Repository Working Group	J. Kunze
	California Digital Library
	October 30, 2009

Checkm: a checksum-based manifest format

Abstract

Checkm is a general-purpose text-based file manifest format. Each line of a Checkm manifest is a set of whitespace-separated tokens, the first of which identifies the corresponding digital content by filename or URL. Other tokens identify digest algorithm, checksum, content length, and modification time. Tokens may be left unspecified with '-', the degenerate case being a simple file list. It is up to tools that use the Checkm format to specify any further restrictions on tokens (e.g., allowed defaults and digest algorithms) and on overall manifest completeness and coherence. Checkm is designed to support tools that verify the bit-level integrity of groups of files in support of such things as content fixity, replication, import, and export. A manifest may be single-level or multi-level (hierarchical), the latter being useful, for example, in harvesting material from very large web sites (cf. sitemaps).

---

1. Checkm overview

Checkm (pronounced "check 'em") is a simple text-based manifest format for digital content. A manifest is a set of lines, each of which describes a unit of content via up to six whitespace-separated tokens. The meaning of a token is given by its position within the line. For example, the first three tokens give the name of the content, a checksum algorithm, and a digest (checksum) computed using that algorithm, respectively. Here's a a manifest identifying two files with MD5 checksums.

```
# My first manifest.  Two files total.
# Filename         Algorithm   Digest
book/Chapter9.xml     md5      49afbd86a1ca9f34b677a3f09655eae9
images/r862.png       md5      408ad21d50cef31da4df6d9ed81b01a7
```

Checkm is purely concerned with format and not with such things as completeness and fitness for a given application. So it defines the meanings of the six tokens but does not mandate their use. For example, a file package transfer tool could require use of four tokens, only two of which would be required by a fixity checking tool. The next example is a bare-bones manifest in which all but the first token have been dropped, which is a useful degenerate case when only a list of named units of content is needed.

```
# My second manifest.  Just a list of files.
# Filename          (no other tokens given)
http://example.org/i/chap9.xml
http://example.org/i/chap9fig2.png
```

To leave tokens unspecified that would occur in the middle rather than at the end of a line, give those tokens as '-'. For example, a package transfer application that also renames files might use the following manifest.

```
# My third manifest.
# Filename and Target specified, not Alg, Digest, Length, or ModTime
http://example.org/i/chap9.xml      -   -   -   -  book/Chapter9.xml
http://example.org/i/chap9fig2.png  -   -   -   -  images/r862.png
```

Each non-comment line can contain up to six tokens, and has the form,

```
[@]SourceFileOrURL  Alg  Digest  Length  ModTime  TargetFileOrURL
```

where `[@]` indicates an optional '@' that causes the identified content to be "included" as a manifest extension. In principle there is no upper or lower limit on the number of lines in a Checkm manifest, however, practical considerations may call for extending a single-level manifest to a multi-level manifest.

---

2. Multi-level manifests

If supported, a multi-level manifest permits one large manifest to be spread over a number of smaller manifests. To trigger this, the SourceFileOrURL token that begins a line is preceded by a literal '@'. It invokes a simple inclusion mechanism indicating that the identified content is also in Checkm format and extends the current manifest; this is similar to mainstream sitemap extension mechanisms (cf. **[SITEMAPS]**). A tool can be said to support only single-level Checkm if it does not support multi-level manifests.

Included manifests may themselves recursively include other manifests. There is no limit either to the number of inclusions or to the depth of a multi-level manifest. Cycles in the inclusion graph are generally considered to be in poor taste.

---

3. Checkm lines

The Checkm tokens on a given line all relate to the unit of content identified by the first token on the line. A unit of content is a contiguous sequence of octets (e.g., a file) identified by a filename or URL. Manifest lines end with either LF (hex 0a) or CRLF (hex 0d0a), and lines that begin with '#' are considered "comments" that are to be ignored by processors. Blank lines are also ignored.

Tokens consist of UTF-8 characters **[RFC3629]** separated by linear whitespace (one or more SPACE and/or TAB characters). Any linear whitespace found at the start or end of a line is ignored. Any characters, such as whitespace, not allowed in a URL must be represented using URL percent-encoding **[RFC3986]**.

Tokens may be left unspecified by simply dropping them from the end of the line or by giving them as '-' (hyphen). Checkm is silent about which tokens are required or prohibited and what defaults may be in effect. Checkm is also silent about manifest completeness (which units of content must be included) and hyper-specification (whether one unit of content can or must have more than one line describing it, e.g., resulting from two digest algorithms).

```
[@]SourceFileOrURL  Alg  Digest  Length  ModTime  TargetFileOrURL
TOKEN NUMBER:    1    2       3       4        5                6
```

The token's numbered position determines its meaning, as explained in the correspondingly numbered subsections below.

3.1. [@]SourceFileOrURL: content identifier

The SourceFileOrURL token identifies digital content, and may be given as '-' to leave it unspecified (e.g., the content might be found on Unix "stdin"). This token may be a URL or a relative or absolute filename. To prevent interpretation of a relative pathname that begins with '#' or '@', one can insert "./" in front of the name. Whether this token is a filename or a URL, any characters not allowed in a URL must be represented using URL percent-encoding **[RFC3986]**.

If any SourceFileOrURL token in a manifest is preceded by the optional '@', the line amounts to an "include" statement and the manifest is considered to be "multi-level". Other tokens on that line still relate to the content but the "included" content itself is considered to be an extension of the current manifest. For example, a multi-level Checkm manifest totaling 4 million lines could be represented by a 2000-line manifest, each line of which references a 2000-line single-level manifest.

If none of the lines in a manifest is preceded by '@', the manifest is considered to be "single-level". It is permissible for a tool that conforms to Checkm to declare support for only single-level manifests.

3.2. Alg: algorithm

Alg is either the literal string "dir" (designating a directory), a string specifying a cryptographic checksum algorithm, or '-' to leave it unspecified. The special case of "dir" is useful for listing an empty directory, which has neither a fixed octetstream over which to compute a digest nor a contained filename to imply the directory's existence. For example,

```
# My fourth manifest.  Two files and a directory.
# Filename         Algorithm   Digest
book/Chapter9.xml     md5      49afbd86a1ca9f34b677a3f09655eae9
icons/                dir
images/r862.png       md5      408ad21d50cef31da4df6d9ed81b01a7
```

Implementors of tools that use Checkm are strongly encouraged to support at least two widely implemented checksum algorithms:

"md5" **[RFC1321]**
"sha1" **[RFC3174]**

When using other algorithms, the name of the algorithm should be normalized for use in the manifest's filename, by lowercasing the common name of the algorithm, and removing all non-alphanumeric characters.

3.3. Digest: computed checksum

Digest is a string representing the checksum calculated according to the Alg algorithm over the content, or '-' to leave it unspecified.

3.4. Length of content

Length is the number (base 10) of octets in the identified content, or '-' to leave it unspecified. It is typically useful in providing a rapid test for altered content and for estimating file transfer times.

3.5. ModTime: time last modified

ModTime is a lexically sort-friendly date such as **[TEMPER]** ('YYYYMMDDhhmmss') or **[W3CDTF]** (YYYY-MM-DDThh:mm:ss), or '-' to leave it unspecified. It should represent the UTC time when the content was last modified and is typically useful in incremental or priority harvesting of content (cf. **[OAI]** and **[SITEMAPS]**).

3.6. TargetFileOrURL: other location

TargetFileOrURL is a secondary location for the content that applications would use as necessary. For instance, a transfer tool that also renames files could use this token as the destination name.

4. Conformance Terminology

A tool that uses the Checkm format should document which part of the format it supports. For example, one common restriction could be expressed something like,

> "... which must be a single-level, 3-column Checkm manifest with relative filenames."

This terminology suggests that, for this particular tool, an exception or undefined behavior is the likely result of supplying a Checkm manifest that has any line beginning with '@', a URL, or an absolute pathname, or that has any line with more than or fewer than 3 tokens.

5. Example two-level Checkm manifest

```
# A two-level manifest.
#Filename Alg  Checksum                                 Length
foo.bar   sha1 2eacd0da7aa89b094f5121eb2901bf4de2219ef1    366
foo.bar   md5  3e83471320227c0797a0c251f28db0c5            366
# This next line "includes" the manifest in file "myfirst".
@myfirst  md5  6ab96c8930621d50cef31da4df6d9ed8            264
```

where the included file "myfirst" contains 264 octets and lists two files:

```
# My first manifest.  Two files total.
# Filename         Algorithm   Digest
book/Chapter9.xml     md5      49afbd86a1ca9f34b677a3f09655eae9
images/r862.png       md5      408ad21d50cef31da4df6d9ed81b01a7
```

6. References

[OAI] Lagoze, C. and H. Van de Sompel, "**Open Archives Initiative Protocol for Metadata Harvesting**," June 2002 (**HTML**).

[RFC1321] **Rivest, R.**, "**The MD5 Message-Digest Algorithm**," RFC 1321, April 1992 (**TXT**).

[RFC3174] Eastlake, D. and P. Jones, "**US Secure Hash Algorithm 1 (SHA1)**," RFC 3174, September 2001 (**TXT**).

[RFC3629] Yergeau, F., "**UTF-8, a transformation format of ISO 10646**," STD 63, RFC 3629, November 2003 (**TXT**).

[RFC3986] **Berners-Lee, T.**, **Fielding, R.**, and **L. Masinter**, "**Uniform Resource Identifier (URI): Generic Syntax**," STD 66, RFC 3986, January 2005 (**TXT**, **HTML**, **XML**).

[SITEMAPS] sitemaps.org, "**Sitemaps XML format**," February 2008 (**HTML**).

[TEMPER] Blair, C. and J. Kunze, "**Temporal Enumerated Ranges**," August 2007 (**PDF**).

[W3CDTF] Wolf, M. and C. Wicksteed, "**Date and Time Formats (W3C profile of ISO8601)**" (**HTML**).

Author's Address

John A. Kunze
California Digital Library
415 20th St, 4th Floor
Oakland, CA 94612
US
Fax: +1 510-893-5212
Email: jak@ucop.edu