DBL Metadata v2.3 – RFC v0.31

February 15th 2019 by Mark Howe, for the Digital Bible Library

Contents

1 Overview

1.1 Goals

1.1.1 Provide timely support within DBL for new features

- USX3
- Audio timing files
- Biblica TTT
- Diglot print entries

1.1.2 Begin implementing major Scripture Burrito features

- Multi-ecosystem support via “namespaced” IDs
- Entry subtypes via conventions
- Orthogonal treatment of IDs
- Rich support for “public” licenses

1.1.3 Smaller schema tweaks and fixes

1.2 Constraints

- Deliver by the end of May 2019
- Maximize new functionality
- Minimize disruption within DBL
- Minimize disruption for DBL partners

1.3 Process to date

- Incremental progress from the first unified DBL Schema (2.0) through two rounds of fixes and extensions, in partnership with DBL’s major IPCs and LCHs.
- Discussion at the ETEN Bible Tools Portability Working Group during the second half of 2018, including a detailed discussion paper in November 2019
- Detailed discussion with Jeff Klassen and Sean Morrison of DBL in December 2019, notably at a meeting in Swindon, UK.

1.4 Outstanding issues

Many of the proposed 2.3 changes have been discussed quite extensively, and many of those changes stem from concrete needs within DBL. The following areas require further work:

1.4.1 Progress tracking

DBL currently has two ways to do this. PT uses the less flexible option. The PT Reg devs were surprised to learn that DBL had that information at all so, presumably, progress.bible is getting that information from somewhere else? We should decide on a long-term strategy, with the PT ecosystem, and implement it in v2.3.

1.4.2 Convergence of metadata with PT Reg, REAP and progress.bible

The aim here is to fix a number of low-level differences between how data is stored, levels of confidentiality being one important example.

1.4.3 Enhanced resource support

This discussion encompasses two issues:

1. Fixing the current non-standard and clumsy handling of non-canonical PT resources such as translation handbooks.
2. Providing a mechanism for storing a wide range of para-biblical content such as lexicons, glossaries, concordances and translation notes.

The preferred solution is to address the PT resource issue as part of an initial mechanism for resources in general. If this cannot happen in a timely fashion, DBL will need to find a short-term solution for v2.3. (Both options will involve breaking changes for Paratext, and the first option implies two sets of breaking changes.)

1.4.4 Public license support

At present, “open access” entries within DBL do not have a license. We need a way to handle the wide range of Creative Common-like and Open-Source-like licenses that exist. If this cannot happen it will need to wait for Scripture Burrito 0.1.

1.4.5 Consultation about braille

DBL’s evolving braille support has been developed with Compass Braille, who produce most of the braille editions of UBS texts. Other organizations do similar work, with different technology. It would therefore be good to review braille support to make sure that it can be used by a wide range of braille producers.

1.4.6 Consultation about sign language video

DBL has developed an approach that has worked for a small number of entries. Since the Deaf Bible Society is working with multiple sign language teams across the world, and in the light of Project Hummingbird, it would be sensible to compare metadata schemes with a view to harmonizing fields whenever possible.

1.4.7 Canonical Javascript expression of DBL Metadata

This is a longstanding request from api.bible, and would be useful to other DBL clients such as Nathanael. We could also aim for convergence with PT Reg metadata, and begin to replace the multiple, arbitrary JSON formats returned by different parts of DBL with something based on the same canonical format.

1.5 Roadmap

1.5.1 End February ’19

- Initial approval of changes by DBL

1.5.2 End March ’19

- Consultation with PT ecosystem, REAP/progress.bible, YouVersion and api.bible
- Decisions about progress tracking

1.5.3 End April ’19

- Integration of public license and/or enhanced resource support
- Consultation about braille and video with Compass, DBS and others
- Finalization of DBL Metadata JSON format

1.5.4 End May ’19[1]

- Final discussions in Dallas, publication of final schemas and documentation

1.5.5 End June ’19[2]

- DBL Migration/Support to v2.3

2 “Breaking changes”

In the general case, all changes to an XML schema are potentially breaking, since the schema acts as a contract between producer and consumer that specifies what can safely be assumed about the document. A major benefit of schema is that developers do not need to write defensive code for cases excluded by the schema contract. It follows that any changes can potentially lead to surprises. Specifically:

- **Relaxing** constraints is a potential problem for consumers, who may find that values fall outside of expected ranges, or that switches based on element names do not handle unexpected content.
- **Tightening** constraints is a potential problem for creators, who may find that the output of their toolchains no longer leads to a valid document.
- **Changing** structure may mean that data is not found, or that the unexpected elements trigger exceptions.

Whether any of these problems will actually occur is harder to predict, since most XML processing models skip aspects of the document. This is why identifying issues is something that, to a large extent, needs to be done by the creators and consumers.

Strategies to limit the pain of transitioning to v2.3 include

- making most new elements optional
- defining default behaviour that corresponds with v2.2 behaviour
- restricting changes to the scope of specific media where possible
- keeping document structure changes as simple as possible

3 Namespaced IDs

Right now, DBL metadata assumes that uids resolve to records on the DBL server. Other parts of the Bible ecosystem do something similar with respect to other servers. For projects to be portable across ecosystems, we need to know how to resolve uids against one or more server.

1 Depending on final date of meeting, to be hosted by the Deaf Bible Society with the participation of sign language partners, as well as REAP/Progress.Bible.
2 Maybe depending on the date of the next DBL Summit (typically in Europe around June).

3.1 Use cases

3.1.1 Entry ID

DBL entries are 16-character hex strings derived from the underlying Mercurial ID.

3.1.2 User ID

DBL does not currently expose user IDs within metadata documents, but doing this would facilitate identification of users across systems, notably in the archiveStatus section.

3.1.3 Agency ID

This is a 16-character hex string, used in the agencies section.

3.1.4 License ID

This is currently either a positive integer, “owned” or “open-access”.

3.2 Declarations

Server declarations would look something like this:

```
<idServer prefix="dbl">https://thedigitalbiblelibrary.org</idServer>
<idServer default="true">http://atlantisbibleconsortium.net</idServer>
<idServer prefix="myServer" local="true">http://localhost::8080</idServer>
```

The URI is the portable identifier (following a popular Internet convention, although the schema definition of a URI tends to be more or less “a string”) which, eventually, will also provide a path to various server functionality via a standard API.

The “prefix” attribute (lower-case letters) provides a convenient label for referring to that identifier within the document. The “default” attribute denotes the identifier associated with unqualified IDs within the document. The “local” attribute can be used to label identifiers that are not portable (which may be useful for internal testing and for private ecosystems).

Server declarations are optional. If there are no declarations, only unqualified IDs are permitted, and those IDs are assumed to refer to DBL. (This default behaviour means that existing DBL metadata documents should “do the right thing” when processed.)

IdServer elements, when present, should be the first children of the DBLMetadata root element.

3.3 ID syntax

This needs discussion with major, existing ecosystems, but the initial proposal is

```
<prefix>::<id>
```

where

- “prefix” is an NCName[3],
- “id” is a string matching

  ```
  [0-9A-Za-z]([0-9A-Za-z_-]{0,30}[0-9A-Za-z])?
  ```

3 “Non-Colon Name” in XML-speak or, roughly, a sensible name for a variable in most languages.

It is conceivable that a server might use the same id value for different categories of id (eg a user 27 as well as an organization 27). Context within the metadata document should disambiguate these cases. (The alternative would be something like “dbl-license-1” which seems unwieldy.)

Note that DBL does not need to accept entries with ids that do not conform to its current format: ids for any particular server must be constructed to match the constraints of that server. (A discovery mechanism would obviously be useful here.)

3.4 Relax revision format

DBL currently uses auto-increment integers, which are a good fit for publishing (eg “revision 6” makes sense to people in the publishing world.) There is no underlying version control mechanism.

The PT send and receive server, and many other servers in the Bible ecosystem, use a version control system such as Git or Mercurial. These systems use long strings to identify commits, which play a similar role to revisions within DBL.

It seems likely that both systems will continue for the foreseeable future, so Scripture Burrito needs to be handle a range of revision identifiers. The id regex above seems like a good starting point.

As with entry ids, no server including DBL is required to accept entries with revision ids that do not conform to its current format.

3.5 Migration

The plan is for documents with no idServer elements and no qualified IDs to validate, and to have the same semantics as v2.2 documents.

DBL will need an additional schema to check the syntax of entry ids and revisions. DBL’s schema pipeline should be able to handle this.

Eventually, DBL and other servers will need to unpick the server prefix logic in order to resolve IDs, but this does not need to happen before the release of v2.3. It will become an issue with SB 0.1.

4 Identification

4.1 Multiple *Local

See the discussion of multiple language support.

4.2 systemId

4.2.1 Add DBL type

Currently, the DBL system id appears as an attribute in the root element. In a portable world, DBL uids might appear in a document within some other ecosystem. To represent this, we need a systemId for DBL. Also, it may be that SB 0.1 moves to an orthogonal representation where all systemIds are stored at the same level and none are stored in the root element. A DBL systemId would look like this:

```
<systemId type="dbl">
      <id>482ddd53705278cc</id>
      <revision>2</revision>
</systemId>
```

4.2.2 x-* types

The systemId type mechanism was created when DBL needed to work with a small number of large ecosystems. Future ecosystems may be small – maybe a national denomination or even one church. It may not always make sense to add such organizations to the schema and, when it does, this will take some time. Some architectures involve local servers (on a VPN, an intranet or even localhost), and testing sometimes requires server changes. Supporting types matching

```
x-[a-z]{1,}
```

provides a way to introduce new or private ecosystems without rewriting schema:

```
<idServer prefix="mvah">https://markspersonaltranslationproject.fr</idServer>
...
<systemId type="x-mvah">
      <id>idInMyPersonalFormat</id>[4]
      <myDetail>something-that-interests-me</myDetail>
</systemId>
```

The type of all x-* systemIds should correspond to an idServer declaration.

4.3 basedOn

This would uniquely identify the revision on which the entry is based, which might be from a different ecosystem. This information is potentially useful for forensics. It also provides a mechanism for 3-way diffing of documents when the two deltas are from different ecosystems.

```
<basedOn type="dbl">
      <id>482ddd53705278cc</id>
      <revision>1</revision>
</basedOn>
```

4.4 Migration

Adding new systemId types should not be disruptive unless processors are using enums to store that information. The basedOn element would be ignored by most processing models.

5 Agencies

5.1 Multiple *Local

See the discussion of multiple language support.

5.2 Less compulsory fields for upload variant

DBL metadata is used as the basis of job requests which, in effect, provide a contract between client and server about what will be uploaded and how it should be tagged. Some fields are set by the server, but some of those fields are currently required by the schema. The result is that clients need to provide values that are not used, but which will cause an upload to fail if they are incorrect.

4 The required fields for systemIds vary and, for a field type that is not well-known, there is no way to guess what those field might be. I suggest limiting children of systemIds to a key/value representation, which at least prevents anyone from turning systemId into an open-ended storage solution. Obviously, to be useful, a processor needs to be able to get details about ecosystem expectations, and this is one of many reasons why portability is meaningless without discoverability through convergent APIs.

An example of a field that is already optional is the revision id, since the server sets this. (I think this would also be true of most VC systems.)

The major pain point is agencies information. DBL does not allow the rightsHolder of an entry to be changed via a client upload, so the uid of the rightsHolder needs to match. Other, denormalized information about the rightsHolder is currently required, but is not validated so, eg, it's possible to end up with a rightsHolder with a uid for UBS and every human-readable field for SIL.

The proposal is that, for uploading, only the uids of agencies should be required.

5.3 contributor/content should be optional

This is currently the only boolean required for contributing agencies, and looks like a schema typo.

5.4 Migration

Loosening the above requirements is unlikely to break anything.

6 Type

6.1 License

6.1.1 "Private" licenses

DBL has a rich model for "private" licenses. In addition to determining whether an entry can be downloaded by a given Library Card Holder, they are also used to filter information such as as publications, cross-references and footnotes. It would be useful to be able to tell if the metadata document has been filtered.

```
<license type="dbl">
      <id>326</id>
      <uri>https://app.thedigitalbiblelibrary.org/api/license/326</uri>
      <serviceOptions>
            <option type="footnotes">false</option>
            <option type="publications">protestant catholic</option>
      </serviceOptions>
</license>
```

6.1.2 "Public" licenses

First, some DBL-centric context from Sean Morrison:

> I wanted to comment on this, so the terms used are at least commonly understood.
>
> DBL has a notion of licenses, as well as license agreements. Licenses describe terms of use, a license agreement is an agreement between two parties for the use of a text under the terms of a license. Open-access describes a situation where the text is freely available (i.e. anonymous download), independent of the license.
>
> The unfortunate situation is that DBL controls access to texts via the license agreement mechanism (e.g. if a license agreement is in effect, a member of the receiving organization can download the text as described by the agreement). Early on we wanted to allow the anonymous download of texts, so we implemented a feature whereby a copyright holder could make the text available and anyone could download it. We did this by modelling it as a license agreement without termination date.

Making texts freely available for download and defining the license under which they can be used are really separate concerns that were unfortunately conflated.

DBL should handle “private” licenses better, and it is essential for the wider Bible ecosystem to support a rich model of licensing that can handle both bilateral, commercial licenses and Creative Commons/Open Source licenses, as well as public domain content.

DBL could consider including a suitable and timely proposal from the SB Working Group into DBL Metadata 2.3.

6.2 Tristate REAP-compatible replacement for isConfidential

DBL metadata has a flag called isConfidential, where a true value means “life-changing consequences may follow for third parties if we get this wrong”. REAP has a 3 or 4-level way of encoding confidentiality. It would make sense for DBL to adopt REAP’s approach.

6.3 Migration

Adding a license element is unlikely to break anything.

Adding rich public license support ought not to break anything, although it’s hard to say without a concrete proposal.

Reworking the isConfidential flag requires really good communication with all partners to ensure that the new flag is interpreted correctly.

7 Format

7.1 Conventions

7.1.1 Semantics

A convention is a way to optionally label subtypes within the broad medium types supported by DBL metadata. The intention is to provide

- increased content flexibility in an orderly way
- tighter server-side checking of uploads
- better visibility for content variation for consumers.

Servers should check that uploaded entries comply with declared conventions. Consumers may assume that this is the case. The semantics of no conventions are *“caveat emptor”*, ie there are no guarantees about content.

7.1.2 Syntax

```
<convention type="structure" version="1.0">usx_dirs</convention>
```

where “type” is one of “structure”, “content” or “content-format”. Convention elements would be optional children of the “format” element.

7.1.3 Convention enums

These require careful conversations with partners. An initial list for text entries might include

- **usx-refs** (ie “USX includes machine-generated references that PT only includes when it can parse cross-references, and which may not be attempted by some USX creators.)
- **usx-dirs** (ie the USX_1, USX_2 etc convention currently used by PT, but which has been slated for replacement for years since it leads to data duplication

- **versed-paras**, replacing the current text format field, denoting that the processor should treat each verse as a new paragraph.

x-* convention enums should also be supported.

7.2 Text Format

The text medium is implicitly the “something like what PT does” medium, ie there are other character-based media which are not considered to be text.

7.2.1 Remove versedParagraphs

This field was intended to describe a convention, and can thus be represented using the new convention scheme.

7.2.2 Syntax

We need a way to provide the USX version, and maybe to say whether the content is in USX or USFM[5].

```
<syntax>usx</syntax>
<syntaxVersion>3.0</syntaxVersion>
```

7.3 Audio Format

7.3.1 Conventions

- whole-chapter
- book-dirs

7.3.2 Roles for non-canonical audio files

- introduction

7.3.3 Role for timing files

- audio-timing

7.4 Video Format

7.4.1 Conventions

- whole-chapter
- book-dirs
- roles-in-uris (the Nathanael video wizard convention for naming imported files so that their canonical roles may be derived)

7.4.2 Roles for non-canonical video files

- bible-menu
- book-menu
- frontmatter
- backmatter

5 USFM versioning is a strange concept, since the version is not marked anywhere in a USFM document, and a document may become 3.x when the user adds one tag, and revert to 2.x when the tag is removed. This seems to me to be another argument in favour of using USX where portability is required.

- copyright
- book-introduction

7.5 Print Format

7.5.1 TTT convention

Biblica has developed a typesetting toolchain around their Tagged Text Toolbox (TTT). It would like to share content in this format, which is essentially InDesign XML. So

- tagged-text-toolbox

7.5.2 Roles

- printBody
- printCover
- printThumbnail

7.5.3 Metadata for thumbnail JPEG

Thumbnail images (for PoD catalogs) are a recent addition to DBL. It would be desirable to provide some metadata about these images, for example

- width
- height
- color model

7.5.4 Enforce exactly 1 publication

Multiple publications are possible with text, audio and video but are not considered appropriate for print and braille, which are strictly expressions in their own right.

7.5.5 fonts element should be optional (and never empty)

This is a schema error, and also an inconsistency compared with the rest of DBL Metadata, where many elements are optional but container elements should not be empty. So

```
element fonts { printFormatFontElement* }
```
should become

```
element fonts { printFormatFontElement+ }?
```

7.6 Braille Format

7.6.1 liblouis => brailleConvertor

Liblouis is an extremely widely-used library for braille transliteration, but it is not universally used. The proposal is to change

```
<liblouis>3.7.0</liblouis>
```
to

```
<brailleConvertor>liblouis-3.7.0</brailleConvertor>
```

7.6.2 table/source => src

This is just a consistency thing.

7.6.3 Enforce exactly 1 publication

Multiple publications are possible with text, audio and video but are not considered appropriate for print and braille, which are strictly expressions in their own right.

7.7 Migration

Adding conventions is unlikely to break anything, but introducing content that does not follow previous implicit conventions may be breaking.

Removing versedParagraphs is unlikely to be breaking since PT never provided a way to use this flag.

Adding the syntax section should not break anything, but introducing USX3, let alone USFM, is very likely to break things downstream.

The introduction of TTT print entries could break Publishing Assistant, if PA could import from DBL, which, apparently, it can't.

Enforcing one publication for print and braille shouldn't break anything since this has been best practice from the start.

Changes for braille currently only need to work for Compass Braille, so they should be easy to manage.

8 Language(s)

8.1 BCP-47[6]

This is today's preferred language-labelling system. DBL Metadata should probably support this[7]. Given that DBL already supports a lot of language labels, it might be time to decide how many of them are still needed.

8.2 Multiple language support

8.2.1 Languages element

This would handle languages in a similar way to countries, ie a wrapper element plus one or more language elements.

If multiple languages are specified, exactly one of them should be marked as the default:

```
<default>true</default>
```

8.2.2 *Local

In entries that use multiple languages, it may be necessary to provide multiple localizations. In most places, localized names end with "Local". In all these cases, multiple elements should be allowed and, when more than one language element exists in the languages section, the language code should be provided too:

```
<nameLocal bcp="eng-GB">Hello!</nameLocal>
<nameLocal bcp="eng-US">Howdy!</nameLocal>
<nameLocal bcp="fra-FR">Bonjour !</nameLocal>
```

6 https://tools.ietf.org/html/bcp47

7 There are some interesting challenges. For example, it is clear that "eng" != "fra". But what about "eng" and "eng-US"? Are those two different languages or just two levels of obsession? That kind of comparison is probably baked into a lot of places.

8.3 Migration

Putting the language element under a languages element is likely to be a breaking change. Multiple *Local may also produce unexpected from, eg

```
/DBLMetadata/identification/nameLocal/text()
```

One short-term strategy would be to not process entries with multiple language elements.

9 Countries

9.1 Multiple *Local

See the discussion of multiple language support.

10 Names

10.1 Multi-language support

Book names (and other names) may be available in multiple languages, and it may be appropriate to link multiple names to one node in the publication hierarchy. The proposed solution follows the *Local approach:

```
<name id="book-jhn">
      <abbr>Jn</abbr>
      <short bcp="eng">John</short>
      <short bcp="fr">Jean</short>
</name>
```

10.2 Migration

Multiple elements in the case of multiple languages may well be a breaking change which can be mitigated by not processing multiple-language entries in the short term.

11 Manifest

11.1 Tighten checksum regex

The schema currently allows checksums in the S3 partial upload format. This happened to accommodate such checksums during migration, but it turns out that those checksums were wrong. The checksum regex should validate MD5s only.

11.2 Progress within manifest?

See the discussion under the progress section.

11.3 Migration

The checksum change should only invalidate documents containing invalid checksums, which could be considered to be A Good Thing.

12 Publication

12.1 Multiple *Local

See the discussion of multiple language support.

12.2 metaContent

The content elements of the publications section is where data sources, names and roles meet. This currently works well for the content itself. However, some data related to publishable content needs to be given a role too. The first example of this is audio timing files. The proposed solution is to allow metaContent children of content or division elements that relate such data sources to publishable content:

```
<content src="MAT.usx" name="book-mat" role="MAT">
      <metaContent src="timing/MAT.xml"/>
</content>
<content src="MRK.usx" name="book-mrk" role="MRK">
      <metaContent src="timing/MRK_1-6.xml" role="MRK 1-6"/>
      <metaContent src="timing/MRK_7-16.xml" role="MRK 7-16"/>
</content>
```

In the second example above, the metaContent has a role within the scope of the containing content element.

12.3 peripheral ids in metadata role enum

This enables the tagging of extra-canonical content without relying on well-known file names. The list, from the USFM 3 spec[8], would be

Role	**Meaning**
abbreviations	Table of abbreviations
alphacontents	Alphanumeric Contents
chron	Chronology
cnc	Concordance
contents	Table of Contents
cover	Cover
foreword	Foreword
glo	Glossary
halftitle	Half Title Page
imprimatur	Imprimatur
intbible	Introduction to the Bible
intdc	Deuterocanon Introduction
intepistles	Introduction to Epistles
intgospels	Introduction to Gospels
inthist	Introduction to History
intnt	Introduction to New Testament
intot	Introduction to the Old Testament
intpent	Introduction to the Pentateuch
intpoetry	Introduction to Poetry
intprophesy	Introduction to Prophecy

8 https://ubsicap.github.io/usfm/peripherals/index.html

Role	Meaning
lxxquotes	Quotes from LXX in NT
maps	Map Index
measures	Weights and Measures
ndx	Names Index
preface	Preface
promo	Promotional Page
pubdata	Publication Data
spine	Spine
tdx	Topical Index
title	Title Page

12.4 Migration

The most likely issue is unexpected metacontent children, although many processing models would ignore these.

13 Copyright

13.1 Language attribute for statementContent

As with the *Local discussion above, a multi-language entry may require copyright statements in multiple languages, via bcp attributes:

```
<copyright>
    <fullStatement>
      <statementContent type="xhtml" bcp="eng">
        <p>&#169; 2017 Bridge Communications Systems.</p>
      </statementContent>
      <statementContent type="xhtml" bcp="fra">
        <p>&#169; 2017 Bridge Communications Systems.</p>
      </statementContent>
    </fullStatement>
  </copyright>
```

13.2 Migration

As with *Local, multiple statementContent elements have the potential to break accessors via xpath-like technology.

14 Promotion

14.1 Replace promoVersionInfo with promotionContent

The idea here is to make the copyright and promotion sections more similar:

```
<promotion>
    <promotionContent type="xhtml" bcp="eng">
        <h1>Great Choice Dude!</h1>
```

```
    </promotionContent>
    <promotionContent type="xhtml" bcp="fra">
        <h1>Excellent choix, mec !</h1>
    </promotionContent>
</promotion>
```

14.2 Migration

Changing element and attribute names is likely to be breaking.

15 Progress

DBL Metadata currently supports two mechanisms for tracking translation progress. The first, which is supported by PT, uses a top-level section to list the progress for each book:

```
<progress>
      <book code="GEN" stage="4"/>
      <book code="EXO" stage="1"/>
      <book code="JOS" stage="2"/>
      <book code="LUK" stage="4"/>
  </progress>
```

There are two issues with this:

- It means another potentially long list in the metadata
- More importantly, it can only record progress for books when, in reality, progress on introductions and other para-canonical content may also be important.

The alternative mechanism, which has probably never been used, is to record the progress against manifest entries.

```
<resource checksum="0e6c24ebcf1ca2e928578ab239b69687" mimeType="application/xml"
size="296803" uri="release/USX_2/1CH.usx" progress=”37”/>
```

Progress can therefore be logged against any document in the entry, without bloating the metadata document. One possible argument against this approach is that project tracking and manifest information may be generated by very different routes. Also, PT currently duplicated most canonical content several times when multiple booklists are specified (but maybe we should fix the duplication of content).

For Metadata 2.3 we should pick one solution that can handle progress on peripherals, and remove support for the unused solution(s).

15.1 Migration

This obviously depends on which option we pick (and on whether anyone is actually consuming progress information via DBL.)