fhiso

CFPS 104
(Call for Papers Submission number 104)

Proposal to use GEDCOM X's container format

Submitted by: Smith, Richard

Type: A comment on a submitted paper

Comment on: 91

Created: 2014-02-01

URL: Most recent version: http://fhiso.org/files/cfp/cfps104.pdf
This version: http://fhiso.org/files/cfp/cfps104_v1-0.pdf

Description: This paper considers the proposal to ratify GEDCOM X as a FHISO standard, and supports the adoption of GEDCOM X's container file format.

Keywords: GEDCOM X, container files, jar files, MIME types, metadata, Dublin Core, modularity, URIs

Abstract

This paper considers the proposal to ratify GEDCOM X as an FHISO standard and proposes, without prejudice to ratification of the remaining parts of GEDCOM X, that the FHISO adopt GEDCOM X's container file format. The advantages of GEDCOM X's use a new `X-DC-conformsTo` header to denote conformance with a specific standard are discussed with particular reference to their use within a future modular FHISO standard. The paper concludes with an analysis of certain specific technical aspects of the GEDCOM X container format, and makes some suggestions as to how they might be improved.

1 Introduction

In CFPS 91, Ryan Heaton makes "a substantive vision and proposal to validate and ratify the GEDCOM X specification set" [1]. In its current state as a 'stable draft', GEDCOM X is a suite of nine specifications: *(i)* a container file format; *(ii)* a set of standard metadata headers; *(iii)* a conceptual data model; serialisation formats in *(iv)* XML and *(v)* JSON; *(vi)* a date format; and vocabularies for *(vii)* event types, *(viii)* fact types, and *(ix)* name part qualifiers. This paper suggests that the FHISO should consider the various GEDCOM X modules separately on their merits, probably as one of several competing proposals in each area.

If, for certain modules, reconcillation between GEDCOM X and the FHISO is not possible, the FHISO may wish to standardise some but not all of GEDCOM X's modules. Whilst such a difference between GEDCOM X and the FHISO's standard would be unfortunate, it is inevitable that sooner or later a vendor will wish to handle forgo some part of the established standard and handle an aspect of genealogical data in their own way. It would be wise to prepare for this eventuality from the start, and allowing for the possibility that the FHISO will not accept all of GEDCOM X's modules is good preparation for that eventuality.

2 The GEDCOM X container format

The GEDCOM X 'File Format' specification [2] defines a container file format as a means of bundling and compressing files and associating metadata with them. Despite it's name, it does *not* specify the format in which any genealogical data is encoded: that is the subject of other GEDCOM X specifications. The use of a container format such as this was discussed in CFPS 93 [3, §1] with particular reference GEDCOM X's format, although that paper did not go so far as to propose that FHISO adopted one.

This paper commends the GEDCOM X's container format as an excellent example of the reuse of existing technology. The `zip` file format is well established and

widely implemented, and it has been used as a container format in many recent technologies including Java and the two main office application formats, OpenDocument and Office Open XML. Metadata is included in a mandatory manifest file: a simple text file, identical in format to the one used in `jar` files. (The current GEDCOM X draft is perhaps remiss in not acknowledging the `jar` manifest, and neither guaranteeing compatibility with it nor explicitly noting any present or potential incompatibilities with it.) Metadata is encoded using a combination of HTTP headers and Dublin Core metadata terms, and a short list of standard headers are defined in the GEDCOM X 'Standard Header Set' specification [4].

This paper proposes that the FHISO adopts a container file format which is aligned as closely as possible with the GEDCOM X container format. Specifically, it is proposed that the FHISO adopts §1, §2 and §4 of the GEDCOM X 'File Format' specification [2], except that:

— the specification's URI in §1.1 should be left undefined;
— the normative reference to the GEDCOM X XML Serialisation specification should be removed from §1.1; and
— in §4.2, the default `Content-Type` should be left undefined.

This paper also proposes that the whole of the GEDCOM X 'Standard Header Set' [4] be adopted except that:

— the specification's URI in §1.1 should be left undefined;
— the normative reference to the GEDCOM X 'Date Format' specification should be removed from §1.1;
— the type used in §3.3 to represent date fields should be left undefined.

The purpose of these omissions is simply to get an uncontroversial core of GEDCOM X functionality that can be used by the FHISO without introducing dependencies on the other GEDCOM X specifications. In due course, it seems likely and desirable that full alignment can be achieved between the FHISO's use of these modules and their definition by GEDCOM X, even if the FHISO chooses not to ratify the whole of GEDCOM X. Except for the few specific technical concerns discussed in §4, the proposed omissions should not be taken as criticism of or disagreement with the omitted parts of the GEDCOM X specifications.

3 Modularity in GEDCOM X

One appealing feature of GEDCOM X's container format is the `X-DC-conformsTo` manifest header. As its name suggests, this is defined by Dublin Core and is used to record "an established standard to which the described resource conforms" [11]. Standards are referenced by URI, and GEDCOM X makes use of this header is manda-

tory. Presumably GEDCOM X intends that each of its separate specifications (or at lest each of those used in the container) be listed in the header. The current drafts are slightly unclear on this point as none of the examples include the header, and the 'File Format' specification is the only one to state that its URI must be listed in the header.

The GEDCOM X container is a general-purpose format without any customisation for genealogical use. The `X-DC-conformsTo` header could list any standards, genealogical or otherwise. Only by referencing a genealogical standard does the container become a genealogical container, and in principle there is no reason why the same container format cannot be used with several different, perhaps competing genealogical standards.

Sooner or later a vendor will reject some part of whatever standard the FHISO produces, or will introduce a major extension to it. If the FHISO standard is modular, as the GEDCOM X one is, then the `X-DC-conformsTo` header provides a way for an application to specify exactly which modules of the standard it is using, and vendor extensions can also be listed there. Similarly, as the FHISO standard evolves to meet future needs, new modules or major revisions of exist ones may be needed, and their URIs can be listed in this header.

It is because including the specification's URI in the `X-DC-conformsTo` signifies full conformance with that specification that this paper leaves proposes to strike the GEDCOM X URIS. For the reasons already given, this paper proposes initially accepting a slightly reduced version of these two GEDCOM X specifications, and therefore compliance with the reduced FHISO version does not imply full compliance with the GEDCOM X version. It is vehemently hoped that this difference will be of short duration during the early development of the FHISO standard, and that in fullness of time complete harmonisation can be achieved for the 'File Format' and 'Standard Header Set' standards, regardless of the fate of the other GEDCOM X standards.

4 Specific technical concerns

Although this is the longest section of this paper, that fact should not be construed to imply fundamental problems exist in the the two GEDCOM X specifications considered here. These concerns are with certain specific technical details of the specifications, and are concerns that the GEDCOM X community may wish to consider irrespective of whether the FHISO chooses to ratify GEDCOM X.

4.1 Relative URIs in containers

The proposal above omitted the whole of §5 of the 'File Format' specification which defined how URIs, particularly relative ones, were to resolved for documents in containers. This was primarily to avoid a dependency on the XML serialisation of GEDCOM X.

However, there was a second reason for it. According to §5.3 of the GEDCOM X 'File Format' specification:

> Per RFC 3986 Section 5.1, relative references are only usable when a base URI is known. For the purposes of resolving relative references within a GEDCOM X file, the base URI is defined as the URI to the directory at the root of the `zip` file. The base URI of a GEDCOM X file aligns with the concept of the "Base URI from the Encapsulating Entity" as defined by RFC 3986 Section 5.1.2.

That is, according to GEDCOM X, if a `zip` file contains two files, `a/a.xml` and `b/b.xml`, the former can link to the latter by using `b/b.xml`, a link relative to the root of the `zip` archive. This is somewhat counter-intuitive and contrary to what is done with some other `zip`-based container formats, such as Java's `jar` format. In a `jar` archive, the correct way of referencing `b/b.xml` from `a/a.xml` is with `../b/b.xml`, a link relative to the source document [6, §4.3.1.1].

The GEDCOM X specification references RFC 3986 [7, §5.1] to justify its handling of relative URIs. Relative URIs are resolved relative to some base URI, and that RFC says:

> If no base URI is embedded, the base URI is defined by the representation's retrieval context. For a document that is enclosed within another entity, such as a message or archive, the retrieval context is that entity. Thus, the default base URI of a representation is the base URI of the entity in which the representation is encapsulated.

This says that when a base URI is not embedded in the document (e.g. with an `xml:base` attribute), it is *defined by* the enclosing entity (i.e. the `zip` file); and if it is not defined by enclosing entity, then the base URI of that enclosing entity is used. In the case of a `zip` file, every document in the archive must have a file name, which may include a path. That file name is the base URI of the document, and because the archive contains a specific base URI for each document it contains, the fall-back case of "the base URI of the entity in which in the representation is encapsulated" is never required.

Exactly this behaviour can be seen in the resolution of relative URIs in multipart MIME documents, described, for example, in RFC 2557 [8]. A `Content-Location` header is used to specify the filename, including a path, and URIs are resolved

relative to that. The HTTP 1.1 RFC confirms that this header defines the document's base URI [9, §14.14]. A `zip` file should probably be considered analogously to a multipart MIME document, and each document contained in it should behave as if it had a `Content-Location` header containing its file name and path. This results in the behaviour used in `jar` archives.

4.2 Absolute URIs to documents within containers

The GEDCOM X 'File Format' specification does not provide a syntax for a URI to a specific document within a container. The `jar:` URI scheme does precisely this for a `jar` archive, and the provisional IANA registration for the scheme states that it "works for any ZIP based file" [10]. An example `jar:` URI is:

```
jar:http://example.com/genealogy.gedx!/tree/smith.xml#A34
```

The part before the `!/` is a URL from which the container can be fetched, while the part after it is he path within the archive. This paper proposes that, where absolute URIs are required to entities inside an archive, they should be `jar:` URIs of this form. They are likely to be particularly useful in providing a unverisally-unique identifier for a person or other entity in a genealogical document. In the example above, the identifier `A43` might be used to identify a particular individual in that data file, and if so, the complete URI, including fragment identifier, provides a unique way of representing that individual.

4.3 Dates in manifest headers

Two of GEDCOM X's standard headers have values that are dates. They are defined in terms of the `created` and `modified` terms defined in Dublin Core's metadata terms [11]. Dublin Core does not define the syntax to be used in expressing dates, and GEDCOM X wisely defines the syntax it requires for these properties. This is unlikely to cause a conflict if Dublin Core do define a date syntax in the future as when such changes have been made previously, Dublin Core have created a new version of their namespace for the purpose.

Nevertheless, the XML Schema `date` and `dateTime` types have become the *de facto* standard for representing the Dublin Core date fields. This can be seen from the examples in the Dublic Core specification, and in the many technologies to have used it. As discussed in CFPS 103 [12], the GEDCOM X date type is subtly incompatible with XML Schema's [13, §3.2.7]. It seems unfortunate to adopt the Dublin Core dates only to deviate from standard practice in their use, even if this is permitted by the Dublin Core specification.

This paper suggests that the trivial timestamp format presented in CFPS 103 might be an acceptable alternative for representing the Dublin Core date fields. It is compatible with XML Schema's `dateTime` and thus with the normal use of Dublin Core. It also disallows form such as recurring or approximate dates that GEDCOM X permits (possibly unintentionally) in the `X-DC-created` and `X-DC-modified` headers.

4.4 Multi-valued manifest headers

The GEDCOM X 'Standard Header Set' specification says [4, §3]:

> Each header *may* supply multiple values. This is done either in a single name-value pair by separating each value by a comma *or* in multiple headers of the same name.

The decision to say this is no doubt influenced by similar language in the HTTP RFC which says [9, §4.2]:

> Multiple message-header fields with the same field-name MAY be present in a message if and only if the entire field-value for that header field is defined as a comma-separated list. It MUST be possible to combine the multiple header fields into one "field-name: field-value" pair, without changing the semantics of the message, by appending each subsequent field-value to the first, each separated by a comma.

There is an important difference here. In HTTP, multiple headers may be present and may be merged only where a list of values is expected; but GEDCOM permits all headers to be merged. This seems wrong as many of the headers are inherently single-valued. The `Content-Length` header is such an example. Indeed, of the HTTP headers listed in GEDCOM X's 'Standard Header Set', only `Content-Encoding` and `Content-Language` are allowed to contain multiple values in an HTTP header. Ideally GEDCOM X should follow the HTTP RFC on this matter.

Where multi-valued headers are permitted, the GEDCOM X specification is ambiguous over their interpretation: is a comma a separator between values, or is it part of the value? This ambiguity does not arise with HTTP headers because HTTP defines the syntax of the value, and either commas are not allowed or they must be quoted. This is not the case, however, for the non-HTTP headers that GEDCOM X defines from Dublin Core. The `X-DC-creator` header's value is URI, and URIS may legitimately contain commas. (The rules on when a comma must be escaped as `%2C` in a URI are somewhat arcane [7], but escaping is not required in many common cases.) The problem is compounded by the fact that HTTP makes it optional to put whitespace after a comma separating multiple values.

The current HTTP RFC does not define any multi-valued headers that have URIS as their values, so it provides no guide. Nevertheless, multi-valued URI headers

do exist in HTTP, and the `Link` header is a good example. It was present in the earlier HTTP RFCs but later moved to separate document [14]. The URI in a `Link` header is delimited by angle brackets: viz. `<http://example.com>`. This paper proposes that the two GEDCOM X URI headers are formatted similarly. If required for compatibility, the delimiters could be made optional when a single value is present, but it is then no longer possible for an application to join identically-named headers wih a comma without reference to the header's format.

4.5 Improving GEDCOM X modularity

Although GEDCOM X is specified as a collection of separate components with well-defined dependencies between them, those dependencies are not as minimal as they could be. Two such unnecessary dependencies that are relevant to this paper are the 'File Format' specification's dependencies on the 'XML Serialization Format' and 'Conceptual Model' specifications (the latter dependency being unacknowledged). This paper encourages GEDCOM X to remove them. These dependencies largely arise from §3.1 of the 'File Format' specification, which requires that:

- "all resources defined by the GEDCOM X Conceptual Model MUST be represented using the GEDCOM X XML media type as defined by the GEDCOM X XML Format specification;" and
- every container must contain at least one such XML file.

This paper suggests that both of these requirements are out of place. In the latter case, there is no good reason why a container might not contain just a colllection of images or other media, together with the metadata encoded in the manifest file. In some cases, there may be good reason to want an XML file that contain a more detailed description of the resources, but it seems unreasonable to require one even if no more detailed description is available.

The reason the first requirement is out of place is that it is almost meaningless. Given genealogical data in another format, say GEDCOM 5.5 or some proprietory format, does it conform to the GEDCOM X data model? It impossible to answer in the afirmative unless the format's specification explicitly says so, and only one specification currently does so. As a result, this requirement boils down to a prohibition on the use of GEDCOM X JSON files in the container. If such a proscription is desirable, it would be more usefully expressed in the JSON specification.

If the intention was to prohibit other, non-GEDCOM X formats, this paper believes such a restriction is inappropriate. If a researcher has been supplied data in another format by colaborator, it is entirely reasonable that they may wish to store the original data file unaltered for future reference, and it seems entirely reason-

able that they should placed in a container.

4.6 The default `Content-Type`

The 'File Format' specification makes the default value of the `Content-Type` header '`application/x-gedcomx-v1+xml`'. This paper opposes that choice on two grounds: one technical and one logistical. The technical grounds are that the `Content-Type` header already has two different defaults, depending on context. In a MIME message, the default is '`text/plain; charset=us-ascii`' [15, §5.2]; and in HTTP, the default is '`application/octet-stream`' [9, §7.2.1]. This difference is unfortunate, but it seems unhelpful to compound it by adding a third. As GEDCOM X defers to HTTP in defining many of its headers, if default is needed, it would perhaps be best to fall back on HTTP's default. But this paper suggests a better solution is to remove the question by making the header mandatory.

The logistic reason for wanting to avoid GEDCOM X's default is that it ties the container format, not only to a specific serialisation format, but to a specific version of it. Assuming a version 2 of the XML serialisation is released, once it become commonplace, the current default will be hinderance. A new version of the container format could also be released, differing perhaps only in its choice of default `Content-Type`, but that seems unnecessary and undesirable. Why tie the container specification's release cycle to that of a specific serialisation? And why break compatibility with tools that only understand the container (for example, a tool to merge containers) simply to update the default `Content-Type`?

If it is anticipated that a container will contain many small XML files, the repetition of their `Content-Type` can be eliminated by adding a `Default-Content-Type` header to main header section of the manifest, and only allowing per-resource `Content-Types` to be missing when the container specifies a default.

References

[1] Ryan Heaton, 2013, *A substantive vision and proposal to validate and ratify the GEDCOM X specification set*, `http://fhiso.org/files/cfp/cfps91.pdf`

[2] Intellectual Reserve Inc., 2013, *The GEDCOM X File Format*, `http://gedcomx.org/file/v1`

[3] Richard Smith, 2013, *Requirements for storing media objects* (CFPS 93), `http://fhiso.org/files/cfp/cfps93.pdf`

[4] Intellectual Reserve Inc., 2013, *The GEDCOM X Standard Header Set*, `http://gedcomx.org/headers/v1`

[5] Intellectual Reserve Inc., 2013, *The GEDCOM X Date Format*, `http://gedcomx.org/date/v1`

[6] Sun Microsystems Inc., 2004, *JavaHelp 2.0 System User's Guide*, `http://download.java.net/javadesktop/javahelp/jhug.pdf`

[7] T. Berners-Lee, R. Fielding & L. Masinter, 2005, *Uniform Resource Identifier (URI): Generic Syntax* (RFC 3986), `https://www.ietf.org/rfc/rfc3986.txt`

[8] J. Palme, A. Hopmann & N. Shelness, 1999, *MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)* (RFC 2557), `https://www.ietf.org/rfc/rfc2557.txt`

[9] R. Fielding, *et al.*, 1999, *Hypertext Transfer Protocol — HTTP/1.1* (RFC 2616), `https://www.ietf.org/rfc/rfc2616.txt`

[10] Dave Thaler, 2012, *Resource Identifier (RI) Scheme name: `jar`*, `https://www.iana.org/assignments/uri-schemes/prov/jar`

[11] Dublin Core Metadata Initiative, 2012, *DCMI Metadata Terms*, `http://dublincore.org/documents/dcmi-tersm/`

[12] Richard Smith, 2014, *A trivial timestamp type* (CFPS 103), `http://fhiso.org/files/cfp/cfps103.pdf`

[13] World Wide Web Consortium, 2004, *XML Schema Part 2: Datatypes (Second Edition)*, `http://www.w3.org/TR/xmlschema-2/`

[14] M. Nottingham, 2008, *Web Linking* (RFC 5988), `https://www.ietf.org/rfc/rfc5988.txt`

[15] N. Freed & N. Borenstein, 1996, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* (RFC 2045), `https://www.ietf.org/rfc/rfc2045.txt`