The Common Locale Repository - Update

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

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 prevent accurate data transfer.

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 ICU[1], OpenOffice.org, and POSIX and POSIX-like operating systems such as Linux, Solaris, and AIX.

2. Common XML Locale Repository Group

OpenI18N[2], formerly known as "Li18nux", is a workgroup of the Free Standards Group[3]. It is a volunteer organization, which develops free and open standards for internationalization. The Linux Application Development Environment, or LADE,

[1] ICU — International Components for Unicode: http://oss.software.ibm.com/icu/
[2] Openi18n - http://www.openi18n.org
[3] Free Standards Group site: http://www.freestandards.org

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.

3. Objectives

The goals of this project are to:

a) *Produce a common format for the encoding and interchange of locale data, as an XML locale markup language specification.*
b) *Collect locale data from a variety of platforms.*
c) *Make a repository of such data available for download by the public.*
d) *Implement a process whereby data is determined to be valid, and labeled in the repository as such.*
e) *Provide enablement for Web Services*[4] *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 collected from various sources is 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 among 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 exist which can compare between multiple platforms in the repository, and comparison charts have been produced based on that data. A process will be defined whereby submitted data is given to experts for vetting, and then marked as such in the repository. This vetting process, and the comparison charts, are described in detail in Appendices A and D, respectively.

Actual discrepancies found when comparing the data have only reinforced 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.

[4] W3C Web Services Scenarios: http://www.w3.org/TR/ws-i18n-scenarios/

The common repository of vetted, verified data is intended to be used by platforms as their single source for data. When this is accomplished, the goal of platform consistency can be realized.

The Common Locale Data Repository - Version 1.0 has been approved for release by the OpenI18n steering committee as of January 16, 2004.

5. Locale Data Markup Language

Locale Data Markup Language[5] is the XML format for specifying locale data in the repository. Version 1.0 of this specification was released June 24th, 2003. This paper refers to version 1.0, plus the latest errata available from the same website.

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 `<ldml>`.

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=CHF`	German in Germany, with Collation according to phonebook order, and Swiss Franc currency.

Language and Territory[6] codes follow ISO-639[7] and ISO-3166[8], respectively. Two-letter codes are used where they exist, otherwise three-letter codes are used. (See also the OpenI18N convention on locale naming[9], and RFC 3066[10] 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`).

[5] Locale Data Markup Language: http://oss.software.ibm.com/icu/locale/
[6] 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

```
<ldml>
 <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>
</ldml>
```

5.2. Inheritance

Besides taking up space in the Repository, redundant data adds needlessly to the maintenance burden. The Locale Data Markup Language 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.

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 (UCD[11]). 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.

[11] UCD: ftp://ftp.unicode.org/Public/UNIDATA/UnicodeCharacterDatabase.html

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

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>
```

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.
calendar	gregorian	(default)
	islamic	Astronomical Islamic
	chinese	Traditional Chinese calendar
	islamic-civil	Civil (algorithmic) Islamic calendar
	hebrew	Traditional Hebrew Calendar
	japanese	Imperial Calendar (same as Gregorian except for the year, with one era for each Emperor)
	buddhist	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:

```
<ldml draft="true"> … </ldml>
```

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&collation=phonebook
```

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

5.7. Escaping Characters

Extra syntax is required to represent Unicode code points which XML cannot otherwise represent, such as white space, control characters, and NULL. For example, the NULL character cannot be represented by an entity such as "`&#x0000;`", which is not legal XML. In a Locale Data Markup Language XML document, this may be written as follows:

```
<cp hex="0">
```

5.8. <dates> Element

This top-level element contains information regarding the formatting and parsing of dates and times. <calendars>, <localizedPatternChars> and <timeZoneNames>. See the Locale Data Markup Language specification for more details on the latter two

5.8.1. <localizedPatternChars> and <timeZoneNames>

This sub-element contains translated replacements for date format pattern characters (e.g. 'm' for month, etc.) for display use.

5.8.2. <timeZoneNames>

This sub-element contains translated names of time zones.

5.8.3. <calendars>

This sub-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 differ from the Gregorian calendar only in the year and era values. For example, the Japanese calendar has many more eras (one for each Emperor), and the

years are 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 '`type`' 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:

```
<dates>
 <calendars>
    <calendar type="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"/>
            <dateFormatLength type="full">
                <dateFormat>
                    <pattern>EEEE, MMMM d, yyyy</pattern>
                </dateFormat>
            </dateFormatLength>
            <dateFormatLength 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>
            </dateFormatLength>
        </dateFormats>
        <timeFormats>
           …
             <pattern>h:mm:ss</pattern>
           …
        </timeFormats>
    </calendar>
    <calendar class="japanese">
        <eras>
            <eraAbbr>
                <era type="0">Showa</era>
                <era type="1">Heisei</era>
                …
            </eraAbbr>
        </eras>
```

```
  </calendar>
</calendars>
```

5.9. <numbers> Element

This element supplies information for formatting and parsing numbers 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 <decimalFormats>, <scientificFormats>, <percentFormats>, and <currencyFormats> elements. Each of these elements has a similar structure. For example, <decimalFormats>, contains one or more <decimalFormatLength> elements. These are distinguished by the type attribute, which describes a pattern length such as short, medium, or long.

```
<decimalFormats>
  <default type="long">
  <decimalFormatLength type="long">
    <decimalFormat>
      <pattern>#,##0.###;-#,##0.###</pattern>
    </decimalFormat>
  </decimalFormatLength>
  <decimalFormatLength type="short">
    <decimalFormat>
      <pattern>#,##0;-#,##0</pattern>
    </decimalFormat>
  </decimalFormatLength>
 </decimalFormats>
```

The semicolon ";" separates positive and negative patterns.

```
<currencyFormats>
  <currencyFormatLength type="medium">
   <currencyFormat>
      <special xmlns:ooo="http://www.openoffice.org">
            <ooo:msgid="FixedFormatstype9"/>
            <ooo:usage="FIXED_NUMBER" formatindex="4"/>
      </special>
      <pattern>¤ #,##0.00;(¤ #,##0.00)</pattern>
   </currencyFormat>
  </curencyFormatLength>
</currencyFormats>
```

In the currency case, the international currency symbol, ¤, is replaced with the national currency symbol located in the appropriate <currencies> element. Information about which currency is the default for a given locale is not stored in the locale, but is in a separate "supplemental" data component.

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

5.10. <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 'Y'. Y and ü have a secondary (accent) difference, ü and Ü have a tertiary (case) difference.

```
<collations>
 <collation type="standard" >
  <settings caseLevel="on"/>
  <rules>
   <reset>Y</reset>
   <s>ü</s>
   <t>Ü</t>
   ...
  </rules>
 </collation>
</collations>
```

5.11. <special xmlns:yyy="xxx"> Element

The <special> element may occur anywhere, and allows for arbitrary additional annotation and data that is platform-specific. It has one required attribute, xmlns, which specifies the unique XML namespace of the special data.

The following example demonstrates the inclusion of transform (transliteration) data, which is used by ICU, but not part of the Locale Data Markup Language spec. The DOCTYPE element must be at the top of the locale, and specifies that the "ldmlICU.dtd" definition must be considered for parsing.

[12] UTS #10: Unicode Collation Algorithm http://www.unicode.org/reports/tr10/

```
<!DOCTYPE ldml SYSTEM http://www.openi18n.org/spec/ldml/1.0/ldml.dtd" [
  <!ENTITY % icu SYSTEM
"http://www.openi18n.org/spec/ldml/1.0/ldmlICU.dtd">
   %icu;
]>

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

5.12. Other Elements

For more detail about these elements, please see the Locale Data Markup Language specification.

<displayName>

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

<delimiters>

common delimiters for bracketing, such as quotation marks

<characters>

information about the characters commonly used in the locale, and other information helpful in picking among character encodings

<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, the markup language often uses elements. 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 XML[13]), and therefore line breaks and spaces would be collapsed, changing the meaning of the data.

7. Open Issues

- The possibility of different input (parsing) and output (formatting) symbols has been discussed, to allow greater flexibility of user input.
- The design process must be opened up for future collaborations. For now, the website, CVS repository for read access, and newsgroups are available.

[13] XML: http://www.w3.org/TR/REC-xml#AVNormalize

Appendix A: Data Collection and Vetting Process

(Baldev S. Soor)

1. Data Collection Process

When gathering data for a country and language, it is important to have multiple sources for that data in order to weed out any bias. Contributions will be invited. A separate document details the format of the contributions. Contributors can be individuals or organizations.

The ICU locale data will be taken as the initial source. The goal is to come up with a reasonable set of data within a short time; the expectation is that it will be modified and improved, in successive versions, by more input from the open-source community and experts resident in the countries.

When using existing data, we may have to extrapolate from the available sources because there may not be a direct match between the XML format required, and that particular source. Members are encouraged to use local contacts to help with the extrapolation.

2. Data Scrubbing Process

Once data for a country and language has been received, the data from the different sources will be compared to show agreements and differences. The data differences will be resolved.

2.1 Resolution Procedure

Data contributed to the group from different sources may be in conflict. For example, a contribution on abbreviated month names may show each abbreviated name ending with a period and another contribution for the same abbreviated month names may not show the trailing period. A resolution process will be used to resolve these conflicts.

Note that there are two types of data in the repository:

a) Contributor specific data: The contributor can be an individual or an organization. The group will not make any changes to the data. Changes to the data are up to the contributing party. The only request is that all changed data be versioned, and the Version Numbering Scheme be used.

b) Common Data: This is decided by the group. Normally this would be by consensus of the members attending the regular meetings using the following process:

   i. Each organization (company or government) that has contributed a substantially complete resource tree (substantial amount of locale data for a number of locales) can designate up to 3 voting members

   ii. All proposals must be in writing on the localerepository mailing list. These proposals can be amended in a meeting, however.

   iii. When a proposal is introduced for the first time in a meeting, any voting member can ask for the decision to be delayed to the next meeting, to allow time for study.

iv. Any proposal can be passed by consensus; if no voting member calls for a vote, the proposal passes.

v. If a proposal is called for a vote, the proposal is passed by a simple majority of voting members.

vi. When more than two options are proposed, approval voting is used to decide among the options. With approval voting, each voting member checks off all the options that are acceptable to them. (Thus more than one option can be selected by each voting member.) The option with the largest total checks wins. [14]

Members are encouraged to use local language and country contacts, inside and outside their organization, to help vet current common data and any new proposals for addition or amendment of common data. In particular, national standards organizations are encouraged to be involved in the data vetting process. All people involved in vetting data should compare any proposed changes against the data in the comparison charts[15] and indicated which platforms the proposed changes align with, or whether they are different than all of the platforms.

2.2 Prioritization

In anticipation that there may be conflicting common practices or standards for a given country and language, we will use keyword variants to reflect the different practices. For example, for German we will distinguish between PHONEBOOK and DICTIONARY collation.

When there is an existing national standard for a country, the goal is to follow that standard as much as possible. Where the common practice in the country deviates from the national standard, or if there are multiple conflicting common practices, or options in conforming to the national standard, or conflicting national standards, such differences in the common data repository will be distinguished by keyword variants or variant locales.

Where a data item is identified as following a particular national standard (or other reference), the goal is to keep that data aligned with that standard. There is, however, no guarantee that data will be tagged with any or all of the national standards that it follows.

3. Data Release Process

3.1 Version Numbering Scheme

The locale data is frozen per version. Once a version is released, it is never modified. Any changes, however minor, will mean a newer version of the locale data being released. The versioning scheme is x.y.z, where z is incremented for bug fixes, y is incremented for any significant additions (such as new locale data), and x is incremented for any change in format (such as the addition of new elements to the spec).

The initial version number will be 1.0.0

[14] More on Approval Voting: http://forum.icann.org/election/395ACB7A00000002.html
[15] Comparison Charts: http://oss.software.ibm.com/icu/locale/ (also see Appendix D)

3.2 Release schedule

An early release of a version of the common data will be issued as an ALPHA release. This will be followed by a BETA release, three months later. The FINAL version will be released three months after the BETA release.

Appendix B: Data Formats

This section provides basic information about data formats for fields in the Common Locale Data Repository. It is intended to clarify the meaning of some of the data for localizers and other people reading the CLDR comparison charts.

The formats are described in the Locale Data Markup Language specification[16]. Another resource is the ICU Locale Explorer[17], where the results of different formats can be viewed. Note that some fields are expected to be in English, since they indicate internal IDs and *not* translated text, such as the layout orientation, which has the value "top-to-bottom".

Collation (Sorting Sequence)

The exact collation sequence for a given language may be difficult to determine. The base ordering of characters can be fairly straightforward, but there are quite a few other complications involved. For more information, see UTS #10: Unicode Collation Algorithm[18] (UCA).

For readability, the rules are presented here in Java/ICU rule format, rather than XML; for the same reason, we prefer the bug reports to also use that format, even though the end result will be in XML. For more information, see ICU Collation Customization.[19]

Comparison Charts

In the comparison charts, rules are gathered programmatically; gathering the data for the rules is complicated by a number of factors, including:

i. Different data sources use a different "base" for tailoring (the Common rules use the UCA as a base).

ii. Sometimes special features are handled programmatically, such as having an SHADDA mark apply to all Unicode characters instead of just Arabic letters.

iii. Sometimes characters out of the repertoire are simply ignored, thus having a_ < ab, since _ (z with over-dot) is ignored.

iv. Sometimes orderings from other platforms will not abide by the UCA well-formedness criteria, so there may be cases where for characters X and Y:

- X < Y
- XY and YX are not contractions

[16] Locale Data Markup Language: http://oss.software.ibm.com/icu/locale/
[17] Locale Explorer: http://oss.software.ibm.com/cgi-bin/icu/lx
[18] UCA: http://www.unicode.org/reports/tr10/#Introduction
[19] Collation Customization: http://oss.software.ibm.com/icu/userguide/Collate_Customization.html

- but XY = YX.

At the moment there are a number of known problems with the gathering mechanism that may result in spurious rules. For more information, see the list of known errors and omissions in the Locale Repository bug database[20].

Pitfalls

There are a number of pitfalls with collation, so be careful. In some cases, such as Hungarian or Japanese, the rules can be fairly complicated (of course, reflecting that the sorting sequence for those languages is complicated).

1. **Only tailor expected data.** We focus on the required collation sequence for a given language with normal data. So we don't include full-width characters for a European collation sequence, such as
   - ... CSCS <<< ____ ...
   - ... CSCS <<< \uFF23\uFF33\uFF23\uFF33 ... (equivalently)
2. **Tailor trailing contractions.** If a sequence of characters is treated as a unit for collation, it should be entered as a contraction.

   ```
   & c < ch
   ```

   One might think that sequence like "dz" doesn't require that, since it would always come after "d" followed by any other letter; it is a "trailing contraction". But in unusual cases, that wouldn't be true; if "dz" is a unit sorted as if it were a distinct letter after "d", one should get the ordering "d_" < "dz". This will only happen if "dz" is a contraction, such as

   ```
   & d < dz
   ```

1. **Watch out for Expansions.** If you have a rule like &cs < d, and "cs" has not occurred in a previous rule as a contraction, then this is automatically considered to be the same as &c < d / s; that is, the d *expands* as if it were a "cs" (actually, primary greater than a "cs", since we wrote "<"). This expansion takes effect until the next primary difference.

   So suppose that "ccs" is to behave as if it were "cscs", and take case differences into account. You might try to do this with the rules on the left:

Rules (Wrong)	**Actual Effect**
& C < cs <<< Cs <<< CS & cscs <<< ccs <<< Cscs <<< Ccs <<< CSCS <<< CCS	& C < cs <<< Cs <<< CS & cs <<< ccs / cs <<< Cscs / cs <<< Ccs / cs <<< CSCS / cs <<< CCS / cs

[20] Known Issues: http://www.openi18n.org/locale-bugs/public?findid=18

But since the CSCS has not been made a contraction in previous rules, this produces an automatic expansion, one that continues through the entire sequence of non-primary differences, as shown on the right. This is *not* what is wanted: each item acts like it expands compared to the previous item. So CCS, for example, will act like it expands to CSCScs!

What you actually want is the following:

Rules (Right)	**Actual Effect**
& C < cs <<< Cs <<< CS & cscs <<< ccs & Cscs <<< Ccs & CSCS <<< CCS	& C < cs <<< Cs <<< CS & cs <<< ccs / cs & Cs <<< Ccs / cs & CS <<< CCS / CS

In short, when you have expansions, it is always safer and clearer to express them with separate resets. There are only a few exceptions to this, notably when CJK characters are interleaved with Hangul Syllables.

2. **Don't tailor what you don't have to.** Example: Maltese was sorting character sequences **before** a base character using the following style:

```
& B
< (dotted c)
<<< (dotted C)
< c
<<<C
```

This works, but is sub-optimal for two reasons.

1. it tailors c/C when it doesn't need to be; any extra tailoring generally makes for longer sort keys.
2. by tailoring c/C, it puts other those things that are after b/B after c/C instead. See the Unicode Collation Charts[21] for examples.

The correct rules should be:

```
& [before 1] c < (dotted c) <<< (dotted C)
```

This finds the highest primary (that's what the 1 is for) character less than c, and uses that as the reset point. For Maltese, the same technique needs to be used for dotted-c and dotted-C.

[21] Unicode Collation Charts: http://www.unicode.org/charts/collation/

Exemplar Characters

The exemplar character set contains the commonly used letters for a given modern form of a language, which can be for testing and for determining the appropriate repertoire of letters for charset conversion or collation. It is not a complete set of letters used for a language, nor should it be considered to apply to multiple languages in a particular country. In general, the test to see whether or not a letter belongs in the set is based on whether it is acceptable in that language to always use spellings that avoid that character. For example, the exemplar character set for en (English) is the set [a-z]. This set does not contain the accented letters that are sometimes seen in words like "résumé" or "naïve", because it is acceptable in common practice to spell those words without the accents. The exemplar character set for fr (French), on the other hand, must contain those characters: [a-z é è ù ç à â ê î ô û æ œ ë ï ÿ].

Punctuation and other symbols should not be included. Sequences of characters that act like a single letter in the language — especially in collation — are included within braces, such as [a-z á é í ó ú ö ü _ _ {cs} {dz} {dzs} {gy} ...]. Where combining marks are used generatively, and apply to a large number of base characters (such as in Indic scripts), the individual combining marks should be included. Where they are used with only a few base characters, the specific combinations should be included. Wherever there is not a precomposed character (e.g. single codepoint) for a given combination, that must be included within braces. For example, to include sequences from the “Where is my Character?[22]” page on the Unicode site, one would write: [{ch} {t_} {x_} {__} {__} {i__} {__}], but for French one would just write [a-z é è ù ...]. When in doubt use braces, since it does no harm to included them around single code points: e.g. [a-z {é} {è} {ù} ...].

The exemplar character set for Han characters is composed somewhat differently. It is even harder to draw a clear line for Han characters, since usage is more like a frequency curve that slowly trails off to the right in terms of decreasing frequency. So for this case, the exemplar characters simply contain a set of reasonably frequent characters for the language, based on presence in character set standards.

This set is case insensitive; this means it does not need both upper and lower case characters. The ordering of these characters are irrelevant. If the sort order or collation order is needed, then please refer to the collation section of this guide.

Display Names: Languages, Scripts, Territories, Currencies, Time Zones

These items are perhaps the simplest to translate. The typical behavior if any particular one is not available is to display the corresponding ISO code (or RFC 3166 code). Thus if there is no localization of FR (France), then the string "FR" will appear.

[22] Where is my Character? - http://www.unicode.org/standard/where/

For a list of the kinds of IDs that can be localized, see the English-US locale data[23], which has all territories, languages, scripts and currency names that can be translated.

Because these lists are long, it may be more difficult to collect all the data at once. If that is the case, we suggest first providing translations of at least the elements that are most familiar to people with the target language. This might be, for example, the countries, languages, scripts, and (modern) currencies for:

1. Countries having the target language as an official language
2. G7 + BRIC countries: US, Canada, Japan, UK, France, Germany, Italy; Brazil, Russia, India, China
3. Countries adjacent to the countries in #1

Languages

The language codes are based on ISO-639. [24] The translations for the names of the other languages in the current given language should be cased according to how the language name would appear in the middle of a sentence. If the user of this data is going to use the name at the beginning of a sentence, the developer (not the translator) will title case the word. For example, English will translate fr to be "French", but Spanish will translate fr to be "francés".

Scripts

The script codes are based on ISO-15924.[25] A script is the set of characters used to display a language. For example, English is written with the Latin script, and the ISO-15924 code for Latin is "Latn".

Some language can be written in more than one script. For example, the Serbian language can be written in the Latin or Cyrillic scripts, and Uzbek can be written with Cyrillic, Latin or Arabic scripts.

Territories

The region codes are based on ISO-3166.[26] These region codes include countries, territories and some other places of interest around the world.

Currencies

The currency codes are based on ISO-4217.[27] There are two things can be translated here. One part is the display currency symbol (e.g. the $ for the US dollar), and the other part is

[23] English (US): http://oss.software.ibm.com/cvs/icu/~checkout~/locale/diff/main/en_US.html
[24] ISO-639: http://lcweb.loc.gov/standards/iso639-2/
[25] ISO-15924: http://www.evertype.com/standards/iso15924/
[26] ISO 3166: http://www.iso.org/iso/en/prods-services/iso3166ma/
[27] http://www.bsi-global.com/Technical+Information/Publications/_Publications/tig90.xalter *and* http://www.bsi-global.com/Portfolio+of+Products+and+Services/Books+Guides/Consumer/th42090.xalter

the display name (e.g. the "US Dollar" for $). The ISO-4217 code is used as the key for these translations.

Time Zones

A time zone uses the Olson IDs, such as America/Los Angeles. These IDs provide not only the information as to the offset from GMT (UTC), but also the daylight savings information. Thus, for example, America/Phoenix is distinct from America/Denver because they have different daylight savings time behavior. There are many cases where there is no modern difference between two IDs; they are only distinguished in the past. For example, America/Chicago and America/Indianapolis are currently the same, but differed in the past. For the possible timezone IDs that can be translated, see Olson IDs[28].

Most countries do not span multiple time zones, and may not have localized names for particular time zones. In that case, probably the best localizations are something of a form like "New York Time".

The fields are the long and short (abbreviated) versions of:

- **generic** time zone name, e.g. “Pacific Time”
- **summer/daylight savings** time, e.g. “Pacific Standard Time”
- **standard** time, e.g. “Pacific Daylight Time”

Numbers

Number Patterns

There are different formats for different types of numbers. For example, here are some patterns (using the unlocalized symbols) with the corresponding text. While the format is localized, the *characters to specify the format pattern* are not localized. That is, the decimal point always is represented by the period (full stop) character, as in “3.14159”, and the grouping separator (thousands, etc.) is always represented by a comma, as in “1,000,000”.

The NumberElements resource affects how these patterns are interpreted in a localized context. Here are some examples, based on the French locale. The "." shows where the decimal point should go. The "," shows where the thousands separator should go. A "0" indicates zero-padding: if the number is too short, a zero (in the locale's numeric set) will go there. A "#" indicates no padding: if the number is too short, nothing goes there. A "€" shows where the currency sign will go. The following illustrates the effects of different patterns for the French locale, with the number "1234.567"

Pattern	Text
#,##0.##	1 234,57
#,##0.###	1 234,567

[28] Olson Data: http://oss.software.ibm.com/cvs/icu/~checkout~/locale/olson_ids.txt

####.#####	1234,567
####.0000#	1234,5670
00000.0000	01234,5670
# ##0,00	1 234,57

decimalFormats

The normal locale specific way to write a base 10 number.

currencyFormats

Use \u00A4 where the local currency symbol should be. Doubling the currency symbol (\u00A4\u00A4) will output the international currency symbol (a 3-letter code).

percentFormats

Pattern for use with percentage formatting

scientificFormats

Pattern for use with scientific (exponent) formatting.

Quoting rules

Single quotes, ('), enclose bits of the pattern that should be treated literally. Inside a quoted string, two single quotes ('') are replaced with a single one ('). For example: `'X '#' Q '` -> **X 1939 Q** (Literal strings underlined.)

Number Elements

Localized symbols used in number formatting and parsing.

decimal

- separates the integer and fractional part of the number.

group

- groups (for example) units of thousands: 10^6 = 1,000,000. The grouping separator is commonly used for thousands, but in some countries for ten-thousands. The interval is a constant number of digits between the grouping characters, such as 100,000,000 or 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. So "#,##,###,####" == "######,####" == "##,####,####".

list

- separates lists of numbers

percentSign

- symbol used to indicate a percentage (1/100th) amount. (If present, the value is also multiplied by 100 before formatting. That way 1.23 => 123%)

nativeZeroDigit

- Symbol used to indicate a digit in the pattern, or zero if that place would

otherwise be empty. For example, with the digit of '0', the pattern "000" would format "34" as "034", but the pattern "0" would format "34" as just "34". As well, the digits 1-9 are expected to follow the code point of this specified 0 value.

patternDigit

- Symbol used to indicate any digit value, typically #. When that digit is zero, then it is not shown.

minusSign

- Symbol used to denote negative value.

plusSign

- Symbol used to denote negative value.

exponential

- Symbol separating the mantissa and exponent values.

perMille

- symbol used to indicate a per-mille (1/1000th) amount. (If present, the value is also multiplied by 1000 before formatting. That way 1.23 => 1230 [1/000])

infinity

- The infinity sign. Corresponds to the IEEE infinity bit pattern.

nan - Not a number

- The NaN sign. Corresponds to the IEEE NaN bit pattern.

currencySeparator

This is used as the decimal separator in currency formatting/parsing, instead of the DecimalSeparator from the NumberElements list. This item is optional in the CLDR.

currencyGroup

This is used as the grouping separator in currency formatting/parsing, instead of the DecimalSeparator from the NumberElements list. This item is optional in the CLDR.

Dates

Day & Month Names

All of the following should have the appropriate grammatical form for appearing in a date format.

monthNames (Long Names)

These are the full month names, like "September".

monthAbbr (Short Names)

These are the abbreviated month names, like "Sep".

dayNames (Long Names)

These are the full Gregorian day names, like "Monday"

dayAbbr (Short Names)

These are the abbreviated day names, like "Mon"

Date & Time Options

firstDay

A number indicating which day of the week is considered the 'first' day, for calendar purposes. Because the ordering of days may vary between calendar, keywords are used for this value, such as sun, mon,... These values will be replaced by the localized name when they are actually used.

minDays (Minimal Days in First Week)

Minimal days required in the first week of a month or year. For example, if the first week is defined as one that contains at least one day, this value will be 1. If it must contain a full seven days before it counts as the first week, then the value would be 7.

weekendStart, weekendEnd

Indicates the day and time that the weekend starts or ends. As with firstDay, keywords are used instead of numbers.

Date & Time Patterns

The following are characters used in patterns to show the appropriate formats for a given locale. These characters are replaced with the appropriate values when a date or time is being formatted.

Characters may be used multiple times. For example, if y is used for the year, 'yy' might produce '99', whereas 'yyyy' produces '1999'. For most numerical characters, the number of characters specifies the field width. For example, if h is the hour, 'h' might produce '5', but 'hh' produces '05'. For some characters, the count specifies whether an abbreviated or full form should be used.

Sym.	Description
`G`	Era - Replaced with the Era string for the current date.
`y`	Year - Use two for the short year, or 4 for the full year
`M`	Month - Use one or two for the numerical month, three for the abbreviation, or four for the full name.
`d`	Date - Day of the month. Use one or two for zero padding.
`h`	Hour [1-12]. Use one or two for zero padding.
`H`	Hour [0-23]. Use one or two for zero padding.
`K`	Hour [0-11]. Use one or two for zero padding.
`k`	Hour [1-24]. Use one or two for zero padding.
`m`	Minute. Use one or two for zero padding.
`s`	Second. Use one or two for zero padding.
`S`	Millisecond - Use 1,2, or 3. shows the most significant digits.
`a`	AM or PM
`E`	Day of week - Use three for the short day, or four for the full name.
`D`	Day of year - Use 1-3

F	Day of Week in Month- use one.
w	Week of Year. Use one or two for zero padding.
W	Week of Month - use 1
z	Timezone. Use 3 for the short timezone (i.e. PST) or 4 for the full name (Pacific Standard Time). If there's no name for the zone, it'll show up as GMT+/-hh:mm.
A	Year (of "Week of Year"). [May differ from Calendar year, see comments under 'Week of Year', above.]

For more information, see the Java SimpleDateFormat documentation.[29]

localizedPatternChars

These are characters that can be used when displaying a date pattern to an end user. This can occur, for example, when a spreadsheet allows users to specify date patterns. Whatever is in the string is substituted one-for-one with the characters "GyMdkHmsSEDFwWahKzYe", with the above meanings. Thus, for example, if "J" is to be used instead of "Y" to mean Year, then the string would be: "GyMdkHmsSEDFwWahKzJe".

Quoting rules

Single quotes, ('), enclose bits of the pattern that should be treated literally. Inside a quoted string, two single quotes ('') are replaced with a single one ('). For example: `'class of 'YYYY' at 'h' o''clock'` -> class of 1939 at 6 o'clock (Literal strings underlined.)

AM / PM

Even for countries where the customary date format only has a 24 hour format, both the am and pm localized strings must be present and must be distinct from one another. Note that as long as the 24 hour format is used, these strings will normally never be used, but for testing and unusual circumstances they must be present.

Eras

There are only two values for an era in a Gregorian calendar, "BC" and "AD". These values can be translated into other languages, like "a.C." and and "d.C." for Spanish, but there are no other eras in the Gregorian calendar. Other calendars have a different numbers of eras. Care should be taken when translating the era names for a specific calendar.

[29] SimpleDateFormat: http://java.sun.com/j2se/1.4.1/docs/api/java/text/SimpleDateFormat.html

Week of Year

Values calculated for the Week of Year field range from 1 to 53. Week 1 for a year is the first week of that year that contains at least the minimum days in a week days. Weeks between week 1 of one year and week 1 of the following year are numbered sequentially from 2 to 52 or 53 (if needed). For example, January 1, 1998 was a Thursday. If the first day of the week is MONDAY and the minimum days in a week is 4 (these are the values reflecting ISO 8601 and many national standards), then week 1 of 1998 starts on December 29, 1997, and ends on January 4, 1998. However, if the first day of the week is SUNDAY, then week 1 of 1998 starts on January 4, 1998, and ends on January 10, 1998. The first three days of 1998 are then part of week 53 of 1997.

Values are similarly calculated for the Week of Month.

BIDI Ordering

In the comparison charts, the ordering of characters for BIDI languages is up to the browser. The ordering may differ from the typical ordering in context, so may be helpful to copy the text from the charts and paste into typical environments to ensure that the ordering is correct. This is especially the case for date, time and number formats.

***Note*:** If you ever need to convert text from hex format (\uXXXX) to real characters or back, you can do the following:

1. go to the ICU Transform Demo - http://oss.software.ibm.com/icu/demo/
2. set *Compound 1* to NFC, and *Compound 2* to [:^ascii:] hex
3. paste what you want to convert into *Input*, and click *Transform*.
4. this will also give the NFC form of the text, which is preferred.

Appendix C: Filing Locale Data Bug Reports

Locale data bugs and feature requests are filed at the Common Locale Repository website.

However, before filing a bug report:

1. Review the Data Formats (Appendix B), so that you are familiar with the format of the data in question.
   - Even for items as simple as the months of the year, there are issues of grammatical form that you should be acquainted with.
2. Consult the appropriate comparison chart (Appendix D).
   - The purpose is to compare the data against that from other platforms.
3. *Do not file bugs against platform data.*
   - That data simply represents what was gathered on the platform, and is not under the control of this project.
   - However, you can file bugs on the *tools,* if it appears that the platform data is not being correctly gathered in XML or displayed as HTML in the comparison charts.
4. If you have any questions, you can try the newgroups at news://www.openi18n.org

In the bug report, include the following information:

- The specific locale.
- The date the comparison chart was generated (at the bottom of the chart).
- The line(s) of data that are incorrect.
- The justification for the change: mention a standard or other sources if there are any.
- If one of the other data sources is correct, then that data source, otherwise the corrected data to use.
- Wherever possible, include sample test data. This is especially important for collation (sorting) rules!

Please group all bugs for a single locale into a single bug report, wherever possible.

Bugs may also be filed on the supplemental data[30], which is only available in XML format. Note that the default currency for a given country is in the supplemental data, not in the locale data. Similarly, the timezones for different regions are to be derived from Olson data[31], and are not in the locale data. The *localized names* for currencies and timezones, on the other hand, will be in the locale data (if available).

We have tried to make the format of the comparison charts as useful as possible. If you have suggestions for improvements, please also file it as a bug report.

---

[30] Supplemental Data: http://oss.software.ibm.com/cvs/icu/locale/common/xml/supplementalData.xml
[31] Olson IDs: http://oss.software.ibm.com/cvs/icu/~checkout~/locale/olson_ids.txt

Collation Bugs

Please supply some short test cases that illustrate the correct sorting behavior as a list of lines in sorted order. Try to include cases that show the boundary behavior by including high suffixes, such as the following:

- *Rules:*

  & *c* < *cs*
  `& cs <<< ccs / cs`

- *Test Data:*

```
c
cy
cs
cscs
ccs
cscsy
ccsy
csy
d
```

Test out any suggested rules before filing a bug, using Locale Explorer:

1. Go to
   http://oss.software.ibm.com/cgi-bin/icu/lx/en/utf-8/?_=root&EXPLORE_CollationElements=
2. Pick the appropriate locale
3. Follow the instructions at the bottom to use your suggested rules on your suggested test data.
4. Verify that the proper order results.

Appendix D: CLDR Comparison Charts

The Common Locale Data Repository comparison charts provide comparisons between locale data from different sources. The files are organized by locale. In each file, the first three columns identify the data item, while the subsequent columns contain data. There may be different numbers of columns per locale, based on the available comparison data. The Common data is in the first data column, with other data sources following (where available). The latter sources are generated with public APIs.

Field Formats

The format of many of the fields in the comparison chart will be clear from the Name and ID, such as the months of the year. The format for others, such as the date or time formats, is structured and requires more interpretation. If you have any questions about the format, consult the overview in the Data Formats section, especially the notes on Collation (Appendix B).

Bug Reports

For information on filing bug reports on the data, see Appendix C, Filing Bug Reports.

Columns

The links within the top header cells point to the XML data files, either for this locale or for a related locale (the parent or root). Each column is marked with a color at the top. The successive rows provide a comparison of the data, with the following color coding:

- Missing data is indicated by a white cell.
- Data that is the same as some column to the left uses the same color (with an equals sign in the cell).
- Data that is identical except for a case change is indicated with a dagger (†).

Data that is different than any column to the left has the column's normal color.

Examples from different locales:

N.	ParentNode	Name	ID	COMMON (nl_NL, nl, root)	WINDOWS (nl_NL)	SUNJDK (nl_NL, nl, root)	IBMJDK (nl_NL, nl, root)	SOLARIS (nl_NL)	OPEN_OFFICE (nl_NL)	AIX (nl_NL)	LINUX (nl_NL)	HP (nl_NL
...												
56	dayNames	day	sat	zaterdag	=	=	=	=	=	=	=	=
...												
5	dateFormat	pattern	full	yyyy 'm.' MMMM d 'd.',EEEE	yyyy 'm.' MMMM d 'd.'	EEEE, yyyy, MMMM d	=	yyyy MMMM dd HH:mm:ss		yyyy m. MMMM dd d.		=
...												
120	territories	territory	ZA	Suid-Afrika	Suid Afrika				South Africa	...		
...												
79	languages	language	be	_____	_____†		_____†	...				
...												
35	currency	displayName	EUR	___	...							
...												
2	characters	exemplarC		[a-z _ _ _ _ _ _ _ _ _]	...							
...												
723	types	type	phonebook	Telefonbuch-Sortierregeln	...							

Collation

The collation pages are separated off for easier viewing. There are only currently three comparison columns for collation.

COMMON (xml UCA)	LINUX (xml Base (en_US))	WINDOWS (xml base (en))

The XML link points to the data file, while the base link points to the collation base. The base for the Common collation rules is the UCA. For the other data sources, the base is chosen to be an ordering for one of the locales sharing the same script. That permits the rules to only contain differences.

Appendix E: Locale Data Markup Language Version 1.0 Errata

Since version 1.0 of the Locale Data Markup Language was released June 24th, 2003, a number of corrections have been made. They are reflected in the examples and text above, and are briefly summarized here:

- "`currency=pre-euro`" is no longer used as a currency type.
- In the `<special … >` keyword example, `%icu` should be `%posix`
- U+0000 should be used for escaping, instead of \u0000
- The document's title should be Locale Data Markup Language, and not Locale Data Interchange Format.
- The collation format changed slightly, and has a new element, `<base>`, which contains an alias element that points to another data source that defines a *base* collation. This allows collation rules to be more compact and maintainable,

For more details, please see the full errata page, via http://oss.software.ibm.com/icu/locale/