

Common XML Locale Repository

Dr. Mark Davis Steven R. Loomis
mark.davis@us.ibm.com *srloomis@us.ibm.com*

Copyright © 2003 IBM Corporation

1. Introduction

Unicode has provided a foundation for communicating textual data. However, the locale-dependant data used to drive features such as collation and date/time formatting may be incorrect or inconsistent between systems. This may not only present an irritating user experience, but cause problems when data is transferred between systems.

The Common XML Locale Repository is a step towards solving these problems, by providing an interchange format for locale data and developing a repository of such data available.

In this document, a “Locale” is an identifier that refers to a set of linguistic and cultural preferences. Traditionally, the data associated with such a locale provides support for formatting and parsing of dates, times, numbers, and currencies; for the default units of currency; for measurement units, for collation (sorting), plus translated names for time zones, languages, countries, and scripts. They can also include text boundaries (character, word, line, and sentence), text transformations (including transliterations), and support for other services. Because locale data changes over time, the data must be versioned to provide stability.

Examples of platforms with their own locale data are ICU¹, OpenOffice.org, and POSIX and POSIX-like operating systems such as Linux, Solaris, and AIX.

2. Common XML Locale Repository Group

OpenI18N² is formerly known as "Li18nux" and is a workgroup of the Free Standards Group³. It is a voluntary workgroup which develops free and open standards for internationalization. The Linux Application Development Environment, or LADE, subgroup is responsible for issues regarding API support or base level application environment requirements. The Common XML Locale Repository project, in turn, is sponsored by the LADE subgroup.

Common XML Locale Repository

3. Objectives

The goals of this project are to:

1. *Produce a common format for the encoding and interchange of locale data, as an XML locale markup language specification.*
2. *Collect locale data from a variety of platforms.*
3. *Make a repository of such data available for download by the public.*
4. *Implement a process whereby data is determined to be valid, and labeled in the repository as such.*
5. *Provide enablement for Web Services⁴ as defined by the W3C Web Services task force. Allow web services to have consistent locale information regardless of underlying platform, operating system, or software version. Provide a way to access locale data that a client-specified platform would expect (for example, a server requesting Windows data for a locale).*

4. The Locale Repository

Locale data will be collected from various sources, and made available via the repository. Currently, the Repository exists as a source code control database. Data will be accessible via HTTP as well, to enable automated tools for importing portions of the repository into different environments.

The repository distinguishes from data gathered from different platforms, for example, ICU, openOffice.org, and Solaris.

The repository is designed to allow access to the locale data for use in application environments, for example to create a set of POSIX locales based on data in the repository, and also to enable comparisons of data between platforms.

Tools have already been developed which can compare between multiple platforms in the repository. A process will be defined whereby data is submitted to experts for vetting, and then marked as such in the repository.

Discrepancies found when comparing data collected as of this writing only reinforce the need for such a common repository. Some of these differences seem to be errors, while others may indicate regional differences, or even personal preferences of the people verifying/collecting the data. The use and spelling of abbreviations vary somewhat from platform to platform, as do punctuation and case.

5. Locale Data Markup Language

LDML⁵ is the XML format for specifying locale data in the repository. Each locale's data is stored in a separate XML file, for example `fr_BE.xml` or `en.xml`, and the top level element is named `<localeData>`.

Common XML Locale Repository

5.1. Locales and the <identity> Element

Locales consist of four parts: the language, the territory, the variant, and finally any locale options. Only the language code is required.

Here are some example locales:

Locale	Description
En	English
fr_BE	French in Belgium
de_DE	German in Germany
sv_FI_AL	Swedish in Finland, Åland region.
de_DE@collation=phonebook,currency@pre-euro	German in Germany, with Collation according to phonebook order, and Currency in pre-Euro form.

Language and Territory⁶ codes follow ISO-639⁷ and ISO-3166⁸, respectively. Two-letter codes are used where they exist, otherwise three-letter codes are used. (See also the OpenI18N convention on locale naming⁹, and RFC 3066¹⁰ standards for language tagging.)

The variant codes specify particular variants of the locale, typically with special options. For example, the variant “AL” specifies Åland, an autonomous region of Finland.

Options are key-value pairs which request alternate forms of the locale. The currently defined types are `collation`, `currency`, and `calendar`.

Below is an example <identity> element, which identifies the locale data as being part of the `sv_FI_AL` locale (that is, `sv_FI_AL.xml`).

```
<localeData>
  <identity>
    <version number="1.1">Various notes and changes</version>
    <generation date="2002-08-28"/>
    <language type="sv"/>
    <territory type="FI"/>
    <variant type="AL"/>
  </identity>
</localeData>
```

5.2. Inheritance

Besides taking up space in the Repository, redundant data adds needlessly to the maintenance burden. LDML relies on an inheritance model, whereby the resources are collected into bundles, and the bundles organized into a tree. Data for the many Spanish locales does not need to be duplicated across all of the countries having Spanish as a national language. Instead, common data is collected in the Spanish language locale, and territory locales only need to supply differences.

The parent of all of the language locales is a generic locale known as root. Wherever possible, the resources in the root are language and territory neutral.

Common XML Locale Repository

Given a particular locale id "en_US_someVariant", the search chain for a particular resource is the following:

```
en_US_someVariant → en_US → en → root
```

In some cases, the searching is done within a resource. For example, with calendars (discussed below), all non-Gregorian calendars inherit their data from the Gregorian class.

Where this inheritance relationship is not supported by a target system, such as with POSIX, the data logically should be fully resolved in converting to a format for use by that system, by adding *all* inherited data to each locale data set.

In addition, the locale data does not contain general character properties that are derived from the Unicode Character Database data (UCD¹¹). That data being common across locales, it is not duplicated in the repository. Constructing a POSIX locale from the following data requires use of that data. In addition, POSIX locales may also specify the character encoding, which requires the data to be transformed into that target encoding.

5.3. Aliasing

The contents of any element can be replaced by an alias, which points to another source for the data. The resource is to be fetched from the corresponding location in the other source.

The following example demonstrates a locale "zh_HK" which has a collation element aliased to "zh_TW". Both locales use Traditional Chinese collation, which has a considerable disk footprint.

```
<localeData>
  <identity>
    <language type="zh"/><territory type="HK"/>
  </identity>
  <collations>
    <alias source="zh_TW"/>
  </collation>
</localeData>
```

5.4. type Attribute

Any element may have a type specifier, to indicate an alternate resource that can be selected with a matching type=option in the locale id modifiers, or be referenced by a default element of the form <default type="xxx">. The following example demonstrates multiple elements of different types used to select differing number formats.

```
<numberFormats>
  <default type="scientific"/>
  <numberFormatStyle type="decimal">...</numberFormatStyle>
  <numberFormatStyle type="percent">...</numberFormatStyle>
  <numberFormatStyle type="scientific">...</numberFormatStyle>
</numberFormats>
```

Common XML Locale Repository

The currently defined optional key/type combinations include:

key	type	Description
collation	phonebook	For a phonebook-style ordering (used in German).
	pinyin	Pinyin order for CJK characters
	traditional	For a traditional-style sort (as in Spanish)
	stroke	Stroke order for CJK characters
	direct	Hindi variant
	posix	A "C"-based locale.
currency	euro	For data using the Euro currency
	pre-euro	For data using older currency.
calendar	gregorian	(default)
	arabic	Astronomical Arabic
	chinese	Traditional Chinese calendar
	civil-arabic	Civil (algorithmic) Arabic calendar
	hebrew	Traditional Hebrew Calendar
	japanese	Imperial Calendar (same as Gregorian except for the year, with one era for each Emperor)
	thai-buddhist	Thai Buddhist Calendar (same as Gregorian except for the year)

5.5. *draft and standard Attributes*

Any element may be marked with `draft="true"` to indicate data that has not yet been verified. The following example shows an entire locale which is in draft stage:

```
<localeData draft="true"> ... </localeData>
```

Similarly, the `standard=` attribute denotes any element with data designed to conform to a particular standard. It may be a single string, or a comma separated list.

```
<collation standard="MSA 200:2002"> ...  
  
<dateFormatStyle type="decimal" standard="ISO 8601,  
http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNU  
MBER=26780&ICS1=1&ICS2=140&ICS3=30,DIN 5008">
```

5.6. *Data Access*

Data in the repository can be accessed via `http`. Given a base address for the repository, a URL can be constructed requesting data by version, platform, and locale. For example, if the base URL is "`http://openi18n.org/locale`" and the platform is "`icu`", a URL could be constructed such as:

```
http://openi18n.org/locale/icu/de_DE.xml?version=2.2&currency=pre-euro
```

This URL will request the German (in Germany) locale data, of version 2.2, and it will request that all `<currency>` elements returned have a matching `type="pre-euro"` attribute.

5.7. *<calendars> Element*

This element contains multiple `<calendar>` elements, each of which specifies the fields used for formatting and parsing dates and times according to the given calendar. The month names are identified numerically, starting at 1. The day names are identified with short strings, since there is no universally accepted numeric designation.

Many calendars will only differ from the Gregorian calendar in the year and era values. For example, the Japanese calendar will have many more eras (one for each Emperor), and the years will be numbered within that era. All other calendars inherit from the Gregorian calendar (which must be present), so only the differing data will be present. Calendars are distinguished by the `'class'` attribute, which identifies which class of calendar it is, such as Gregorian, Japanese, and so on.

The following example shows a condensed Gregorian calendar definition, and a portion of the Japanese calendar definition for comparison:

```
<calendars>
  <calendar class="gregorian">
    <monthNames>
      <month type="1">January</month>
      <month type="2">February</month>
    </monthNames>
    <dayNames>
      <day type="sun">Sunday</day>
      <day type="mon">Monday</day>
    </dayNames>
    <eras>
      <eraAbbr>
        <era type="0">BC</era>
        <era type="1">AD</era>
      </eraAbbr>
    </eras>
    <dateFormats>
      <default type="medium"/>
      <dateFormatStyle type="full">
        <dateFormat>
          <pattern>EEEE, MMMM d, yyyy</pattern>
        </dateFormat>
      </dateFormatStyle>
      <dateFormatStyle type="medium">
        <default type="DateFormatsKey2">
          <dateFormat type="DateFormatsKey2">
            <pattern>MMM d, yyyy</pattern>
            <displayName>DIN 5008 (EN 28601)</displayName>
          </dateFormat>
          <dateFormat type="DateFormatsKey3">
            <pattern>MMM dd, yyyy</pattern>
          </dateFormat>
        </default>
      </dateFormatStyle>
    </dateFormats>
  </calendar>
```

Common XML Locale Repository

```
<calendar class="japanese">
  <eras>
    <eraAbbr>
      <era type="0">Taika</era>
      <era type="1">Hakuchi</era>
    </eraAbbr>
  </eras>
</calendar>
</calendars>
```

5.8. **<numbers> Element**

This element supplies information for formatting and parsing numbers and currencies. It has three sub-elements: `<symbols>`, `<numberFormats>`, and `<currencies>`.

The `<symbols>` element gives information about the textual representation of individual components of a formatted number, such as digits, separators, and signs.

```
<symbols>
  <decimal>./</decimal>
  <group>,</group>
  <list>;</list>
  <percentSign>%</percentSign>
  <nativeZeroDigit>0</nativeZeroDigit>
  <patternDigit>#</patternDigit>
  <plusSign>+</plusSign>
  <minusSign>-</minusSign>
  <exponential>E</exponential>
  <perMille>%</perMille>
  <infinity>∞</infinity>
  <nan>_</nan>
</symbols>
```

Patterns for formatting and parsing numbers are contained under the `<numberFormats>` element, which contains one or more `<numberFormat>` elements. These are distinguished by the `type` attribute. Common types are decimal, percent, scientific, and currency. The international currency symbol, ¤ , is replaced with the national currency symbol located in the appropriate `<currencies>` element.

```
<numberFormats>
  <numberFormatStyle type="decimal">
    <numberFormat type="long">
      <pattern type="positive">#,##0.###</pattern>
      <pattern type="negative">-#,##0.###</pattern>
    </numberFormat>
  </numberFormatStyle>
  <numberFormatStyle type="percent">
    <numberFormat type="short">
      <pattern type="positive">#,##0%</pattern>
    </numberFormat>
  </numberFormatStyle>
```

Common XML Locale Repository

```
<numberFormatStyle type="currency">
  <numberFormat type="medium">
    <special owner="http://www.openoffice.org"
      msgid="FixedFormatstype9"
      usage="FIXED_NUMBER" formatindex="4"/>
    <pattern type="positive">Q #,##0.00;</pattern>
    <pattern type="negative">(Q #,##0.00)</pattern>
  </numberFormat>
</numberFormatStyle>
</numberFormats>

<currencies>
  <default type="USD" />
  <currency type="USD">
    <displayName>dollar</displayName>
    <symbol>$</symbol>
  </currency>
  <currency type="JPY">
    <displayName>yen</displayName>
    <symbol>¥</symbol>
  </currency>
</currencies>
```

5.9. *<collations> Element*

The `<collations>` element contains one or more `<collation>` elements, and provides information about linguistic collation (sorting) of text. The base (root) locale is defined to have collation behavior according to the Unicode Collation Algorithm (UTS #10)¹², and all other locales have collation rules which are defined in terms of tailorings (deltas) relative to the UCA.

Below is a partial example taken from the Swedish tailorings, which defines characters that sort following 'Z'.

```
<collation>
  <base UCA='3.1.1'>
  □ <settings caseLevel="on" />
  □ <rules>
    <reset>Z</reset>
    <p>æ</p>
    <t>Æ</t>
    <t>aa</t>
    <t>aA</t>
    <t>Aa</t>
    <t>AA</t>
    ...
  </rules>
</collation>
```

5.10. *<special> Element*

The `<special>` element may occur anywhere, and allows for arbitrary additional annotation and data that is platform-specific. It has one required attribute, which specifies the owner of the special data. For uniqueness, that owner is specified by means of a URL.

The following example demonstrates the inclusion of transform (transliteration) data, which is used by ICU, but not part of the LDML spec.

```
<special owner="http://oss.software.ibm.com/icu/">
  <transforms>
    <transform type="Latin">
      &#x03B1; &lt; a ; &#x0391; &lt; A ;
      &#x03B2; &lt; v ; &#x0392; &lt; V ;
    </transform>
  </transforms>
</special>
```

5.11. *Other Elements*

For more detail about these elements, please see the LDML specification.

<displayName>

a translated name that can be presented to users when discussing the particular service, for example, in a GUI

<localizedPatternChars>

translated replacements for date format pattern characters (e.g. 'm' for month, etc.) for display use

<timeZoneNames>

translated time zone names

<delimiters>

common delimiters for bracketing, such as quotation marks

<encodings>

information helpful in picking among character encodings used for the locale

<layout>

specifies general document-layout features

<localeDisplayNames>

translated names for scripts, languages, countries, and variants

<measurement>

specifies the measuring system in use, for example, "metric"

6. Design Decisions

- Rather than use attributes, LDML often uses elements in many situations. For example, rather than have multiple `<numberFormat>` elements, all patterns could be represented with attributes:

```
<numberFormats decimalFormat="0.##" percentFormat="#.##0%">
```

Although this appears to be more compact, there are a number of difficulties.

- Inheritance becomes more complex, because not only elements, but individual attributes must be processed.
- Programmatic processing of the data is difficult, because attribute names must be special cased whereas multiple elements are easier to enumerate.
- Attribute values are normalized (see XML¹³), and therefore line breaks and spaces would be collapsed, changing the meaning of the data.

7. Open Issues

- The LDML specification is currently in Draft stage, pending OpenI18N steering committee final approval. (As this paper is based on a working version of that draft, elements and attributes are subject to change.)
- The possibility of different input (parsing) and output (formatting) symbols has been discussed, to allow greater flexibility of user input.
- The process for vetting data, and the mechanism for labeling data as valid, have not been defined yet. The current proposal is to identify data according to who (what organization) has vetted it. Data which is not marked as vetted, should be considered experimental. Another open issue is what to do with locales which are partially correct, and whether the valid label should be applied on a per-locale or a per-element basis.
- The versioning and release process of the repository has not been finalized.
- Plans are to obtain OpenI18N Steering Committee approval of the draft spec in early 2003. Planned call-for-public-review will go out by the first quarter of 2003.
- First release (1.0) is planned by the second or third quarter, 2003. More details to follow.
- Open up the process for future collaborations. For now, the website, CVS repository for read access, and newsgroups are available.

Endnotes and References

- ¹ ICU — International Components for Unicode: <http://oss.software.ibm.com/icu/>
- ² OpenI18N site: <http://www.openi18n.org>
- ³ Free Standards Group site: <http://www.freestandards.org>
- ⁴ W3C Web Services Scenarios: <http://www.w3.org/TR/ws-i18n-scenarios/>
- ⁵ LDML specification: <http://oss.software.ibm.com/icu/locale/> (This paper refers to the December 31, 2002 version of the LDML specification.)
- ⁶ Note: The territory code is sometimes referred to as the “country code”, although not all territories covered by ISO-3166 are actually countries.
- ⁷ ISO-639: http://www.chemie.fu-berlin.de/diverse/doc/ISO_639.html
- ⁸ ISO-3166: <http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html>
- ⁹ OpenI18N Locale Naming Guide: <http://www.openi18n.org/localenameguide/>
- ¹⁰ RFC-3066: <http://www.ietf.org/rfc/rfc3066.txt>
- ¹¹ UCD: <ftp://ftp.unicode.org/Public/UNIDATA/UnicodeCharacterDatabase.html>
- ¹² UTS #10: Unicode Collation Algorithm <http://www.unicode.org/reports/tr10/>
- ¹³ XML: <http://www.w3.org/TR/REC-xml#AVNormalize>