— Open Source WARC Tools —
Non-Functional Requirements

v1.0

Mark Middleton

Hanzo Archives Limited, Bonchurch Manor, Bonchurch Shute, Ventnor, Isle of Wight, PO38 1NU, United Kingdom

Registered address: 64 Clifton Street, London, EC2A 4HB, U.K.
Registered in England. Company No. 5410483. VAT No. 912 8708 19

T +44 78 0144 6473 F +44 87 0164 1643

markm@hanzoarchives.com

http://www.hanzoarchives.com

Table of Contents

Introduction

1. Background

The main goal of WARC Tools 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 WARC files.

WARC files are produced by web archiving crawlers, such as Heritrix — the open-source, extensible, web-scale, archiving quality web crawler developed by the Internet Archive with the Nordic National Libraries — and Hanzo's own commercial crawlers.

The project is lead by Hanzo Archives, in collaboration with the Internet Archive Web team, and supported by the International Internet Preservation Consortium (IIPC).

The core functionality of WARC Tools will be implemented as a library, and the functionality to be made available to end users external systems as command line tools, extensions to existing tools, and simple web applications for accessing WARC content. In addition the library will have APIs, Java and dynamic language bindings and will be made available as a software library for developers in these languages.

The library and tools will be scriptable (command lines in shell scripts, dynamic language bindings to the library), and programmable (dynamic language bindings, Java packages, and the C library itself).

Migration and interoperability with legacy tools is an important application for these tools and this project will implement functionality for legacy tools and the artefacts created by them — this will enable rapid progression and adoption of WARC by existing web archiving institutions and the mainstream web development community. To this end, integrating or linking to HTTrack, curl and wget for example are all going to be part of WARC Tools.

The library and tools will be implemented in ANSI C and will be highly portable, with build/installation on various Linux and Unix distributions, as well as Windows, together with unix man pages, build and installation guides, developer guides, etc.

The library and the tools will be open source, together with installers, documentation and man pages.

1.1. Resources

The project home page, including source code repository: http://code.google.com/p/warc-tools/

The project mailing list: http://groups.google.com/group/warc-tools

2. Proposed Non-functional Deliverables

This project will develop and release the deliverables identified in the document "Functional Requirements Specification", these deliverables will be shaped considerably various non-functional requirements, summarised as deliverables in the following table.

DELIVERABLE	DESCRIPTION
Installation scripts and tools	Linux install scripts for the library and tools, apache install scripts and lighttp install scripts
Unix command line style tools	The deliverables may be used as a library to be embedded in larger systems or may be used as command line tools, Unix 'pipe' friendly and scriptable
Community release and documentation	An open source repository, documentation, source code, release management, mailing list and ongoing community support
Orthogonal technology	All software deliverables within this project will be written in C with dynamic language bindings, thereby providing a fully independent implementation and evaluation of the standards for WARC and IIPC using technology favoured by the Linux development communities, the most prevalent open source tools outside of Java. This is important for a standards forum such as IIPC, where a dominant implementation in a single language can bias a standard towards that implementation, or worse, the implementation may replace the standard.

Non-functional requirements do not specify 'what' a program is to perform, but may strongly influence 'how', and therefore influence architectural and design decisions.

For example, for a given feature of the tools, differing non-functional requirements may lead the program to function as a command line tool or as a web service or a batch program, but the actual functionality would be the same.

Requirements

This section describes the non-functional requirements of the WARC Tools at a level of detail sufficient to guide the architectural shape of the solution and the project as a whole.

3. Technical Requirements

NFR 1 — The core functionality of WARC Tools, specifically the WARC writer/reader, shall be implemented as a comprehensive, standalone, general purpose software library — libwarc.

NFR 2 — WARC v0.17 shall be the baseline version of the the WARC standard used in this project.

3.1. Installation scripts and tools

NFR 3 — WARC Tools shall be highly portable, running on a range of Linux/Unix platforms and XP as a minimum, together with man pages, build and installation guides, developer guides, etc.

3.2. Unix command-line style tools

NFR 4 — Utility and application level functionality of WARC Tools shall be made available to end users as command line tools, extensions to existing tools, and simple web applications for accessing WARC content.

3.3. Community release and documentation

NFR 5 — Communication and support shall be provided to the open source community. This support must be provided for at least term of the project.

NFR 6 — The library and each tool or extension shall be documented for developers and end users, using man pages and other common document forms.

NFR 7 — Patches implemented for third party projects shall be documented for developers and end users, using man pages and other common document forms.

NFR 8 — Patches implemented for third party projects shall be contributed and distributed to the appropriate community

NFR 9 — The code and documentation shall be licensed using an open source license.

NFR 10 — Installation scripts and/or instructions shall be made for the library and tools for each target platform, including Linux, Unix and Windows, as well as apache and lighttp installation scripts

3.4. Orthogonal technology

NFR 11 — The WARC Tools shall be implemented as a C library

NFR 12 — Library functionality shall have APIs and dynamic language bindings to be made available as software libraries for developers.

NFR 13 — The functionality of the API shall enable the appropriate functionality in the library scriptable[1] and programmable[2].

NFR 14 — Web server plug-ins shall conform to the Apache 'mod' API and shall operate correctly with both Apache and Lighttp servers and shall be compatible with appropriate web sever programming standards, such as WSGI in Python and the Servlet API in Java.

NFR 15 — The deliverables shall be made available in source code, named as "libwarc-version.tar.gz", where version is the version of the library. A number of binary distributions may be made, including RPM, DEB and EXE.

NFR 16 — The deliverables shall be ported to the following platforms: Linux, FreeBSD, Mac OS X 10.5 and Windows.

NFR 17 — The final WARC API shall be coded in C with minimum dependencies or external libraries (e.g. GZIP). It must not include any assembly code.

NFR 18 — The C library will be developed in such a way as to allow it to be used with the Simplified Wrapper and Interface Generator, or SWIG, (www.swig.org) in order to wrap the C code and allow it to be called natively in a wide variety of languages.

NFR 19 — An interface implementation may be made for Python to allow the library to be used natively by Python programmers requiring no knowledge of the internal implementation of libwarc or C.

NFR 20 — An interface implementation shall be made for Java to allow the library to be used natively by Java programmers requiring no knowledge of the internal implementation of libwarc or C.

NFR 20 — The functionality of the library shall be exposed in such a way as to fit the metaphors and paradigms of the specific implementation languages (C, Java, Python).

[1] Command lines in shell scripts and dynamic language bindings to the library

[2] Dynamic language bindings, Java packages and the C library itself

Discussion

4. Change log

4.1. Changes from v0.1 to v0.2

Small edits to section 2 intro.

Added wording on support to NFR 5.