times from either the local machine or any machine on the local network, causing more clients to begin processing the server data concurrently. This will form the parallel arc2warc conversion strategy used in WARC Tools Phase III.

FR 13 — It shall be possible to perform a “dry-run migration”. That is, a blank conversion to generate useful reports (performance issues, bad ARCs, time estimation ...) without writing real WARC data (i.e. simulating only). Dry run is defined entirely by the configuration.

While doing a migration, we shall be able to detect and avoid duplicate documents. For this purpose, we will implement checksum-based deduplication. Other deduplication mechanisms can be implemented with pre- record actions.

NFR 14 — Deduplication may be run before migration, inside a batch process.

FR 14 — Duplicate detection shall find and report WARC records with the same checksums. A centralised database will be used to store previously processed records, against which new ones will be compared.

Post migration database can be used for reporting on migrations job-by-job.

### Validation Tool

A tool will be created to validate content from a collection of WARC files.

```
$ arc_warc_verify <ARCFILE> <WARCFILE> <USER_DEFINED_ARC_READER> [options]
```

This command validates whether a migration carried out with arc_warc_migrate was successful or not. The validation process looks at each record contained in the ARC files, extracts the corresponding content from the WARC files and compares them by computing a checksums. It will also be possible to compare a user-defined set of metadata values.

FR 15 — To validate that the migration succeeded, arc_warc_verify will use the METADATA included in the newly generated WARC files to match records with their corresponding ARC files (see FR4, and FR5).

An option will be provided to let the tool sample a random percentage of records rather than perform an exhaustive check. This is provided mainly for rapid verification of large collections, advisable when confidence in the process is high.

FR 15.1 — Core validation shall use payload checksum comparisons.

FR 16 — Sampling will be provided where applicable to quickly validate the conversions (FR 15.1 and FR 18 for example).

For this purpose, arc_warc_verify will be aware of record MIME-Types and will sample record types in proportion to their representation.

By default, the process will use WARC Tools ARC reader and WARC reader/writer. As an alternative it can use an external ARC reader (for example arcreader from Heritrix) or even a wrapper on a non-ARC based extractor.

FR 17 — Validation shall use Heritrix's arcreader to double check that the original ARC record was correctly converted to WARC.

FR 18 — It shall be possible to perform a round trip validation: migrate the newly created WARC file back to ARC and compare both checksums (NFR 5)

### Repackaging Tool

A tool will be created to extract WARC records from a collection and generate a new set of WARC files. This is useful for testing, QA, initial crawl planning and transferring selected material to third parties.

```
$ warc_repackage -i <WARC_FILE> <WARC_PATTERN> [options]
```

WARC Repackage takes as input a set of WARC files, extracts the content according to the supplied options, and writes the content to new WARC files.

FR 19 — It shall be possible to repackage WARC files by filtering records based on URL (regular expressions)

FR 20 — It shall be possible to repackage WARC files by filtering records based on MIME-Types.

FR 21 — It shall be possible to repackage WARC files by filtering records based on size.

FR 22 — It shall be possible to repackage WARC files by filtering records based on timestamp (e.g., dates interval).

FR 23 — It shall be possible to repackage WARC files by filtering on any field in the WARC specification.

FR 24 – Repackaging will allow pre and post record, file and job operations

FR 24.1 – Pre operation can prevent a file or record being processed by returning value

Options will specify a number of filters, based on URL, timestamp, WARC metadata, etc., using both simple and advanced regex mode (similar to “grep -e”).

The identified records are written into new WARC files, with output names based on the WARC_PATTERN specified. Options may be specified for output WARC file size, etc.

FR 25 — Each repackaged WARC file shall include a user-defined METADATA record which will describe the extraction context (e.g., filter used, involved WARC files, ... - see also FR4).

Additional metadata records will be added to show the output WARCs as repackaged.

## Reporting Application

The reporting application consists of two components:

1. WARC Summary Tool to report on the content of a collection of WARC files
2. WARC Browser Integration to provide a web user interface for the reports

**WARC Summary Tool**

A tool will be created to report on the content of a collection of WARC files. The basic report form is similar to the form produced by the crawlers; this can be useful when a crawler report is not otherwise available for a collection.

FR 26 — The migration framework shall provide tools to build reports from WARC files

The summary tool also enables fine grained manipulation of the reports and the production of useful variations.

```
warc_summary WARC_FILES [options]
```

WARC_FILES can be a list of WARC files (including wildcards, etc.) or a directory WARC files.

Example reporting outputs:

1. (default) summary - similar to a crawl-report, shows the number of documents, start / end date for contents, etc, with options to detect and report multiple crawls (if the WARC files are from multiple crawls), similar to “lscrawl”.
2. mimetypes - a mimetype report, with options to show a breakdown by host, domain , crawl, etc.
3. hostnames - a host report, with options to show a breakdown by crawl , number of documents, etc.
4. status codes - a frequency table of status codes, with options to show breakdown by host, domain, crawl, etc.
5. pseudo crawl log - a list of time-ordered URLs and status information from the WARC file, with options to show a breakdown by crawl, host, domain, etc.

NFR 16 — Repackaging filters can be used by the summary module

FR 27 — It shall be possible to export summaries in various formats (e.g., XML, CSV)

This command will also have options for XML and CSV output and filtering, for example, to restrict the report by time ranges, domain or other metadata values.

**WARC Browser Integration**

The proposed work will include the integration of ‘’warc_summary” and “warc_browser” (from phase II), which will display a directory full of WARC files. Integration will enhance the front page to show links to view and explore the above reports including graphs.

FR 28 — Enhance the WARC browser to display aggregated WARCs summaries in its UI

FR 29 — Enhance the WARC browser to display a manifest of WARC files and their locations

## Quality Assurance Tool

**WARC Comparator**

A tool will be created to make a comparison of similar crawls, based on analysis of their WARC files.

```
$ warc_compare <CRAWL_1_WARC_FILES> ... <CRAWL_N_WARC_FILES> [options]
```

where WARC_FILES can be a list of WARC files (including wildcards, etc.) or a directory of WARC files. Options have a variety of functions:

- filters to limit by URL patterns, timestamp, metadata records, etc.,
- to show reports broken down by domain, hostname, etc., and
- to add a graph generator for the purpose of WARC quality assurance (Q.A).

FR 30 — Provide a “diff“ tool to compare WARC sets based on defined criteria (e.g., timestamp, hostname, etc.)

FR 31 — Provide a tool to draw difference graphs between WARC collections (see FR 30)

A WARC ”set comparator” enables comparison of crawl results contained in two sets of WARC files. It will be possible to identify important differences (deltas) between multiple/repeated crawls of the same seed. This will be very useful for QA of repetitive crawls.

FR 32 — Provide a way to view crawls deltas for quality assurance

For example, consider a monthly crawl of “www.foo.org”. A WARC Set Comparator would provide a delta report showing the following information:

- Percentage change in what is provided in the summary
- A listing of changed pages, with links to the major deviations

**Enhanced WARC Browser**

Small enhancements to the WARC Browser will be implemented to ensure that it provides a good quality browsing experience. Typical enhancements will include a proxy mode and server-side rewriting using a rules-based approach.

FR 33 — Implement WARC browser server side rewriting

FR 34 — Integrate a proxy mode inside the WARC browser

Another enhancement is to ensure the Search Tools project is updated and integrated; this will produce full text indexes and provide full-text search functionality to the WARC Browser and on the command line.

FR 35 — Integrate the full-text search (search-tools project) module with the WARC browser to provide users with WARC indexing/searching capabilities.