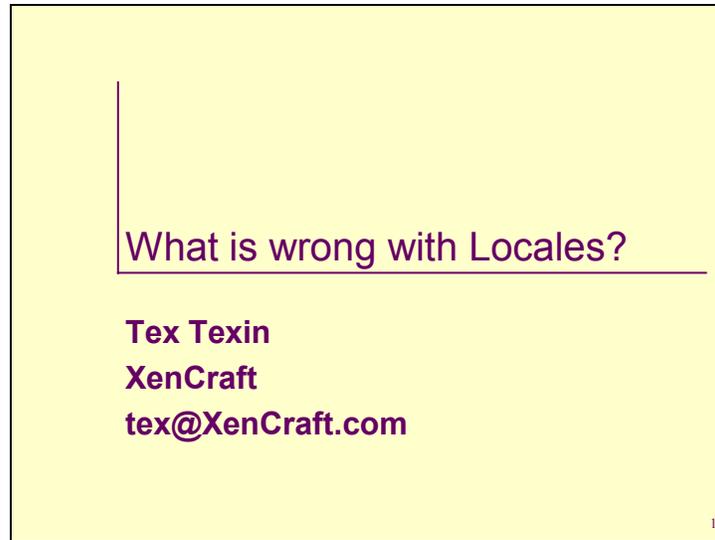


What is wrong with Locales?



Objectives

- Describe locales and their intended purpose
- Introduce their design and operational problems
- Seed ideas for the panel discussion

2

"Locale" is a mechanism used in the Web, Java, and many other technologies to establish user interface language, presentation formats, and application behavior. Many application developers make use of locales and presume their applications now have correct international behavior. This session will describe the locale mechanism, and introduce some of the design and operational problems with locales.

It will suggest some potential solutions. After this presentation, there will be a panel discussion with audience participation to evaluate the problems further as well as consider ideas for solutions.

What is wrong with Locales?

Agenda for: What is wrong with Locales?

- **Intended usage**
- **Definition(s)**
- **Software requirements for locales**
- **Problems**
- **Summary**

3

Locales
Intended Usage

- **Shorthand description of user's cultural preferences**
- **Easy way to set default i18n/l10n behaviors**
- **Communicate user's software capabilities**
- **Vehicle for processes to agree on data formatting and interpretation for interoperability**

4

Locales are used in several different ways.

Locales are a way to succinctly represent the cultural preferences (such as date and number formatting, page size, etc.) of a user, based on stereotypes. An American user will likely want MDY date format for example.

This information can then be used to configure software for the user and save the user from having to address many details. The user only needs to change the settings that he or she disagrees with, rather than setting values for every item.

Some applications may use the locale information about the client's locale to estimate the capabilities of the client platform. For example, to guess which encodings, and fonts may be supported by the client. Java for example may use the locale information to determine which encoding to produce files with.

Integrated applications may use locale to determine how shared data is to be parsed or interpreted. For example, a database application may use the locale information so that the server sorts the data according to the collation needed by the client. A number sent as a character string to the server will be parsed by the server using the thousand separator and fractional separator determined by the client's locale.

What is wrong with Locales?

Locales

User Cultural Preferences

- **User Interface language**
- **Text format (quote, hyphenation, justification, case, direction)**
- **Data format (date, time, number, currency)**
- **Measurement Units**
- **Input method**
- **Keyboard layout**
- **Encodings**
- **Fonts**
- **Collation**

5

Here are a few of the preferences that may be determined by locale.

Often the language of the user interface is set by locale.

Rules for rendering text can also be controlled by locale. For example, locale may determine the quote characters that are used (e.g. double quote vs. guillemet)

Any of the internationalization or localization settings of software may be influenced by locale.

What is wrong with Locales?

Locales

User Cultural Preferences

- **Tex is en-US, therefore he prefers:**
 - American English UI and keyboard
 - M-D-Y hh:mm am/pm (GMT-0800 to -0400)
 - USD \$9,999,999.99
 - A<B<C...<Z
 - Fonts, encoding 1252 / ISO 8859-1
 - Titles: Mr./Ms./Mrs., Address: Big to small
 - Baseball and hot dogs

6

To make the example more concrete, user Tex has selected a locale of “en-US”, English as spoken in the United States.

Therefore, he is likely to use the settings shown here, such as an American English user interface and an American keyboard layout.

Month-day-year calendar format, etc.

The software might even consider that he likes Baseball and eats hot dogs...

Locales
Who uses Locales?

- **HTML** - to aid in rendering
- **XML** - language identification
- **UNIX** - specific categories of information per locale, globally
- **Java** -each locale-sensitive class maintains its own locale-specific information
- **Microsoft**- uses a different mechanism also called locale.

7

HTML specifies that the lang attribute can be used to guide rendering.

<http://www.w3.org/TR/html401/struct/dirlang.html>

The language attribute uses the same format as Locale, described by RFC 1766.

(<http://www.ietf.org/rfc/rfc1766.txt>)

Although the lang attribute can be specified in HTML FORMS, there is no indication from the standard how the values should be used. In particular it doesn't say if the attribute can be used for other than rendering, for example parsing data formats that are in the form.

XML 1.0 also supports a language attribute xml:lang referencing RFC 1766. The specification says it is to be used for language identification, with no other details provided. (<http://www.w3.org/TR/REC-xml#sec-lang-tag>)

UNIX defines specific categories of cultural information that are associated with locale IDs. The locale is global to the program.

http://www.opengroup.org/onlinepubs/007908799/xbd/locale.html#tag_005_001

Java extends the usage of locale to a multithreaded, multi-locale, object-oriented environment. <http://java.sun.com/products/jdk/1.1/intl/html/intlspec.doc2.html>

What is wrong with Locales?

Locales UNIX categories

- **LC_CTYPE-** character classification and case conversion
- **LC_COLLATE-** collation order
- **LC_TIME-** date, time formats
- **LC_NUMERIC-** number formatting
- **LC_MONETARY-** monetary formatting
- **LC_MESSAGES-** formats of messages and interactive responses

8

http://www.opengroup.org/onlinepubs/007908799/xbd/tag_005_001

This slide shows the kind of data associated with Unix locales.

Locales
Naming Convention

- **Series of tags (RFC 1766)**
 - Locale = primary-code ("-" subcode)*
- **Primary 2-letter tag is Language**
- **“i” is IANA registered, “x” is private**
- **(Optional) 2-letter subtag is Country**

“fr”, “fr-CA”, “fr-FR-euro”
“en-cockney”, “i-navajo”, “x-klinton”
“hr-latin”, Đorđe Balašević
“hr-cyrillic” Ђорђе Балашевић

9

Here is the naming convention used for locales. They are based on RFC 1766.

(<http://www.ietf.org/rfc/rfc1766.txt> “Tags for the Identification of Languages”)

Although each markup language or programming language may define a capitalization convention, locale names are case insensitive.

2 letter primary tag is an ISO 639-1 language code.

If subtag is a 2 letter code it is an ISO 3166-1 country code.

Note that values other than 2 letters are allowed, but then they are not 3166 country codes.

The primary tag value "i" is reserved for IANA-defined registrations.

The value "x" is reserved for private use. Subtags of "x" will not be registered by the IANA.

IANA does not register locale names that are in the 639-3166 format.

The examples of hr-latin, hr-cyrillic show how a language (Croatian) that is written with either of 2 scripts might be identified.

The name next to each of the Croatian language tags is the same but is written in each script.

What is wrong with Locales?

All set!

- **Has a naming convention**
- **Based in existing standards,**
 - ISO 639-1, ISO 3166, RFC 1766
- **Has a locale name registry (IANA)**
- **Employed by significant standards and technologies**
- **We are golden!**

10

Since locales are already in use, and are being extended into new standards such as XML and Java, they must be useful and robust methods for identifying user preferences and cultural conventions, and communicating them between software processes.

Or are we?

- **Are locales a good indicator of user's cultural preferences?**
- **Are these ISO standards a good basis for software to build on?**
- **Are the semantics well-defined? Where?**
- **Do all the technologies define locales the same way, to allow integration?**

11

It sounds good, but perhaps we should take a closer look at locales to understand:

Does the naming convention correlate well with each user's cultural preferences?

The ISO standards were not designed for these particular purposes. How well do the goals of ISO with respect to these standards, align with the requirements of software applications?

As a developer, when given a locale do I know what to do with it? Which behaviors should I provide for any given locale? How do I find out what the expectations are? And how do I know that a software package that I interoperate with, will use the locale in the same way?

Locale

Software requirements

- **Naming Conventions**
 - **Global coverage**
 - **Granularity**
 - **Stability**
 - **Correlation of locale name to culture**
 - **Registered, clearly defined, determinate**

12

Here are what I believe to be the requirements that Locales must satisfy to be useful to software applications.

To be applicable globally, I must be able to represent cultural preferences anywhere in the world.

At the same time, the locale identifier should be able to distinguish adequately between groups of users that have significantly different preferences.

Because software is often used for a long time and yet may need to integrate with newer and older generations of other applications, the locale definitions must be stable. In particular, the identifier used for a particular locale should not have different meanings to different versions, and once defined they should not disappear or be reassigned.

Of course, a locale name should adequately correlate with a culture so that it identifies a group of users with the same preferences.

And for any locale it should be clear which settings a user choosing that name would prefer, or there needs to be a way for developers to identify the appropriate settings (such as a registry). It should also be clear to the user that a particular locale represents his needs and is the appropriate choice (Although the software user interface can help with this, if the choice is not clear from the name alone.)

Locale

Software requirements

- **Selection, Match rules**
- **Semantics**
 - **Defined, Scoped**
 - **Platform, technology-independent**

13

Just as users identify their preferences by choosing a locale name, software needs to select language files, provide a variety of behaviors, etc. based on the user's locale. Since it is generally not possible for software to enumerate and provide all possible locale variations, some type of matching rules are applied to find the best fit of software capabilities to user preferences. So a user requesting Spanish as spoken in Columbia from software that does not have a Columbian Spanish language file, might be able to offer other variations of Spanish or other languages that would meet the user's needs. Behaviors and expectations should be set so that selection is predictable.

The set of behaviors associated with locales should be well-defined.

Perhaps their scope should be established. Currently, for HTML their scope is limited to rendering. Other technologies allow locale to influence interpretation and parsing of user data or choices such as currency. For reasons of integration, it should be made clear what locales influence and do not influence.

Locales should also work (have the same impact) across platforms and across technologies.

Locale Naming Conventions
Language defined by ISO 639-1

- **“primarily for use in terminology, lexicography and linguistics.”**
- **~160 2 letter codes**
- **Stability**
 - ji->yi, in->id, iw->he,
 - sh withdrawn, sr hr
- **Redundancy**
 - No Norwegian, nn Nynorsk, nb Bokmal
- **Scripts, Orthography not referenced.**

14

Let's take a closer look at the constituents of the locale name.

The language portion is based on ISO 639-1 2 letter codes. There is also a 639-2 which has 3 letter codes that are not used for locale names.

(<http://www.loc.gov/standards/iso639-2/englangn.html> lists 639-1 codes.)

This standards says that it is “primarily for use in terminology, lexicography and linguistics.” The standard contains about 160 language codes.

The standard does revise codes from time to time. For example, Yiddish “ji” changed to “yi”. Indonesian “in” became “id”. Hebrew “iw” became “he”. Serbo-croatian “sh” was replaced by Serbian “sr”, and Croatian “hr”.

I find it interesting that text which was adequately described as Serbo-croatian must now be one or the other, but Norwegian, can be labeled Norwegian, or the more specific Nynorsk or Bokmal. I am not a language expert. However, there seems to be a political aspect to the changes rather than a pure linguistic basis.

Software needs stable codes which do not change when governments change, or existing, integrated software becomes obsolete or incompatible.

Note that 639 is terminology based, and doesn't take into account writing style or orthography, whereas scripts are important to software and especially fonts.

So we might need to consider whether ISO 639-1 is the right standard to base locales on for the long term.

Locale Naming Conventions
Country defined by ISO 3166

- **"names of countries, dependencies and other areas of particular geopolitical interest"**
- **239 entries**
- **Awareness of software requirements**
 - use in locale, and domain names
- **Stability**
 - Deleted 26 codes in last 26 years
 - Recent changes: tp->tl, rom-> rou

15

ISO 3166 says that it represents the “names of countries, dependencies and other areas of particular geopolitical interest.”

(<http://www.iso.org/iso/en/prods-services/iso3166ma/index.html>)

The 3166 web page makes it clear that the authors of the standard are aware of its use in software and locales, and in domain names as well. So we have a good sign perhaps that this standard is suitable for software usage.

Only 26 codes have been removed in the past 26 years, although recently a couple codes have changed.

Timor East (In Portuguese "Timor Leste") changed from tp->tl as part of the change from a Portuguese territory to an independent nation.

The change in status justifies a new country code, and depending on the contents of locale, perhaps the locale should change as well.

The definition of locale should make clear if there are country or governmental dependencies. (Such as address, zip, phone, license codes).

Locale Naming Conventions
Language-Country

- **Global coverage**
 - 7000 languages spoken vs. 160 codes
 - 1.5Bn speak Mandarin, Spanish, English
- **Registered, or clearly defined**
 - Any combination of lang-country allowed
 - Vendors unclear on which to support
- **Granularity**
 - Insufficient for time zone, dialects
 - Some cultures span borders

16

How well does Locale meet the remaining requirements? There are 160 language codes, but 7000 languages spoken today. However, the top 3 languages are together spoken by 1.5 billion people.

Language #50 (in decreasing # of speakers), is spoken by 20 Million people. So by the time you get to language #161, perhaps there are not too many speakers represented. On the other hand, if you want to support a truly “World Wide” Web, or meet the needs of minority groups, you need more than 160 language codes. If you want to find out more about the 7000 languages or their relation to the 160 codes, visit the Ethnologue web site: <http://www.ethnologue.com/web.asp>

However, the 160 language codes multiplied by the 239 country codes makes for 38,240 possible locale names. Not all of them make practical sense. So software vendors looking to be economical (or profitable) would like guidance as to which should be supported. At the moment there is no such guidance, and different vendors support different sets of locales.

Countries come in different sizes as well. A geographically small country, with a monolithic (monocultural?) population, might be well represented by a language-country naming convention. But other countries may have multiple cultures (e.g. Belgium) or may span large distances which include multiple time zones or other information that may be part of a locale. And some cultures span borders.

Locale Naming Conventions
Language-Country

- **Stability**
 - ISO codes change
 - Countries and Government's change
 - 25 new nations since 1990
 - Currencies change e.g. euro
 - Language rules change (e.g. Spain-sorting, Germany- capitalization, many spelling)
 - Time zones and other attributes change

17

We have seen that ISO codes may change, and obviously countries and governments change, with 25 new nations being added in the past 12 years. Other elements of a locale may change as well. Recently many of the currencies of Europe changed to the euro.

From time to time, the rules of language change as governments try to improve literacy, or adapt language to new technologies or simply to legitimize current conventions.

Locales need to have a mechanism that adapts a changing world to a fixed set of locale names or identifiers.

Other standards have similar problems, code pages for example, and address the problem with new code pages or including a year or version in the name of the standard. E.g. ISO 8859-1:1987.

Locale Naming Conventions

Language-Country

- **Correlation of locale to culture**
 - does not determine orthography
 - e.g. zh-CN does not stipulate Simplified Chinese
 - Most software has orthographic needs
 - May be one-to-many relationship
 - E.g. Japanese emperor dates and Y-M-D
- **Usefulness in multilingual environments**
 - E.g. sorting

18

To be useful and meaningful, there should be a strong correlation between locale name and a particular culture. However, language and country does not correlate with some aspects of locale or at least cultural preferences that software needs to know about.

The most significant example is that language-country does not indicate orthography or script. Zh-CN refers to the language spoken in China, not necessarily Simplified Chinese vs. Traditional Chinese writing style. Locale for Yugoslavia does not provide guidance as to whether the language should be written in Latin or Cyrillic scripts. (Both are used there.)

Another example is Japan uses two date formats. Sometimes Japanese Emperor data format is used (for Government applications) and sometimes Year-month-day is used with Japanese characters as separators between each part of the date format.

Some might argue that the use of one or the other format is up to the application, just as long or short date formats might need to be an application choice made in other markets. Perhaps, others would like locale to be more specific, perhaps a government variant to distinguish formats required to meet government demands, separate from more informal usage.

Although a locale might imply the appropriate collation rules to use for the user's primary language, in a multilingual environment, more information may be needed to establish the relationship of secondary or other languages in the sorting order.

Locale Naming Conventions
Language-Country-*Variant*

- **Locales allow user-defined variants.**
- **Vendors also define variants.**
 - E.g. fr-FR-euro
- **This can cause conflicts.**
- **Variants undermine interoperability.**
- **Variants should either be used only for private use or registered with IANA for global use.**

19

It is possible to make locale names more specific by adding variant subtags.

During the transition to the euro currency different vendors took different approaches to specifying the old or new currency.

Some changed currency in the existing locales on Jan 1, 2002. Others defined variant tags such as fr-FR-euro to be used with the euro currency or fr-FR-peeuro to represent the old currency.

The lack of standardization around euro locales made for some problems in integrating software.

In addition, since both vendors and users can define variants, the potential for conflicting locale names exists.

Variants are useful for specialized situations, analogous to the Unicode Private Use Area.

But if variants are frequently resorted to, it is an indication of problems in the general locale naming mechanism.

Locale Semantics

- **No universal definition for semantics.**
 - Integration is therefore difficult or risky
 - Unix, Java have some defined locales.
 - Individual locales have different behavior
 - Does fr-FR imply Francs or euro?
- **Scope of locale varies**
 - HTML, XML refer to (only) rendering
 - Java, Unix are more inclusive
 - E.g. Currency

20

To be fair Unix, has defined categories for locale. However, since other technologies use locale differently, and there is no general definition for locale, integrating technologies can be risky. And as has been pointed out, there is no definition for the specific values to be associated with individual locales, so even where there might be general agreement, integration might still be problematic.

There should be agreement on what locale does and does not influence, so that all technologies can be influenced to the same extent, and where choices need to be made in arenas that locale does not influence other more suitable mechanisms can be defined. For example, if locale includes Currency, then this could be added to web standards. On the other hand if it does not, web applications will know to establish another method to establish currency and adapt its use in mixed technology environments.

Locale
Selection, Match rules

- **User identification**
 - **With no definition of behaviors for locale, user selects language-country:**
 - **By birth? upbringing? current location?**
 - Phone format, time zone, can change with location, perhaps address format.
 - **Does using a machine in another country affect user's locale?**
 - Encoding, fonts, keyboard, capabilities might change

21

When software prompts a user for language and country in order to establish the user's cultural preferences, the user does not know the significance of the question. In particular the user may not be aware that this is to be a basis for cultural preferences and which characteristics are associated with each locale. They do know the preferences they want, but may not be clear on the best locale to choose to establish those preferences.

If a user travels, they may not want their locale to change, in order to maintain their current language settings for example. On the other hand, if they are to make use of phone numbers and other information in their new location, they may want time zone, phone format, address format and other information to reflect the current location. It is not clear what users should do in this situation. It might also be the case that what users really need is some hybrid locale where some values are changed based on location and others are maintained based on the user's background.

A further complication occurs when a user travels and uses a machine in another country. As the machine may have only local fonts, keyboard, and encodings installed, to be usable, those elements of the locale must reflect these requirements, even if the user is not a native language speaker.

**Locale
Selection, Match rules**

- **User identification**
 - **No guidance for conflicts**
 - e.g. should jp-US return YMD or MDY
 - **A relocated user in another country can't be specific about his original locale**
 - e.g. (en-US)-CA American in Canada

22

There is no guidance or recommendation as to how software should behave when there are conflicts in locale settings.

For example, should a Japanese speaker in the U.S. choosing jp-US as locale, see the Japanese YMD date format or the American MDY date format?

For some users, it may be frustrating that they cannot be more specific about their language when in a different country.

For example, Canadian English has some words and spellings in common with Britain, and therefore is different from English as spoken in the U.S. But an American in Canada does not have a way of asserting American English in Canada, they can only choose en-CA, English as spoken in Canada.

Locale
Selection, Match rules

- **Rules for matching are not universal**
- **RFC 1766, XML doesn't provide guidance**
- **HTML 4 follows locale hierarchy**
 - fr-FR-euro, fr-FR, fr
- **Java follows locale hierarchy, then default locale hierarchy**
- **Unix seeks locale, then defaults to “C” locale**

23

Given a locale, if there isn't an exact match, how should software choose a suitable alternative?

The XML and RFC 1766 standards do not provide any recommendations for this.

HTML follows a hierarchy by dropping the rightmost tags until a match is found.

So for example, if the locale is fr-FR-euro, the software should look for a match with fr-FR-euro, then fr-FR, then fr.

Java follows a similar hierarchy, but if there is no match with the primary language tag (e.g. fr) then it repeats the process of matching with the hierarchy with the “default” locale.

Unix looks for an exact match on locale and if not found it uses the “C” locale.

Although theoretically, differences in matching strategy could result in different choices of locale, in practice different technologies would not share locale resources so it is not clear how strongly this is responsible for creating differences in locale behavior.

At some point, if locale definitions could be shared across technologies, this would be more significant.

Possible solutions for Locales

- **RFC 3066 uses (~380) 3 letter language codes**
- **Language-country is not sufficient for everything, use it just for language.**
- **Use other names for different categories of capabilities**
- **Use date(s) to identify generation**
- **Define semantics, perhaps XML file**
- **Provide for capability negotiation**

24

Here are some ideas for consideration

Locales could adopt RFC 3066, which upgrades RFC 1766 to use ISO 639-2 3 letter language codes, providing greater ability to specify languages.

<http://www.ietf.org/rfc/rfc3066.txt>

It would also make sense to limit the scope of the language-country naming convention to just language elements of locale.

Then locales would need to have additional names or identifiers for other non-linguistic elements such as currency.

A locale would then be a multi-name way to specify a collection of capabilities. Where a user requested capabilities that the software could not provide there should be some negotiation mechanism or protocol. Perhaps something similar to accept-language might work.

To make the semantics more specific, perhaps instead of a locale name, an XML file with more detailed specifications would be utilized.

Locales could be dated to establish the period of time for which they are current for a particular region.

More time will be devoted to these and other topics in Locale panel.

What is wrong with Locales?

References

- groups.yahoo.com/group/locales/
- www.i18nguy.com/locales/locale-resources.html
- <http://www.ietf.org/rfc/rfc1766.txt>
- <http://www.ietf.org/rfc/rfc3066.txt>
- <http://www.w3.org/TR/REC-xml#sec-lang-tag>
- <http://www.w3.org/TR/html401/struct/dirlang.html>
- lcweb.loc.gov/standards/iso639-2/langhome.html
- www.loc.gov/standards/iso639-2/englangn.html
- www.iso.org/iso/en/prods-services/iso3166ma/
- <http://www.ethnologue.com/web.asp>

25

The first URL is discussion group devoted to locales. If you are interested in the subject, join the group. <http://groups.yahoo.com/group/locales/>

The second URL is a page listing resources with information on locales.

<http://www.i18nGuy.com/locales/locale-resources.html>

The remainder are all URL's referenced in this presentation.

A more complete list is:

HTML <http://www.w3.org/TR/html401/struct/dirlang.html>

RFC 1766 <http://www.ietf.org/rfc/rfc1766.txt>

RFC 3066 <http://www.ietf.org/rfc/rfc3066.txt>

XML 1.0 <http://www.w3.org/TR/REC-xml#sec-lang-tag>

Unix http://www.opengroup.org/onlinepubs/007908799/xbd/locale.html#tag_005_001

Java <http://java.sun.com/products/jdk/1.1/intl/html/intlspec.doc2.html>

ISO 639-2 <http://lcweb.loc.gov/standards/iso639-2/langhome.html>

ISO 639-1 <http://www.loc.gov/standards/iso639-2/englangn.html>

ISO 3166 <http://www.iso.org/iso/en/prods-services/iso3166ma/index.html>

Ethnologue <http://www.ethnologue.com/web.asp>

Summary

- **Language-Country may not be adequate to describe user preferences**
- **Requirements for locale were listed**
- **Stability and cross-technology identifiers and semantics are needed**
- **Variants undermine interoperability**
- **Matching and fallback rules needed**
- **It's time to look for better solutions.**

26