OpenLyrics Documentation

version 0.8

Martin Zibricky, Raoul Snyman

April 15, 2012

Contents

Documentation Contents

Introduction

About

OpenLyrics is a free, open XML standard for Christian worship songs. The goal of OpenLyrics is to provide an application-independent and operating system-independent song format for interoperability between applications.

History

The first version of OpenLyrics was created in 2008, when the OpenLP project leader approached the ChangingSong project leader, and proposed cooperation between OpenLP and ChangingSong to improve data exchange between the two projects. They agreed that a good first step would be to create an independent interoperable data format to provide better song exchange between the two applications.

Furthermore, experiences the leader of the ChangingSong had with the OpenSong project's XML format for songs made him aware that the OpenSong format was not sufficient for the proposed features in ChangingSong, and thus a new format would be necessary in order to implement many of the more advanced features the project planned to develop.

Upon planning the OpenLyrics format, the two leaders decided to make the format as open and inclusive as possible so that other presentation projects, both open source and proprietary, could use this new format as well.

To that end, they decided to make OpenLyrics an XML format. XML is a well established standard with solid support in many programming languages, and there are a plethora of XML libraries for manipulating XML.

The current design of OpenLyrics is based on the OpenSong data format along with some features suggested by users, particularly the ability to use multiple languages for a song (forum) .

Release Numbering

OpenLyrics uses the following release numbering scheme:

```
X.X_pX
```

where `X.X` is the major release number, indicating the data format version and `pX` is the optional minor release number, used when incidental files (documentation, examples, and the like) are updated.

Therefore, this would be a valid release number, indicating a new data format:

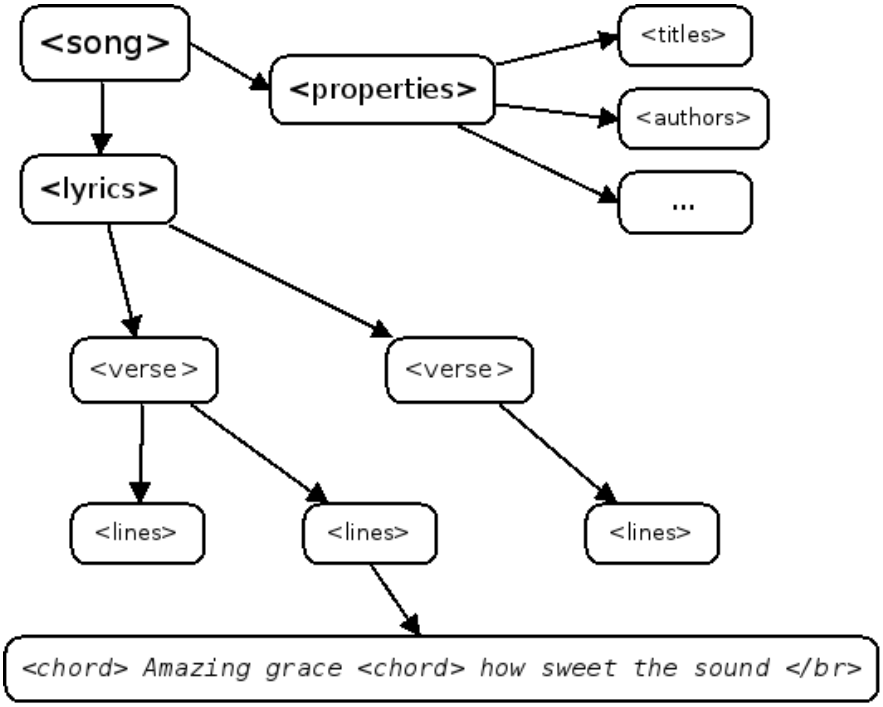
```
0.7
```

And this would be the minor release made when we correct typos in documentation, add an example song, etc:

```
0.7_p1
```

Data Format

Basic Structure



Features

categories
`<theme>`

CCLI support
`<ccliNo>`

chords
`<chord name="D">`

comments in lyrics
`<verse><lines><comment/></lines></verse>`

date of song release
`<releaseDate>`

format version
`<song version="0.7>`

keywords for searching
`<keywords>`

last modification time
`<song modifiedDate="">`

lines of text
`<line>`

multiple authors
`<authors>`

multiple categories
`<themes>`

multiple song titles
`<titles>`
multiple user-defined items
`<comments>`
music properties
`<transposition>` `<tempo>` `<key>`
namespace
`<song xmlns="http://openlyrics.info/namespace/2009/song">`
parts
`<lines part="men">`
slides
`<verse>`
song book
`<collection>` `<trackNo>`
song metadata
`<song version="">` `<song createdIn="">` `<song modifiedIn="">` `<song modifiedDate="">`
song translator
`<author type="translator" lang="cs">`
song variant
`<variant>`
song version
`<customVersion>`
tagging verse type
`<verse name="v1">`
translated lyrics
`<verse name="v1" lang="en">`
translated song title
`<title lang="en">`
translated theme
`<theme lang="en">`
transliterated lyrics
`<verse name="v1" lang="en" translit="he">`
transliterated song title
`<title lang="en" translit="he">`
transliterated theme
`<theme lang="en" translit="he">`
transposition
`<transposition>`
user-defined item
`<comment>`
verse order
`<verseOrder>`

Required Data Items

Here is an example of a song containing only the required XML tags:

```
<song xmlns="http://openlyrics.info/namespace/2009/song"
      version="0.8"
      createdIn="OpenLP 1.9.0"
      modifiedIn="ChangingSong 0.0.1"
      modifiedDate="2010-01-28T13:15:30+01:00">
  <properties>
    <titles>
      <title>Amazing Grace</title>
    </titles>
  </properties>
  <lyrics>
    <verse name="v1">
      <lines>
        Amazing grace how sweet the sound
      </lines>
    </verse>
  </lyrics>
</song>
```

As you can see from this example, a minimal song should only contain:

- metadata
- title
- verse with one line of text

Tags with empty values are not allowed. If a tag is empty, it should be excluded from the XML.

Metadata

Metadata is **mandatory** and should be present in every song. The metadata is useful in the debugging of OpenLyrics implementations in applications.

Metadata is enclosed in the <song> tag as attributes:

```
<song xmlns="http://openlyrics.info/namespace/2009/song"
      version="0.8"
      createdIn="OpenLP 1.9.0"
      modifiedIn="ChangingSong 0.0.1"
      modifiedDate="2010-01-28T13:15:30+01:00">
```

xmlns
: Defines an XML namespace. The value should be always `http://openlyrics.info/namespace/2009/song`

version
: Version of the OpenLyrics format used by a song. This gives applications the ability to notify users if the application doesn't support newer versions of OpenLyrics.

createdIn

String to identify the application where a song was created for the first time. This attribute should be set when a new song is created. It should not be changed with additional updates and modification to the song, even when the song is edited in another application. The recommended content of this attribute is *application name* and *version*, e.g. `OpenLP 1.9.0`.

`modifiedIn`

String to identify the application where a song was edited for most recently. This attribute should be set with every modification. The recommended content of this attribute is *application name* and *version*, e.g. `OpenLP 1.9.0`.

`modifiedDate`

Date and time of the last modification. This attribute should be set with every modification of the song. This attribute should use the ISO 8601 date format, which looks like this:

```
YYYY-MM-DDThh:mm:ss±[hh]:[mm]
```

For example, the 28th of January, 2010, at 30 seconds past 1:15pm in the UTC+1 timezone would look like this:

```
2010-01-28T13:15:30+01:00
```

Encoding and Filenames

Encoding

The recommended encoding for OpenLyrics files is the ubiquitous UTF-8 encoding. *UTF-8* is supported by most programming languages, and using this encoding means that OpenLyrics files can have more than one language per file.

File Names

When creating and saving OpenLyrics files, it is recommended that the song contained in the file should be easily identifiable by looking at the file name. A well-named file would probably use a combination of one or more of the following fields:

- `<titles>`
- `<variant>`
- `<authors>`

In addition to this, the file extension should be `.xml` since OpenLyrics is an XML format.

File name examples:

```
Amazing Grace.xml
Amazing Grace (old hymn).xml
Amazing Grace (John Newton).xml
```

Additionally, file names should not contain characters which could cause issues on any operating system. Most modern operating systems support a wide range of characters in file names, but some of the common characters to avoid are `/`, `\` and `:`.

Compressed file formats should also be taken into consideration when naming files, as some compression formats (most notably ZIP files) cannot handle all valid file name characters. It is recommended that files should be compressed using the 7-Zip format, as this format is

known to handle non-ASCII file names well.

Song Properties

OpenLyrics songs are essentially divided into two sections. The first section, denoted by the `<properties>` tag, contains the various properties of the song, and the second section, denoted by the `<lyrics>` tag, contains the lyrics.

The `<properties>` tag groups various song property tags together. These tags include the `<titles>` and `<authors>` tags. The order of tags within the `<properties>` tag is arbitrary. For example, it doesn't matter if the `<titles>` tag occurs before the `<authors>` tag:

```
<titles><title>Amazing Grace</title></titles>
<authors><author>John Newton</author></authors>
```

Or if the `<titles>` tag occurs after the `<authors>` tag:

```
<authors><author>John Newton</author></authors>
<titles><title>Amazing Grace</title></titles>
```

An application implementing the OpenLyrics format should not depend on any order of tags enclosed in the ``<properties>`` tag.

Titles

The `<titles>` tag is **mandatory**, and every song must contain at least one title:

```
<titles><title>Amazing Grace</title></titles>
```

However, there could be any number of titles:

```
<titles>
  <title>Amazing Grace</title>
  <title>Amazing</title>
</titles>
```

An optional `lang` attribute can be added to the `<title>` tag. This attribute defines the language of the title. The format of this attribute should be `xx` or `xx-YY`, where `xx` is a language code from the ISO-639 standard, and `YY` is a country code. For more details see BCP 47.

The `lang` attribute comes in handy when the song is translated from one language to another and it is necessary to know the translated version of the title, or when a song contains lyrics in multiple languages:

```
<titles>
  <title lang="en">Amazing Grace</title>
  <title lang="de">Staunenswerte Gnade</title>
  <title lang="pl">Cudowna Boża łaska</title>
</titles>
```

An additional `original` attribute, containing a boolean value of either `true` or `false`, can be used to indicate that the associated title is the original title of the song:

```
<titles>
  <title lang="en" original="true">Amazing Grace</title>
  <title lang="pl">Cudowna Boża łaska</title>
</titles>
```

Authors

The <authors> tag is optional. When this tag is present in the song, there should be at least one <author> sub-tag:

```
<authors><author>John Newton</author></authors>
```

There can, of course, be more authors:

```
<authors>
  <author>John Newton</author>
  <author>Johannes Newton</author>
</authors>
```

Three different types of authors can be defined:

- *author of words*:

  ```
  <author type="words">John Newton</author>
  ```

- *author of music*:

  ```
  <author type="music">John Newton</author>
  ```

- *translator*:

  ```
  <author type="translation" lang="cs">Jan Ňůtn</author>
  ```

  When the type is translation, a lang attribute is mandatory. The value of this attribute should be in the same format as the lang attribute of the <title> tag.

Copyright

The <copyright> tag contains the copyright information of the song. In some countries it is a legal requirement to display copyright information during the presentation of songs. The <copyright> tag has no specific format, though it is recommended that the value contains at least the year and copyright holder of the song.

For example:

```
<copyright>Public Domain</copyright>
```

Or:

```
<copyright>1998 Vineyard Songs</copyright>
```

CCLI Number

CCLI stands for *Christian Copyright Licensing International*. CCLI is an organisation that offers copyright licensing of songs and other resource materials to churches and Christian organisations for use in Christian worship. For registered churches, CCLI offers songs and other resources for download. A CCLI ID is assigned to every song. This tag provides integration with CCLI.

The CCLI number (ID) must be a positive integer:

```
<ccliNo>22025</ccliNo>
```

Release Date

The `<released>` tag tracks the date when a song was released or published.

It can be just a year:

```
<released>1779</released>
```

Or a year and a month:

```
<released>1779-09</released>
```

Or a year, month and day:

```
<released>1779-12-30</released>
```

Or even a year, month, day and time:

```
<released>1779-12-31T13:15</released>
```

Transposition

The `<transposition>` tag is used when it is necessary to move the key or the pitch of chords up or down. The value must be a positive or negative integer.

A negative value moves the pitch down by a fixed number of semitones:

```
<transposition>-3</transposition>
```

A positive value moves the pitch up by a fixed number of semitones:

```
<transposition>4</transposition>
```

Tempo

The tempo of a song defines the speed at which a song is to be played. It could be expressed in beats per minute (BPM) or as any text value. The `<tempo>` tag has a `type` attribute which defines whether the tempo is measured in BPM or by a phrase. The `type` attribute therefore can be one of two possible values, `bpm` and `text`.

If the tempo is measured in BPM, it must be a positive integer in the range of 30-250:

```
<tempo type="bpm">90</tempo>
```

If the tempo is expressed as a phrase, it can contain any arbitrary text. For example `Very Fast`, `Fast`, `Moderate`, `Slow`, `Very Slow`, etc.:

```
<tempo type="text">Moderate</tempo>
```

Key

The key determines the musical scale of a song. For example, `A`, `B`, `C#`, `D`, `Eb`, `F#` or `Ab`.

Example:

```
<key>Eb</key>
```

Variant

The `<variant>` tag is used to differentiate between songs which are identical, but may be performed or sung differently.

For example, there could be two songs with the title *Amazing Grace*. One song was published many years ago and one song was published by a well known band, say for instance the *Newsboys*.

For the old song the `<variant>` could be:

```
<variant>Original Hymn</variant>
```

While the `<variant>` by the well known band would list their name:

```
<variant>Newsboys</variant>
```

Publisher

The `<publisher>` tag contains the name of the publisher of the song:

```
<publisher>Sparrow Records</publisher>
```

Custom Version

No song is ever created once, never to be edited again. Songs are updated over time, sometimes to add additional verses, sometimes to fix spelling or grammatical errors. OpenLyrics tries to add in some rudimentary version control in the form of a `<version>` tag, which could be updated whenever a song changes significantly.

This tag can contain any arbitrary text which could help the user to distinguish between various versions of a song.

For example, it could contain a version number:

```
<version>0.99</version>
```

Or a date:

```
<version>2010-02-04</version>
```

Or almost anything else:

```
<version>this is previous version</version>
```

Keywords

Keywords are used for more precise results when searching for a song in a song database. These keywords are stored in the <keywords> tag.

For example, in *Amazing Grace*:

```
<keywords>amazing grace, how sweet the sound, God's grace</keywords>
```

Verse Order

The verse order of a song defines the order in which the verses are typically sung or performed. The verse order is denoted by the <verseOrder> tag.

The verse order is a space-separated string of verse names (which are defined in the <lyrics> section of the file). Verse names can appear multiple times, and should be lowercase. See the <verse> section for more information on verse names.

For example:

```
<verseOrder>v1 c v2 c v1 c</verseOrder>
```

Song Books

Most songs come from some sort of collection of songs, be it a book or a folder of some sort. It may be useful to track where the song comes from, and for this can be done through the <songbook> tag.

Because songs are often found in more than one song book, multiple <songbook> tags can be defined. For this reason, <songbook> tags are wrapped in a <songbooks> tag.

Each <songbook> tag contains two attributes:

name
: The name of a song book is stored in the name attribute.

entry
: As songs are normally indexed in song books, the index of the song is stored in the entry attribute.

Both attributes can contain any text:

```
<songbooks>
  <songbook name="Name of a songbook or collection" entry="48"/>
</songbooks>
```

The name attribute is mandatory but entry is optional:

```
<songbooks>
  <songbook name="Name of a songbook or collection"/>
</songbooks>
```

Themes

Themes are used to categorize songs. Having songs categorized can be useful when choosing songs for a ceremony or for a particular sermon topic. A theme is defined by a `<theme>` tag. A song can have multiple themes, so any `<theme>` tags are wrapped in a `<themes>` tag:

```
<themes><theme>Adoration</theme></themes>
```

A `<theme>` tag has an optional `lang` attribute, which defines the language of the theme. The value of this attribute should be in the same format as the `lang` attribute of the `<title>` tag.

Some examples:

```
<themes>
  <theme>Adoration</theme>
  <theme lang="en-US">Grace</theme>
  <theme lang="pt-BR">Graça</theme>
  <theme lang="en-US">Praise</theme>
  <theme lang="pt-BR">Adoração</theme>
  <theme lang="en-US">Salvation</theme>
  <theme lang="pt-BR">Salvação</theme>
</themes>
```

It is highly recommended that themes should come from the list of themes on the CCLI web site: http://www.ccli.co.za/owners/themes.cfm

Comments

The `<comment>` tag is used to store any additional, unspecified user data in a free-form text field. A song can contain multiple `<comment>` tags, and thus they are wrapped in a `<comments>` tag.

An example:

```
<comments>
  <comment>One of the most popular songs in our congregation.</comment>
  <comment>We sing this song often.</comment>
</comments>
```

Song Lyrics

The second section of an OpenLyrics song is defined by the `<lyrics>` tag. This tag contains words of a song and other data related to it.

The `<lyrics>` tag contains one or more `<verse>` tags. Each `<verse>` tag defines a verse or stanza of a song, and contains a single mandatory attribute, `name`. Each verse can contain one or more `<lines>` tags, which holds a logical grouping of words.

A song should contain at least **one verse**:

```
<lyrics>
  <verse name="v1">
    <lines>
      This is the first line of the text.
    </lines>
```

```
  </verse>
</lyrics>
```

There can be multiple <lines> tags:

```
<verse name="v1">
  <lines>
    This is the first line of the text.
  </lines>
  <lines>
    This is the second line of the text.
  </lines>
</verse>
```

And of course, a song can contain multiple verses:

```
<lyrics>
  <verse name="v1">
    <lines>First line of first verse.</lines>
  </verse>
  <verse name="v2">
    <lines>First line of second verse.</lines>
  </verse>
</lyrics>
```

The <verse> tag is not only used for verses, but also choruses, bridges, etc.

Line Breaks

Within a <lines> tag, a
 tag is used to define breaks between lines.

For example:

```
<lines>
  Amazing grace, how sweet the sound<br/>
  That saved a wretch like me!</br>
  I once was lost, but now am found,<br/>
  Was blind but now I see.<br/>
</lines>
```

Verse Name

As previously mentioned, every <verse> tag has a mandatory name attribute. Verse names should be unique, written in **lower case**, a single word, and should follow the naming convention as laid out in the table below:

Name	Description
v1, v2, v3, ...	first verse, second verse, third verse, ...
v1a, v1b, v1c, ...	first verse part 1, first verse part two, ...
c	chorus
c1, c2, ...	first chorus, second chorus, ...

c1a, c1b, ...	first chorus part 1, first chorus part 2, ...
p	pre-chorus
p1, p2, ...	first pre-chorus, second pre-chorus, ...
p1a, p1b, ...	first pre-chorus part 1, first pre-chorus part 2, ...
b	bridge
b1, b2, ...	first bridge, second bridge, ...
b1a, b1b, b1c, ...	first bridge part 1, first bridge part 2, ...
e	ending
e1, e2, ...	first ending, second ending, ...
e1a, e1b, e1c, ...	first ending part 1, first ending part 2, ...

According to the table above, a song containing two verses (*v1, v2*), a chorus (*c*), a bridge (*b*) and an ending (*e*) would look like this:

```
<lyrics>
  <verse name="v1">
    ...
  </verse>
  <verse name="v2">
    ...
  </verse>
  <verse name="c">
    ...
  </verse>
  <verse name="b">
    ...
  </verse>
  <verse name="e">
    ...
  </verse>
</lyrics>
```

Chords

The OpenLyrics format also provides the ability to include chords in the lyrics of songs. The tag containing a chord name looks like this:

```
<chord name="D7"/>
```

The <chord> tags are mixed in with the lyrics of a song. This tag should be placed immediately before the letters where it should be played:

```
<lyrics>
  <verse name="v1">
    <lines>
      <chord name="D7"/>Amazing grace how <chord name="E"/>sweet the sound<br/>
      That saved <chord name="A"/>a wretch <chord name="F#"/>like me.
    </lines>
  </verse>
```

```
</lyrics>
```

Chords should be written according to this *list of chords*.

Multiple Languages (Lyrics Translations)

The translation of lyrics can be useful for situations where a song is written in a language that the majority of the congregation does not know. A translation of the song can be displayed in a language common to most of the congregation.

OpenLyrics supports the translation of verses by adding a `lang` attribute to `<verse>` tags. To add translations to a particular verse, the `<verse>` tag should be repeated, with the same `name` attribute value as the verse to be translated, and with `lang` attribute set to the language of the translation. The value of the `lang` attribute should be in the same format as the `lang` attribute used in other tags.

Multiple translations of a verse should have the same value of the `name` attribute but different values of `lang`.

For example, this song is written in English and has a German translation for the first verse:

```
<lyrics>
  <verse name="v1" lang="en">
    <lines>This text is in English.</lines>
  </verse>
  <verse name="v1" lang="de">
    <lines>Dieses Text ist auf Deutsch.</lines>
  </verse>
</lyrics>
```

Transliteration

Transliteration is the process whereby words from one writing system are converted to another writing system. For instance there might be a Hebrew song, written in the Hebrew alphabet, which is then rewritten into the English alphabet (but not into English) so that it is easier for the congregation to pronounce the Hebrew words.

Transliteration can be defined by adding a `translit` attribute to the `<title>`, `<theme>` or `<verse>` tags. The value of this attribute should be the same format as the `lang` tags.

The `translit` attribute must be used in conjunction with the `lang` attribute. This is because one writing system can be transliterated into different languages in different ways. For example, Hebrew is transliterated into English a different way than when it is transliterated into French.

In the following example the `lang` attribute defines the language of original alphabet (Hebrew) and `translit` defines the language into which the song was transliterated (English):

```
<verse name="v1" lang="he" translit="en">
...
</verse>
```

As an example, here is a song which was originally written in Hebrew, then transliterated to the English alphabet, and then finally translated into English:

```
<lyrics>
  <verse name="v1" lang="he">
    <lines>□□□ □□□□□</lines>
  </verse>
  <verse name="v1" lang="he" translit="en">
    <lines>Hava nagila</lines>
  </verse>
  <verse name="v1" lang="en">
    <lines>Let's rejoice</lines>
  </verse>
</lyrics>
```

Verse Parts (Groups of Lines)

In some songs, certain lines or sections of the song may be sung by a particular group of people. For example, some songs contain sections where only the men or only the women sing. The `part` attribute, attached to the `<lines>` tag, marks these different sections (or parts) of songs. The value of this attribute is can be any arbitrary text.

For example, a song containing one verse with some words for men and some words for women:

```
<lyrics>
  <verse name="v1">
    <lines part="men">
      First line of words sung by men.<br/>
      Second line of words sung by men.
    </lines>
    <lines part="women">
      First line of words sung by women.<br/>
      Second line of words sung by women.
    </lines>
  </verse>
</lyrics>
```

Comments in Lyrics

The OpenLyrics format also supports comments within lyrics. Comments are useful for adding non-visible information. For example, a comment could contain the style in which to play or sing any particular set of lyrics. Once again, comments are defined by the `<comment>` tag.

For example:

```
<lyrics>
  <verse name="v1">
    <lines>
      <comment>Singing loudly.</comment>
      Text of verse.<br/>
      <comment>Singing quietly.</comment>
      Text of verse.
    </lines>
  </verse>
  <verse name="c">
```

```
    <lines>
      <comment>Singing loudly.</comment>
      Line content.<br/>
      Line content.
    </lines>
  </verse>
</lyrics>
```

Advanced Example

More song examples can be found in the songs directory distributed with the OpenLyrics archive.

Here's an advanced example of the XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<song xmlns="http://openlyrics.info/namespace/2009/song"
      version="0.7"
      createdIn="OpenLP 2.0"
      modifiedIn="ChangingSong 0.0.2"
      <!-- date format: ISO 8601 -->
      modifiedDate="2009-12-22T21:24:30+02:00">
  <properties>
    <titles>
      <title>Amazing Grace</title>
    </titles>
    <authors>
      <author>John Newton</author>
    </authors>
    <copyright>Public Domain</copyright>
    <ccliNo>2762836</ccliNo>
    <releaseDate>1779</releaseDate>
    <tempo type="text">moderate</tempo>
    <key>D</key>
    <verseOrder>v1 v2 v3 v4 v5 v6</verseOrder>
    <themes>
      <theme>Assurance</theme>
      <theme>Grace</theme>
      <theme>Praise</theme>
      <theme>Salvation</theme>
    </themes>
  </properties>
  <lyrics>
    <verse name="v1">
      <lines>
        Amazing grace how sweet the sound<br/>
        That saved a wretch like me.<br/>
        I once was lost, but now am found,<br/>
        Was blind but now I see.
      </lines>
    </verse>
    <verse name="v2">
      <lines>
        'Twas grace that taught my heart to fear,<br/>
```

```
        And grace my fears;<br/>
        How precious did that grace appear<br/>
        The hour I first believed.
      </lines>
    </verse>
    <verse name="v3">
      <lines>
        Through many dangers, toil and snares,<br/>
        I have already come;<br/>
        'Tis grace has brought me safe thus far,<br/>
        And grace will lead me home.
      </lines>
    </verse>
    <verse name="v4">
      <lines>
        When we've been there ten thousand years<br/>
        Bright shining as the sun,<br/>
        We've no less days to sing God's praise<br/>
        Than when we've first begun.
      </lines>
    </verse>
  </lyrics>
</song>
```

Data Validation

Data validation is the process of ensuring that the structure of the data confirms to a defined format. This is especially useful for developers when they add support for a data format to their application. They are able to test that the data they produce is correct and conforms to the format.

In the world of XML, validation involves checking the names of elements, their nesting, names of attribues and the type of values in elements and attributes. Several XML Schema languages exist for the formal definition of an XML structure. For the OpenLyrics data format the RelaxNG chosen and used to create the format definition.

Bundled CLI Script

This section describes how to use bundled script `validate.py` in command prompt.

Prerequisites

Before using the script for validation please ensure that the following is available in your system:

- Python >= 2.5
- lxml

Usage

To execute the script, use the following command:

```
python validate.py  openlyrics_schema.rng  xmlfile.xml
```

`xmlfile.xml` is the file which you need to validate and `openlyrics_schema.rng` contains OpenLyrics RelaxNG XML schema.

Tools and Libraries

The OpenLyrics RelaxNG XML schema can be used in any programming language which has libraries with support for XML schemas. It is also possible, to some extent, to convert RelaxNG schemas to other languages, like DTD or W3C XML Schema.

- Sun Multi-Schema Validator: Java technology tool to validate XML documents against several kinds of XML schemata.
- lxml: Pythonic xml library, offers support for XPath, RelaxNG, XML Schema, XSLT, C14N and more.
- On-line validator.
- Jing: A validator written in Java.
- XSD to Relax NG: An online converter.
- Libxml2: XML C library, with support for RelaxNG validation.

A list of other software for RelaxNG can be found on the RelaxNG site.

RelaxNG XML schema

The following RelaxNG XML schema was created:

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
         ns="http://openlyrics.info/namespace/2009/song">

  <!-- TOP LEVEL -->

  <start>
    <element name="song">
      <ref name="songAttributes"/>
      <ref name="properties"/>
      <optional>
        <ref name="format"/>
      </optional>
      <ref name="lyrics"/>
    </element>
  </start>

  <define name="properties">
    <element name="properties">
      <interleave> <!-- allow occur in any order -->
        <!-- at least one title is always required -->
        <ref name="titles"/>
        <!-- other properties items are optional -->
        <optional>
          <ref name="authors"/>
        </optional>
        <optional>
          <ref name="copyright"/>
```

```
        </optional>
        <optional>
          <ref name="ccliNo"/>
        </optional>
        <optional>
          <ref name="released"/>
        </optional>
        <!-- Music Info -->
        <optional>
          <ref name="transposition"/>
        </optional>
        <optional>
          <ref name="tempo"/>
        </optional>
        <optional>
          <ref name="key"/>
        </optional>
        <!-- Other Info -->
        <optional>
          <ref name="variant"/>
        </optional>
        <optional>
          <ref name="publisher"/>
        </optional>
        <optional>
          <ref name="version"/>
        </optional>
        <optional>
          <ref name="keywords"/>
        </optional>
        <optional>
          <ref name="verseOrder"/>
        </optional>
        <optional>
          <ref name="songbooks"/>
        </optional>
        <optional>
          <ref name="themes"/>
        </optional>
        <optional>
          <ref name="comments"/>
        </optional>
      </interleave>
    </element>
  </define>

  <define name="format">
    <element name="format">
        <ref name="formatTags"/>
    </element>
  </define>


  <define name="lyrics">
```

```
    <element name="lyrics">
      <!-- at least one verse is required -->
      <oneOrMore>
        <ref name="verse"/>
      </oneOrMore>
    </element>
  </define>

  <!-- PROPERTIES -->

  <define name="titles">
    <element name="titles">
      <oneOrMore>
        <element name="title">
          <ref name="nonEmptyContent"/>
          <optional>
            <ref name="langAttribute"/>
            <optional>
              <ref name="translitAttribute"/>
            </optional>
          </optional>
          <optional>
            <attribute name="original">
              <data type="boolean"/>
            </attribute>
          </optional>
        </element>
      </oneOrMore>
    </element>
  </define>

  <!-- AUTHOR info -->

  <define name="authors">
    <element name="authors">
      <oneOrMore>
        <element name="author">
          <ref name="nonEmptyContent"/>
          <optional>
            <choice>
              <attribute name="type">
                <choice>
                  <value>words</value>
                  <value>music</value>
                </choice>
              </attribute>
              <!-- when attrib 'type' value is 'translation' require attribute 'lang'.
                   'xml:lang' can't be used. xml:lang means in what language is the
                   content of an element and this is not the case. -->
              <group>
                <attribute name="type">
                  <value>translation</value>
                </attribute>
                <ref name="langAttribute"/>
```

```
              </group>
            </choice>
          </optional>
        </element>
```

```
      </oneOrMore>
    </element>
  </define>

  <define name="copyright">
    <element name="copyright">
      <ref name="nonEmptyContent"/>
    </element>
  </define>

  <define name="ccliNo">
    <element name="ccliNo">
      <data type="positiveInteger"/>
    </element>
  </define>

  <define name="released">
    <element name="released">
      <!-- allowed values
           1779
           1779-12
           1779-12-31
           1779-12-31T13:15:30+01:00 -->
      <choice>
        <data type="gYear"/>
        <data type="gYearMonth"/>
        <data type="date"/>
        <data type="dateTime"/>
      </choice>
    </element>
  </define>

  <!-- MUSIC INFO -->

  <define name="transposition">
    <element name="transposition">
      <data type="integer">
        <param name="minInclusive">-99</param>
        <param name="maxInclusive">99</param>
      </data>
    </element>
  </define>

  <define name="tempo">
    <element name="tempo">
      <choice>
        <!-- attrib 'type' value 'bpm' - beatss per minute required -->
        <group>
          <data type="positiveInteger">
            <param name="minInclusive">30</param>
            <param name="maxInclusive">250</param>
          </data>
          <attribute name="type">
            <value>bpm</value>
```

```
        </attribute>
      </group>
      <!-- attrib 'type' value 'text' - any text -->
      <group>
        <ref name="nonEmptyContent"/>
        <attribute name="type">
          <value>text</value>
        </attribute>
      </group>
    </choice>
  </element>
</define>


<define name="key">
  <element name="key">
    <ref name="nonEmptyContent"/>
  </element>
</define>

<!-- OTHER INFO -->

<define name="variant">
  <element name="variant">
    <ref name="nonEmptyContent"/>
  </element>
</define>

<define name="publisher">
  <element name="publisher">
    <ref name="nonEmptyContent"/>
  </element>
</define>

<define name="version">
  <element name="version">
    <ref name="nonEmptyContent"/>
  </element>
</define>

<define name="keywords">
  <element name="keywords">
    <ref name="nonEmptyContent"/>
  </element>
</define>

<define name="verseOrder">
  <element name="verseOrder">
    <list>
      <oneOrMore>
        <ref name="verseNameType"/>
      </oneOrMore>
    </list>
  </element>
```

```
  </define>

  <define name="songbooks">
    <element name="songbooks">
      <oneOrMore>
        <element name="songbook">
          <attribute name="name">
            <ref name="nonEmptyContent"/>
          </attribute>
          <optional>
            <!-- 'entry' is like song number but song number must not
                 always be integer and it can contain letters.
                 examples: '153c' or '023', etc. -->
            <attribute name="entry">
              <ref name="nonEmptyContent"/>
            </attribute>
          </optional>
        </element>
      </oneOrMore>
    </element>
  </define>

  <define name="themes">
    <element name="themes">
      <oneOrMore>
        <element name="theme">
          <ref name="nonEmptyContent"/>
          <optional>
            <ref name="langAttribute"/>
            <optional>
              <ref name="translitAttribute"/>
            </optional>
          </optional>
        </element>
      </oneOrMore>
    </element>
  </define>

  <define name="comments">
    <element name="comments">
      <oneOrMore>
        <element name="comment">
          <ref name="nonEmptyContent"/>
        </element>
      </oneOrMore>
    </element>
  </define>

  <!-- FORMAT -->

  <define name="formatTags">
    <!-- Allow only one set of formatting tags for lyrics -->
    <element name="tags">
      <attribute name="application">
```

```
        <ref name="nonEmptyContent"/>
      </attribute>
      <oneOrMore>
        <ref name="formatTagsTag"/>
      </oneOrMore>
    </element>
  </define>

  <define name="formatTagsTag">
    <element name="tag">
      <attribute name="name">
        <ref name="nonEmptyContent"/>
      </attribute>
      <element name="open">
        <ref name="nonEmptyContent"/>
      </element>
      <!-- Close element is optional. Formatting without text may be present.
           e.g. <br/> -->
      <optional>
        <element name="close">
          <ref name="nonEmptyContent"/>
        </element>
      </optional>
    </element>
  </define>

<!-- LYRICS -->

  <define name="verse">
    <element name="verse">
      <ref name="verseAttributes"/>
      <optional>
        <ref name="langAttribute"/>
        <optional>
          <ref name="translitAttribute"/>
        </optional>
      </optional>
      <oneOrMore>
        <ref name="lines"/>
      </oneOrMore>
    </element>
  </define>

  <define name="lines">
    <element name="lines">
      <optional>
        <attribute name="part">
          <ref name="nonEmptyContent"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="break">
          <value>optional</value>
        </attribute>
```

```
      </optional>
      <zeroOrMore>
        <ref name="linesContent"/>
      </zeroOrMore>
      <ref name="linesContent"/>
    </element>
  </define>

  <define name="chord">
    <element name="chord">
      <attribute name="name">
        <ref name="nonEmptyContent"/>
      </attribute>
      <empty/>
    </element>
  </define>

  <define name="tag">
    <element name="tag">
      <attribute name="name">
        <ref name="nonEmptyContent"/>
      </attribute>
      <!-- allow using more formatting tags for text -->
      <!-- e.g. <tag name="bold"><tag name="red">my text</tag></tag> -->
      <choice>
        <oneOrMore>
          <ref name="linesContent"/>
        </oneOrMore>
        <!-- Allow empty tag. Formatting without text may be present.
             e.g. <tag name="br"/> -->
        <empty/>
      </choice>
    </element>
  </define>

  <define name="verseAttributes">
    <attribute name="name">
      <ref name="verseNameType"/>
    </attribute>
  </define>

  <define name="songAttributes">
    <!-- by default: value of type string is required in attr -->
    <attribute name="version">
      <data type="NMTOKEN"> <!-- one word value -->
        <!-- allow only values like: '0.1' '11.2' '13.14.15'
        <param name="pattern">[0-9]+\.[0-9]+(\.[0-9]+)?</param> -->
        <!-- RelaxNG xml schema is specific for openlyrics version -->
        <param name="pattern">0\.8</param>
      </data>
    </attribute>
    <attribute name="createdIn">
      <ref name="nonEmptyContent"/>
    </attribute>
```

```
    <attribute name="modifiedIn">
      <ref name="nonEmptyContent"/>
    </attribute>
    <attribute name="modifiedDate">
      <!-- date format: ISO 8601 -->
      <data type="dateTime"/>
    </attribute>
  </define>

  <define name="verseNameType">
    <choice>
      <data type="NMTOKEN">
        <param name="minLength">1</param>
        <!-- verse - v1, v2, v1a, ...  3 letters: [verse][verse_number][verse_part]
             chorus c, c1, c2, c1a, ca, ...
             pre-chorus - p, p1, p2, p1a, pa, ...
             bridge - b, b1, b2, b1a, ba, ...
             ending - e, e1, e2, e1a, ea, ... -->
        <param name="pattern">(v[1-9]\d?[a-z]?)|([cpbe][a-z]?)|([cpbe][1-9]\d?[a-z]?)</param>
      </data>
      <!-- custom values of verse name - one word name -->
      <data type="NMTOKEN"/>
    </choice>
  </define>

  <define name="langAttribute">
    <attribute name="lang">
      <data type="language"/>
    </attribute>
  </define>

  <!-- transliteration -->
  <define name="translitAttribute">
    <attribute name="translit">
      <data type="language"/>
    </attribute>
  </define>

  <define name="nonEmptyContent">
    <data type="string">
      <param name="minLength">1</param>
    </data>
  </define>

  <define name="linesContent">
    <!-- allow tag 'tag' inside regular text - mixed content -->
    <optional>
      <ref name="tag"/>
    </optional>
    <!-- allow tag 'comment' inside regular text - mixed content -->
    <optional>
      <element name="comment">
        <ref name="nonEmptyContent"/>
      </element>
```

```
    </optional>
    <!-- allow tag 'chord' inside regular text - mixed content -->
    <optional>
      <ref name="chord"/>
    </optional>
    <text/>
    <optional>
      <element name="br">
```

```
        <empty/>
      </element>
    </optional>
  </define>

</grammar>
```

Conversion From Other Formats

Being able to convert songs from other formats to OpenLyrics is necessary to speed up the adoption of the OpenLyrics format and for compatibility with other applications which already support OpenLyrics.

OpenSong

One of the tools bundled with the OpenLyrics source documents is a command line tool to convert OpenSong song files to OpenLyrics. The conversion script should work in most situations, but do make sure that the conversion was successful before removing the old files.

Prerequisites

Before using the script for conversion please ensure that the following is available in your system:

- Python >= 2.5
- lxml

Usage

To execute the script use the following command:

```
python ./opensong2openlyrics.py opensong_file  openlyrics_file.xml
```

Where `opensong_file` is the original song in OpenSong format and `openlyrics_file.xml` is the name of the song in OpenLyrics format.

OpenLP

OpenLP already supports the OpenLyrics format via both importing and exporting. Simply choose "OpenLyrics" from the list when importing songs.

Other Formats

Conversion to other formats is not currently supported.

List of Chords

Ab Ab+ Ab4 Ab7 Ab11 Absus Absus4 Abdim Abmaj Abmaj7 Abm Abm7 A A+ A4 A6 A7 A7+ A9 A11 A13 A7sus4 A9sus Asus Asus2 Asus4 Adim Amaj Amaj7 Am A/D A/F# A/G# Am#7 Am6 Am7 Am7sus4 Am9 Am/G Amadd9 Am(add9) A# A#+ A#4 A#7 A#sus A#sus4 A#maj A#maj7 A#dim A#m A#m7 Bb Bb+ Bb4 Bb6 Bb7 Bb9 Bb11 Bbsus Bbsus4 Bbmaj Bbmaj7 Bbdim Bbm Bbm7 Bbm9 B B+ B4 B7 B7+ B7#9 B7(#9) B9 B11 B13 Bsus Bsus4 Bmaj Bmaj7 Bdim Bm B/F# BaddE B(addE) BaddE/F# Bm6 Bm7 Bmmaj7 Bm(maj7) Bmsus9 Bm(sus9) Bm7b5 C C+ C4 C6 C7 C9 C9(11) C11 Csus Csus2 Csus4 Csus9 Cmaj Cmaj7 Cm Cdim C/B

Cadd2/B CaddD C(addD) Cadd9 C(add9) Cm7 Cm11 C# C#+ C#4 C#7 C#7(b5) C#sus C#sus4 C#maj C#maj7 C#dim C#m C#add9 C#(add9) C#m7 Db Db+ Db7 Dbsus Dbsus4 Dbmaj Dbmaj7 Dbdim Dbm Dbm7 D D+ D4 D6 D7 D7#9 D7(#9) D9 D11 Dsus Dsus2 Dsus4 D7sus2 D7sus4 Dmaj Dmaj7 Ddim Dm D/A D/B D/C D/C# D/E D/G D5/E Dadd9 D(add9) D9add6 D9(add6) Dm7 Dm#5 Dm(#5) Dm#7 Dm(#7) Dm/A Dm/B Dm/C Dm/C# Dm9 D# D#+ D#4 D#7 D#sus D#sus4 D#maj D#maj7 D#dim D#m D#m7 Eb Eb+ Eb4 Eb7 Ebsus Ebsus4 Ebmaj Ebmaj7 Ebdim Ebadd9 Eb(add9) Ebm Ebm7 E E+ E5 E6 E7 E7#9 E7(#9) E7b9 E7(b9) E7(11) E9 E11 Esus Esus4 Emaj Emaj7 Edim Em Em6 Em7 Em/B Em/D Em7/D Emsus4 Em(sus4) Emadd9 Em(add9) F F+ F4 F6 F7 F9 F11 Fsus Fsus4 Fmaj Fmaj7 Fdim Fm F/A F/C F/D F/G F7/A Fmaj7/A Fmaj7/C Fmaj7(#5) Fadd9 F(add9) FaddG FaddG Fm6 Fm7 Fmmaj7 F# F#+ F#7 F#9 F#11 F#sus F#sus4 F#maj F#maj7 F#dim F#m F#/E F#4 F#m6 F#m7 Gb Gb+ Gb7 Gb9 Gbsus Gbsus4 Gbmaj Gbmaj7 Gbdim Gbm Gbm7 G G+ G4 G6 G7 G7+ G7b9 G7(b9) G7#9 G7(#9) G9 G9(11) G11 Gsus Gsus4 G6sus4 G6(sus4) G7sus4 G7(sus4) Gmaj Gmaj7 Gmaj7sus4 Gmaj9 Gm Gdim Gadd9 G(add9) G/A G/B G/D G/F# Gm6 Gm7 Gm/Bb G# G#+ G#4 G#7 G#sus G#sus4 G#maj G#maj7 G#dim G#m G#m6 G#m7 G#m9maj7 G#m9(maj7)

List of Themes

This list of themes comes from CCLI site.

```
Admonition
Adoration
Anointing
Ascension
Aspiration
Assurance
Atonement
Benediction
Beauty
Blessing
Blood
Calvary
Ceremony / Baby
Ceremony / Baptism
Ceremony / Funeral
Ceremony / Wedding
Challenge
Character
Children
Christ
Christian Life
Church
Cleansing
Comfort
Commitment
Communion
Compassion
Confession
Conquering
Consecration
Contentment
Conversion
Courage
```

List of Themes

```
Creation
Creator
Cross
Crucifixion
Declaration
Dedication
Devotion
Devotional
Discipleship
Encouragement
Eternal Life
Evangelism
Faith
Family
Fellowship
Forgiveness
Freedom
Giving
God's Attributes / Faithfulness
God's Attributes / Father
God's Attributes / Forgiveness
God's Attributes / Goodness
God's Attributes / Grace
God's Attributes / Greatness
God's Attributes / His Name
God's Attributes / Love
God's Attributes / Majesty
God's Attributes / Mercy
God's Attributes / Power
God's Attributes / Righteousness
God's Attributes / Will
God's Attributes / Wisdom
God's Word
Grace
Guidance
Healing
Heaven
Heritage
Holiness
Holy Spirit
Hope
Humility
Jesus
Jesus / Bread of Life
Jesus / Friendship
Jesus / Lamb of God
Jesus / Life Of Jesus
Jesus / Living Water
Jesus / Lordship Of Jesus
Jesus / Messiah
Jesus / Name Of Jesus
Jesus / The Light
Joy
Judgement
```

List of Themes

```
Kingship
Liberty
Love
Mercy
Miracles
Missions
Nature
Obedience
Offering
Peace
Petition
Praise
Prayer
Presence
Processional
Promise
Protection
Providence
Provision
Redemption
Refuge
Rejoice
Renewal
Repentance
Resurrection
Revival
Righteousness
Sacrifice
Salvation
Sanctification
Savior
Seasonal / Advent
Seasonal / Christmas
Seasonal / Easter
Seasonal / Epiphany
Seasonal / Harvest
Seasonal / Lent
Seasonal / Palm Sunday
Seasonal / Patriotic
Seasonal / Pentecost
Second Coming
Servanthood
Service
Stewardship
Strength
Testimony
Thankfulness
Tithes
Trials
Trust
Truth
Unity
Victory
Warfare
```

```
Worship
Youth
Zion
```

FAQ

This is a list of Frequently Asked Questions about OpenLyrics. Feel free to suggest new entries!

How do I...

There are no frequently asked questions as yet.

Glossary

metadata

Data not directly related to a song, but useful for debugging and other purposes. Other items include modification time and the name of the application used to create the song.

song properties

Additional information about a song. There can be author name, copyright information, publisher, release date, etc.

theme

Clasifying songs to various groups along suitability.

transliteration

Transcription of text in one alphabet to another while retaining the language it was written in. For instance writing Chinese characters using latin.

Changes in OpenLyrics

Release 0.8 (15 Apr 2012)

- Add support for application specific formatting tags in <lines> tag.
- Drop tag `<line>`, for line endings is used tag
- Rename `<customVersion>` to `<version>`.
- Rename `<releaseDate>` to `<released>`.
- Remove the 'id' attribute of the `<theme>` tag, as it was based on a line number in a file, which is not very accurate, nor very reliable.
- Add script 'tools/convert-schema.py' to convert songs from older OpenLyrics versions.

Release 0.7 (24 Mar 2010)

- Move attribute 'xml:lang' to just 'lang' since the tag could not always contain text in another language but also transliteration.
- Allow empty lines by using syntax `<line/>` or `<line></line>` for that.
- Add 'e' (ending) as a new reserved (standardized) name for verses (including other variants like e1, e2, e1a, ea, ...): `<verse name="e">`.

- Add transliteration support, attribute `translit=""`. can be used in tags <title>, <theme>, <verse>.
- Add multiple songbooks entries. Syntax changed from tags <collection> and <trackNo> to:

```
<songbooks>
  <songbook name="This is a Songbook Name" entry="123"/>
</songbooks>
```

- Add script 'opensong2openlyrics.py' to convert OpenSong songs to OpenLyrics format.
- Add examples of some public domain songs or not copyrighted.

Release 0.6 (22 Dec 2009)

- Drop tag <customData> (is ambiguous).

Release 0.5 (06 Dec 2009) - final draft

- Allow custom verse names <verse name="custom_name_name">.
- Custom tempo <tempo type="custom">steadily</tempo>.
- Stay with only one key (any text) <key>C#</key>.
- Allow any chord notation (any text).
- Restrict ccli theme 'id' to range 1-999.
- Theme value can't be empty.
- Namespace changed from http://www.openlyrics.info/2009/song/namespace to http://openlyrics.info/namespace/2009/song.
- Content of an optional tag is mandatory, when the tag present in xml.

Release 0.4 (21 Nov 2009) - draft

- Move to RelaxNG xml schema.
- Tag attribute change. xml:lang describes the language of an tag. Change from <author type="translation" xml:lang="cs"> to `` <author type="translation" lang="cs">``.
- Xml schema: in 'author' tag attribute 'lang' is required when attribute 'type' contains value 'translation' <author type="translation" lang="cs">.

Release 0.3 (18 Nov 2009) - draft

Release 0.2 (16 Nov 2009) - draft

Release 0.1 (28 Nov 2008) - draft

Projects using OpenLyrics

This is an (incomplete) alphabetic list of projects that use OpenLyrics or are experimenting with it as their supported import/export format.

ExpoSong

ExpoSong uses the OpenLyrics format to store songs. ExpoSong also both imports and exports songs in the OpenLyrics format.

- http://code.google.com/p/exposong/

OpenLP

OpenLP imports and exports songs using the OpenLyrics format. OpenLP also uses a stripped-down version of OpenLyrics internally to store the lyrics of songs.

- http://openlp.org/