FieldWorks 7 Writing Systems

Ken Zook

June 26, 2013

Contents

1 Introduction

FieldWorks 6.0 and earlier used ICU[1] locales for writing systems and stored master information in proprietary XML files. In FieldWorks 7.0 we started using Palaso[2] library code for our writing systems, storing the writing system data in LDML files. In FieldWorks 7.1 we have improved the Palaso and FieldWorks code to provide an SIL standard way for allowing user-defined languages, scripts, regions, and variants in a way that is consistent with the LDML standard. While this document is specifically for FieldWorks 7.1 or later, since most of the code is shared between WeSay and FieldWorks, the details generally apply to both programs as well as any others that make use of the Palaso library.

[1] International Components for Unicode (http://site.icu-project.org/).

[2] Palaso is an acronym for the Payap Language Software Development Group (http://palaso.org/). It is a partnership in Thailand between SIL International and Payap University. Their work includes re-usable software components that are shared between WeSay and FieldWorks, along with other applications.

2 LDML standard

LDML is an acronym for Unicode Locale Data Markup Language. Unicode Technical Standard #35 describes this standard at http://unicode.org/reports/tr35/. LDML provides an XML standard for storing information in the Unicode Common Locale Data Repository (http://cldr.unicode.org/). This is an on-line repository of locale data for many languages of the world. An LDML file holds information including standard identifiers for writing systems, and localized information for calendars, currency, numbers, time zones, language, and country names.

FieldWorks currently follows the LDML standard code for identifying writing systems in any language. This locale identifier generally follows the BCP47, or RFC5646 standard described at http://www.rfc-editor.org/rfc/bcp/bcp47.txt. FieldWorks currently does not make use of most of the locale information that is part of the LDML standard. However, the standard does allow for private use extensions to the standard. This section discusses the standard parts of LDML that FieldWorks uses. The following sections discuss Palaso and FieldWorks additions to the standard that we use in FieldWorks.

The top-level DTD description of a LDML file is

```
<!ELEMENT ldml (identity, (alias |(fallback*, localeDisplayNames?,
layout?, characters?, delimiters?, measurement?, dates?, numbers?,
units?, listPatterns?, collations?, posix?, segmentations?, rbnf?,
references?, special*))) >
```

Of these possible elements, FieldWorks currently uses the identity, collations, and special elements.

Here is a simple LDML file for English (en.ldml) excluding all of the localization information.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ldml SYSTEM "http://www.unicode.org/cldr/dtd/1.5/ldml.dtd">
<ldml>
<identity>
    <version number="$Revision: 1.22 $"/>
    <generation date="$Date: 2007/07/10 20:27:25 $"/>
    <language type="en" />
</identity>
<collations>
    <collation type="standard" >
        <rules>
            <reset>ae</reset>
            <s>æ</s>
            <t>Æ</t>
        </rules>
    </collation  >
</collations>
</ldml>
```

This standard file will not work in FieldWorks because we require a special palaso version number as well as a special element described below.

2.1 Identity element

A locale identifier is defined in the identity element, defined as:

```
<!ELEMENT identity (alias | (version, generation?, language, script?,
territory?, variant?, special*) ) >
```

The following example is a fully defined identity element.

```
<identity>
    <version number="" />
    <generation date="2011-07-20T16:43:48" />
    <language type="zh" />
    <script type="Latn" />
    <territory type="CN" />
    <variant type="fonipa" />
</identity>
```

The important elements here are language, script, territory, and variant. The name of the file should follow these parts. Thus, this ldml file would be zh-Latn-CN-fonipa.ldml. Each of these subtags (parts between hyphens) follow registered standards listed at http://www.iana.org/assignments/language-subtag-registry.While there are recommended case conventions as shown above, all software should ignore case distinctions.

For most languages, the language subtag follows the Ethnologue code (http://www.ethnologue.com/) which is officially called ISO 639-3 (http://www.sil.org/iso639-3/). However, for backwards compatibility, if a language is defined in ISO 639-1, then the two-letter code from this standard must be used instead of the Ethnologue code. There is also an ISO 639-2 standard that is a subset of ISO 639-3. A table comparing the -1 and -2 codes is at http://www.loc.gov/standards/iso639-2/php/English_list.php.

Script subtags must be 4 letters. Territory or Region subtags are 2 letters or 3 digits. Variant subtags can be up to 8 characters and may contain multiple subtags. A language subtag is required. If script, region, or variants are present, they must be in this order following the language subtag. Script, region, and variant subtags should only be used if necessary to distinguish between multiple writing systems for the same language.

Here are a few samples of valid locale identifiers:

en	English
fr	French
fr-fonipa	French using the International Phonetic Alphabet
zh-CN	Chinese in China (simplified HAN characters are assumed)
zh-Latn-CN	A Romanized form of Chinese in China
mta	Manobo, Cotabato

Microsoft Windows only supports certain locales for localization. For example, to localize Farsi, Windows requires the fa locale. The ISO639-3 code for Western Farsi is pes. So if we want users to see localized Farsi in the UI as well as enter Farsi data in the same language, we can’t use pes as the language code. To live with this restriction, FieldWorks automatically converts three ISO639-3 codes to equivalents that Windows accepts for localization.

- Standard Arabic arb is converted to ar
- Western Farsi pes is converted to fa
- Mandarin cmn is converted zh-CN (or zh if another region is specified)

2.2 Collations element

Multiple collation elements can be specified for each writing system to specify sorting. At this point FieldWorks only supports one collation. The DTD specification for collations and collation is as follows:

```
<!ELEMENT collations (alias | (default*, collation*, special*)) >
```

```
<!ELEMENT collation (alias | (base?, settings?, suppress_contractions?,
optimize?, rules?, special*)) >
```

FieldWorks provides four ways of specifying collations using a special palaso element. The possible sort methods are

- Default Ordering
- Same as another language
- Custom Simple (Shoebox style) rules
- Custom ICU Rules

2.2.1 Default Ordering

Default ordering uses the default ICU collation for an unspecified language. This is identical to using custom ICU rules without specifying any rules. In the LDML file, the collations element is left empty.

```
<collations />
```

2.2.2 Same as another language

If your language uses a standard collation specified by the operating system, you can use this option and select the desired language from the Language combo.

In the LDML file, the special palaso:sortRulesType value is set to OtherLanguage, and the alias source is set to the desired language. This example uses default French sorting.

```
<collation>
    <base>
    <alias
        source="fr" />
    </base>
    <special xmlns:palaso="urn://palaso.org/ldmlExtensions/v1">
    <palaso:sortRulesType
        value="OtherLanguage" />
    </special>
</collation>
```

2.2.3 Custom Simple (Shoebox style) rules

Custom simple rules provide a way to specify sort order in a simple way similar to Shoebox and Toolbox. Each line specifies a primary sort sequence and the space-separated characters within a line represent a secondary ordering where there is no primary difference. Here’s an example.

A Á a á
B b
C Ç c ç

While this is similar to Toolbox sorting, it is not identical as it supports tertiary sorting and does not support ignore characters or other special rules available in ICU sorting. More detail for this simple rules option can be seen at http://wesay.org/wiki/How_to_sort_using_a_custom_sort_sequence.

FieldWorks (via palaso) converts the Toolbox specifications into ICU rules and stores them this way in the LDML file.

```
<collation>
    <rules>
        <reset
            before="primary">
            <first_non_ignorable />
        </reset>
        <p>A</p>
        <s>Á</s>
        <s>a</s>
        <s>á</s>
        <p>B</p>
        <s>b</s>
        <p>C</p>
        <s>Ç</s>
        <s>c</s>
        <s>ç</s>
    </rules>
    <special xmlns:palaso="urn://palaso.org/ldmlExtensions/v1">
        <palaso:sortRulesType value="CustomSimple" />
    </special>
</collation>
```

The special palaso:sortRulesType value is set to CustomSimple to indicate Toolbox style sorting. This collation element is how the above simple rules would be stored.

2.2.4 Custom ICU Rules

ICU provides the most powerful way of specifying sorting rules, but the syntax can be confusing. ICU rules are based on the Unicode Collation Algorithm described at http://www.unicode.org/unicode/reports/tr10/. The full list of ICU rules is described at http://userguide.icu-project.org/collation/customization. A convenient online test platform for testing ICU rules is available at http://demo.icu-project.org/icu-bin/locexp?_=root&d_=en&x=col. This site also provides a way to get default rules for most majority languages.

See FieldWorks Help, and Section 8 of http://fieldworks.sil.org/wp-content/TechnicalDocs/ICU%20and%20writing%20systems.doc for more details on ICU rules for common usage.

The following rules sort ‘n with tilde’ as a primary sort after ‘n’ and the ‘ll’ digraph as a primary sort after ‘l’. It also ignores hyphen.

```
&n<ñ<<<Ñ
&l<ll<<<Ll<<<LL
&[last tertiary ignorable] = '-'
```

In the LDML file an ICU collation is indicated by the CustomICU value of palaso:sortRulesType. The rules themselves are stored using the standard LDML rule syntax.

```
<collation>
    <rules>
        <reset>n</reset>
        <p>ñ</p>
        <t>Ñ</t>
        <reset>l</reset>
        <p>ll</p>
        <t>Ll</t>
        <t>LL</t>
        <reset>
            <last_tertiary_ignorable />
        </reset>
        <i>-</i>
    </rules>
    <special xmlns:palaso="urn://palaso.org/ldmlExtensions/v1">
        <palaso:sortRulesType value="CustomICU" />
    </special>
<collation>
```

3 Special additions to the LDML standard

The LDML standard provides the special element to allow users to customize the standard as needed for their applications. One palaso extension was already mentioned in the previous section to identify the type of sorting.

Another special palaso element provides various information for FieldWorks and WeSay programs, including the language abbreviation to use in the FieldWorks detail view, the language name, default font, and the palaso version for the LDML data. FieldWorks currently crashes if the wrong palaso version number is present in a file. Here is a sample illustrating this special element.

```
<special xmlns:palaso="urn://palaso.org/ldmlExtensions/v1">
    <palaso:abbreviation value="Ger" />
    <palaso:defaultFontFamily value="Times New Roman" />
    <palaso:languageName value="German" />
    <palaso:version value="2" />
</special>
```

FieldWorks also adds a special element that includes other information needed in the writing system properties dialog not covered by other LDML elements. This includes whether Graphite is enabled, the special features enabled on Graphite fonts, the full names for script, region, variant, keyboard information, etc. Here is an example from an LDML file demonstrating this special element.

```
<special xmlns:fw="urn://fieldworks.sil.org/ldmlExtensions/v1">
    <fw:graphiteEnabled value="False" />
    <fw:matchedPairs value="&lt;?xml version=&quot;1.0&quot;
encoding=&quot;utf-16&quot;?&gt;
&lt;MatchedPairs&gt;
  &lt;pair open=&quot;(&quot; close=&quot;)&quot;
permitParaSpanning=&quot;false&quot; /&gt;
&lt;/MatchedPairs&gt;" />
    <fw:punctuationPatterns value="&lt;?xml version=&quot;1.0&quot;
encoding=&quot;utf-16&quot;?&gt;
&lt;PunctuationPatterns&gt;
  &lt;pattern value=&quot;.) &quot; context=&quot;WordFinal&quot;
valid=&quot;true&quot; /&gt;
  &lt;pattern value=&quot;), &quot; context=&quot;WordFinal&quot;
valid=&quot;true&quot; /&gt;
  &lt;pattern value=&quot;? &quot; context=&quot;WordFinal&quot;
valid=&quot;true&quot; /&gt;
  &lt;pattern value=&quot;. &quot; context=&quot;WordFinal&quot;
valid=&quot;true&quot; /&gt;
&lt;/PunctuationPatterns&gt;" />
    <fw:quotationMarks value="&lt;?xml version=&quot;1.0&quot;
encoding=&quot;utf-16&quot;?&gt;
&lt;QuotationMarks LangUsedFrom=&quot;de&quot;
ContinuationType=&quot;RequireAll&quot;
ContinuationMark=&quot;Opening&quot;&gt;
  &lt;Levels&gt;
    &lt;Pair Opening=&quot;„&quot; Closing=&quot;“&quot; /&gt;
    &lt;Pair Opening=&quot;‚&quot; Closing=&quot;‘&quot; /&gt;
  &lt;/Levels&gt;
&lt;/QuotationMarks&gt;" />
    <fw:validChars value="&lt;?xml version=&quot;1.0&quot;
encoding=&quot;utf-16&quot;?&gt;
&lt;ValidCharacters&gt;
&lt;WordForming&gt;a￼A￼ä￼Ä￼b￼B￼c￼C￼d￼D￼e￼E￼f￼F￼g￼G￼h￼H￼i￼I￼j￼J￼k￼K￼l￼L￼
m￼M￼n￼N￼o￼O￼ö￼Ö￼p￼P￼q￼Q￼r￼R￼s￼S￼ß￼t￼T￼u￼U￼ü￼Ü￼v￼V￼w￼W￼x￼X￼y￼Y￼z￼Z￼'￼-
&lt;WordForming&gt;
  &lt;Numeric&gt;0￼1￼2￼3￼4￼5￼6￼7￼8￼9&lt;/Numeric&gt;
  &lt;Other&gt;
￼U+0020￼,￼;￼:￼!￼?￼.￼‘￼,￼&quot;￼“￼„￼(￼)￼[￼]￼{￼}￼/￼&amp;amp;&lt;/Other&gt;
&lt;/ValidCharacters&gt;" />
    <fw:windowsLCID value="1031" />
</special>
```

It’s especially hard to read because it contains xml strings quoted in a way that allows it to be stored in an XML attribute. This includes information specified in the Valid Characters and Punctuation Usage dialogs in FieldWorks.

4 Palaso approach to custom subtags

In order to support minority languages in our work, at times we need to specify languages, scripts, regions, and variants that are not in the official list. In order to accomplish this, we provide a way for users to define their own languages, scripts, regions, and variants. At the same time we want to maintain valid subtags for the LDML standard. We do this by making use of some subtags that are reserved for private use. The ones we use are

- custom language: qaa

- custom script: Qaaa
- custom region: QM
- custom variant: x

All user-defined subtags are separated from official subtags by a single x subtag. If there is a custom language, it will be the first subtag following the x and the official language code will be qaa. If there is a custom script, it will be the next subtag following the x and the official script code will be Qaaa. If there is a custom region, it will be the next subtag following the x and the official region will be QM. Any additional subtags following the x are custom variants. Standard variants occur before the x. Palaso code provides common methods for storing and accessing these custom tags so that applications such as FieldWorks and WeSay do not need to be concerned with the details.

Here are some examples demonstrating how this works.

- qaa-x-kal : A make-believe language called Kalaba. FieldWorks uses the first 3 letters of unregistered language names as the custom language code.
- fr-fonipa-x-etic : French using IPA to represent a custom phonetic script.
- en-QM-x-eb : English Ebonics dialect, using the region code to specify a custom dialect.
- zh-CN-x-py : Chinese Mandarin using a custom variant py to indicate Pinyin. (There is now a registered pinyin variant that could be used as well which would then be zh-CN-pinyin.)
- fr-Zxxx-x-audio : A French 'writing system' that holds a file name for an audio recording. Zxxx is a registered script indicating a code for an unwritten document. Audio is a custom variant WeSay and FieldWorks use with Zxxx to indicate an audio recording.
- fr-Qaaa-x-old : French in a custom script called old.
- qaa-Qaaa-QM-fonipa-x-kal-old-dial-etic-v1 : A custom dialect (dial) of Kalaba using a custom script (old) and a custom variant (etic-v1) indicating version 1 of a phonetic script in IPA. This doesn't make a lot of sense, but does demonstrate how all of the pieces can be put together in a standard way.

5 Locating writing systems in FieldWorks

5.1 Writing system tags in FieldWorks data

A FieldWorks .fwdata file is an XML file that stores FieldWorks data. Almost all strings in FieldWorks identify the writing system for a string or portion of a string using a locale identifier. Locale identifiers are used in a few other places as well, such as identifying the writing system for a reversal index and giving a list of current analysis writing systems. The actual details of a writing system are stored in a LDML file in the local writing system store. Without a corresponding LDML file, FieldWorks knows nothing about a writing system other than the locale identifier.

There are only a few contexts where locale identifiers occur in a fwdata file. This makes it fairly easy to search for them or change them manually via an editor such as ZEdit or a CC table.

The following contexts are where they can occur:

ws="xxx" : This identifies the writing system for strings or parts of strings. It occurs as <Run ws="xxx">, <AStr ws="xxx">, or <AUni ws="xxx">. It can also be used in a style as <WsProp ws="xxx".

>xxx< : This is a reference to a writing system for an object such as a ReversalIndex, a CmPossibilityList, etc. It occurs as <WritingSystem>xxx</WritingSystem>.

>xxx yyy< : This is a space-delimited list of writing systems. It currently occurs as <Uni>xxx yyy</Uni> in AnalysisWss, CurVernWss, CurAnalysisWss, CurPronunWss, and VernWss elements that are always together in the fwdata file as part of the languageproject element.

lang="xxx" or lang=&quot;xxx&quot; : This form may occur in a <Uni> element inside a <LiftResidue> element where data may be stored during a LIFT import if there isn't a recognized place to store the data in the FieldWorks model. The user never sees this information, but it is used during a LIFT export to return any LIFT data that is not used by FieldWorks.

One caution when manually changing locale identifiers in a fwdata file is the possibility of ending up with duplicates in AUni or AStr properties. If a duplicate AUni is found during loading only the first one will show up in the property. If a duplicate AStr element is found, FieldWorks will issue an error message and crash without opening the file. It is also slightly possible to create duplicate reversal indexes or corresponding parts of speech lists.

5.2 Writing system definitions

The details for each writing system are stored in a LDML file. A FieldWorks project is a folder that contains the .fwdata file (or .fwdb db4o database if sharing is enabled) as well as subfolders for storing project settings, linked files, writing systems, etc. All writing systems used in the fwdata file are defined by LDML files in the WritingSystemStore directory under the project folder.

In addition to the project WritingSystemStore directory, there is also a global store at c:\ProgramData\SIL\WritingSystemStore (c:\Documents and Settings\All Users\ApplicationData\SIL\WritingSystemStore on XP). When a FieldWorks project is opened, any writing systems for that project that are missing from the global store will be copied to the global store. Whenever a user modifies a writing system, the changes are stored in the local LDML file and then it is copied to the global store.

During installation, FieldWorks installs master LDML files for a few major languages in c:\Program Files\SIL\FieldWorks 7\Templates. At the end of installation, these master files are copied into the global store, replacing any that are already there.

When a project is opened, FieldWorks checks to see if the global store copy of each local LDML file has changed since the project was last opened. If it has, then FieldWorks asks the user whether they want to copy the global changes to the local store, or keep their current writing systems as they last used them.

6 Migrating older writing systems

Various versions of FieldWorks 7.0 attempted to migrate old writing systems from FieldWorks 6.0, where the ICU specifications we used were more lenient than the LDML standard. The migration attempts to convert any locale identifier from 6.0 to a legal 7.0 identifier. In most cases the conversion works successfully, but there may be some exceptions.

A further complication is that cmn, pes, and arb were not converted as described in Section 2.1 when users created them before FieldWorks 7.0.4. In FieldWorks 7.0.4, 7.0.5, and 7.0.6 attempts were made to correct this situation, but because these minor releases were not able to do a data migration, the locale identifiers in the fwdata file may not match the LDML file, although coding changes allowed these versions to use the LDML file even though it didn’t match the internal locale identifiers.

In FieldWorks 7.1, we implemented the custom approach described in Section 4 and also included a data migration to try to correct all of the previous problems with partial migrations.

In FieldWorks 7.1 (migration 7000044) when an older file is opened, we migrate all of the LDML files in the local store as well as the global store to be compatible with the new approach. This includes updating the palaso:version value to 2. We also do a migration of the locale identifiers in the fwdata file to consistently match the revised LDML files.

During this process, if two writing systems in the fwdata file converge to the same identifier during the process (e.g., cmn and zh-CN both converge to zh-CN), then one of the writing system identifiers will have –dupl appended to the variant. Or if that already exists, an incrementing digit will be added (e.g., -dupl2). This allows FieldWorks to work as it did in the older version without crashing. But these duplicates should be removed if they are found. This is especially true if you are using a localized version of Chinese, since the localizations in the file may end up as zh-CN-dupl, which won’t match the zh-CN Microsoft requires. The best way to do these conversions is to use the writing system merge capability from the Project Properties dialog described in the next section.

7 Writing system cleanup

When a writing system is hidden in the Project Properties dialog, it is removed from the AnalysisWss, CurVernWss, CurAnalysisWss, CurPronunWss, or VernWss elements so that it no longer shows in the Project Properties dialog, and any data will not be shown in views. All data remains in the fwdata file so that if the writing system is added again, all data will again be visible.

When a writing system is deleted (right-click option) in the Project Properties dialog, after a confirmation, it will remove all data for this writing system from the fwdata file and delete the local LDML file. It currently does not remove the corresponding global LDML file, so it is still possible to add it back to the project, but any data in that writing system will be gone.

When a writing system is merged (right-click option) in the Project Properties dialog, every occurrence of the original locale identifier will be changed to the target locale identifier and the original LDML file will be deleted from the local store. Note, if the target writing system is hidden, it currently won't be offered as a target for merging. To solve that, Add the writing system first to make it visible, then do the merge. Again, a merge does not affect the global store. If this merge occurs in a multistring or multiunicode property where the target already exists, the duplicate must be eliminated. If the source and target strings are identical, the duplicate is just deleted. If the source and target are not identical, then the source string is appended to the target string with a colon separator.

8 Use of ICU

In FieldWorks 6 and earlier, an ICU locale was needed for each FieldWorks writing system. This required rebuilding a binary ICU locale from source code whenever a change was made to the writing system. In FieldWorks 7 we no longer create new locales for writing systems, but feed in a custom set of rules for sorting rather than using the built-in rules for the locale. As a result, we don't need to create new locales, and this removes the earlier constraints we had with locked ICU files used by one program blocking writing system changes in another program.

8.1 Custom characters

There is one case where we still need to recompile main ICU files. This is when a user needs to specify custom characters, such as new PUA characters. The standard ICU files delivered with FieldWorks already specify SIL code points contained in fonts such as Doulos SIL. Instead of making these changes from within Flex and TE as in the past, in FieldWorks 7 a separate program, UnicodeCharEditor.exe must be run to define characters. This program should only be run when other FieldWorks applications are not running. On an end-user machine you can get to this using Start...FieldWorks 7...Unicode Character Properties Editor. When custom characters are added, the definitions are stored in c:\ProgramData\SIL\CustomChars.xml. Whenever custom character changes are made, in addition to saving information in CustomChars.xml, source files in the Icu40\data directory are modified, then compiled into binary files in Icu40\icudt40l, making them available to FieldWorks programs.

When FieldWorks is uninstalled, CustomChars.xml is not removed. During installation, as part of MigrateSqlDbs.exe, UnicodeCharEditor.exe is run with an –i option to recover any custom character definitions from FW 6.0 or CustomChars.xml and reinstall them into ICU.

9 What to do in Flex if it doesn't list my language

9.1 Can't find my language in Flex language chooser

While in the Select Language dialog, if you have a list of languages showing, but none is what you need, click "None of the above" at the bottom of the list. This will put your initial query in the Chosen Name edit box. Correct or type in the name you want to use for your language in this edit box. Click OK to close the Select Language dialog. Select, or type any other subtag (script region, variant) names if they are necessary to distinguish this writing system from another for the same language, then click OK to close the

Writing System Properties. Flex will create a custom language identifier that will work for your language even though it is not currently available.

9.2 I want to change my language

If you had to use a custom language tag at one point in the life of your project, but in the current version of Flex it is now available language chooser, you can easily change to use this standard identifier. Use Format...Setup Writing Systems, click Modify for the language you want to change, click the Change button at the top right, find your language in the picker, click OK to close the Select Language dialog. That should change all writing systems you have for this language to the new code language code. If there are other subtags (script, region, variant) that are no longer relevant, change these, then click OK to choose Writing System Properties.