Critical Values for I18n Testing

Tex Texin
Chief Globalization Architect
XenCraft

Abstract

In this session, we recommend specific data values that are likely to identify internationalization problems in software intended for global markets.

Based on years of global software experience, these data values are useful for functional or linguistic QA tests of internationalized software. In the last session, data value recommendations included character encoding, postal address, locale and other data types typically used in software and will trigger common internationalization problems. This presentation will offer specific new test suggestions.

Critical Values For I18n Testing

Values that provoke problems or raise issues

- Use real-world business values where possible
- Avoid exotic or contrived examples
- Test failures are therefore important to users

Typical value sets miss key problem types

- Volume testing creates the incorrect impression the software is robust

Critical Values For I18n Testing

This is not an internationalization QA course

Knowledge and practice of traditional i18n functional QA techniques is assumed

- Functional vs. Linguistic QA
- Pseudolocalization (Psèüdõlõcâlîzâtîõn)
- Testing with different locale settings
- Testing with native software (Operating System, Browsers, 3rd party software, etc.) and devices

- Security and other aspects not addressed

Expanded list at i18nQA.com

Agenda

- Introduction
- The usual suspects
- Text me…
- Identify yourself!
- The Date-Time Continuum
- Do you take Credit Cards?
- Parlez vous English

The usual suspects

Unicode Storage Test Values

Background

- UTF-8 characters can be 1-4 bytes (1-4 octets)
- UTF-16 characters can be 1-2 words (1-2 16-bit units)
- U+10000-U+1FFFFF are 4 bytes or 2 words

Typical software text storage errors

- Assume 1 character =1 byte or 1 word
- Assume 1 UTF-8 character <= 3 bytes
- Assume maximum length string will not occur and less is adequate
- Assume text operations do not change length (e.g. case)
- Memory allocation or string indexing “off by one” errors
- Inconsistent definitions of maximum length (e.g. UI <> DB)
- Performance degrades or severe memory leaks with long strings

Unicode Storage Test Values

Maximum length strings using maximum length characters (Supplementary characters > U+FFFF)

Frequently used 4-byte Supplementary characters

- U+2070E, U+20731, U+20779, U+20C53, U+20C78, U+20C96, U+20CCF, U+20CD5, U+20D15 and more

- See i18nguy.com/unicode/supplementary-test.html

- Provokes memory overruns, off by one errors, errors where the validation rules are incorrect

Test1: create maximum length strings with them

Unicode Storage Test Values

Test2: Test character substitution near boundaries

4 Characters 4 Bytes	A	A	A	A
4 Characters 5 Bytes	A	A	A	É

4 Characters 4 Bytes	A	A	A	A
4 Characters 5 Bytes	A	É	A	A

Test Case	Rationale
1. Fill field to maximum length with ASCII characters 2. Replace one character with a Supplementary character	Test that field supports full count of characters not bytes OR If counting bytes, that validation prevents overruns
1. Fill field to maximum length-1 (or -2, -3) with ASCII characters 2. Replace one character with a Supplementary character	Verify that validation prevents overruns

Case rules

This building is for ~~eels~~ congress. (congrès vs congres)

Palais des congrès

Background

- Case character maps can be more complex than English

Typical software errors

- Assuming that string length doesn’t change
- Assuming that case rules are reversible

Failure cases

- Upper(ß) => “SS” Lower(SS) => “ss”
- Length(eßen) = 4 Length(Upper(eßen)) = 5

Case rules

Example real world failure

- Case-insensitive compare used to determine if the record changed and should be saved
  - If (**Upper(CurrentVal) <> Upper(InputVal)**) save-changes;
  - Fails if the User edits the field and the change is not saved

Description	CurrentVal	InputVal	Result
Locale = fr_CA	"résumé"	"resume"	SUCCESS! "RÉSUMÉ"<>"RESUME"
Locale = fr_FR	"résumé"	"resume"	FAIL! "RESUME"="RESUME"
Different lowercase characters with the same uppercase	"essen"	"eßen"	FAIL! "ESSEN"="ESSEN"

Case rules

"ß" U+00DF is a useful character for i18n testing

Tests for poor case mappings assumptions

Test Data	Assumption	Reality	Failure Case
ß (U+00DF)	case rules are reversible	Upper(ß) => “SS” Lower(SS) => “ss”	X=Upper(Lower(X))
ß (U+00DF)	string length doesn’t change with case	Length(eßen) = 4 Length(Upper(eßen)) = 5	X = mem(4) X = Upper(“eßen”)
ß (U+00DF)	Unique strings remain unique with case changes	Upper(eßen)=ESSEN Upper(essen)=ESSEN	X=“eßen” Y=“essen” X<>Y Upper(X)=Upper(Y)

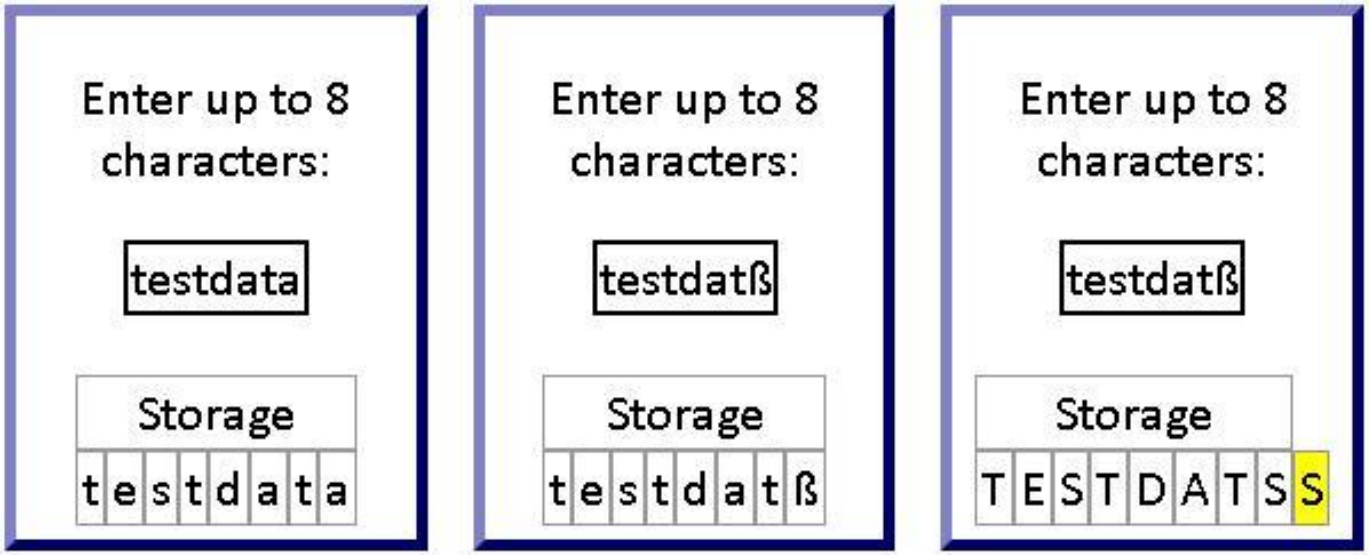
Example Tests For Case Rules

Test for uppercasing overrunning memory

Data Case 1: Use maximum length run of "ß"

- "ßßßßß" 5 chars become 10: "SSSSSSSSSS"

Data Case 2: Edit a maximum length string, replace one character with "ß"



Locale-dependent case rules

Background

- Case mappings are locale-dependent
- Turkish is special as it changes case of ASCII characters

Recommended testing with Turkish locale

- Even if you don't localize for or plan to support Turkey

Test Data	Assumption	Reality	Failure Case
i (U+0069) I (U+0049) İ (U+0130) ı (U+0131)	Case rules for ASCII characters do not change	Locale= **Turkish** Upper(i)=İ Upper(ı)=I	Upper(exit)<>EXIT

Text me!

Character Tests

Syntax Character Tests

Software uses many programming languages and protocols. Each has its own syntax with "special" characters and escapes.

Note: A syntax can rely on a sequence of characters.
Note: A syntax can rely on pairings of characters.
e.g. " " or /* */

Language / Protocol	Example Syntax Characters	escape	Examples
HTML, XML	& < > ' "	&	& €
URL	? & # + / . : = %	%	tex.com?a=san+jos%C3%A8+%26+CA
Javascript	' " \ // /* */ \u	\	\n \\ \u20AC

Syntax Character Tests

Potential Failure cases

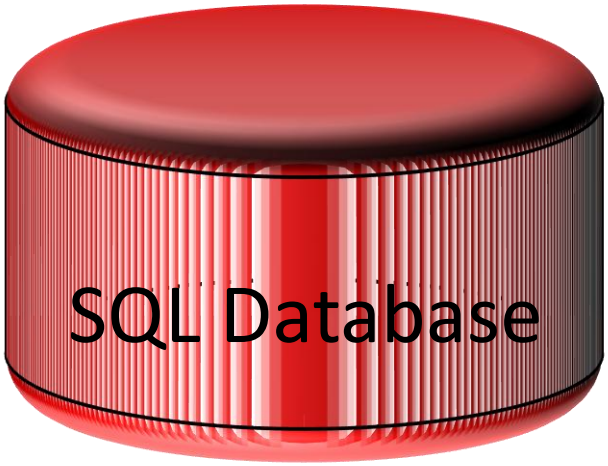
- Syntax characters are not escaped
- Syntax characters are escaped twice
- Syntax characters used in 2 or more languages, are escaped and unescaped in different orders

Problems can occur because data comes from different sources, goes thru different code paths, and is intended for different uses.

- E.g. Text in properties files destined for HTML vs e-mail vs javascript alert vs URL, et al
  - Each has different escape requirements

Syntax Character Tests

Storage Formats



Properties Files

Logic Stack

Java

PHP

Regex

Libraries

Output Formats

URL

Text

HTML + Javascript

E-mail

Syntax Character Tests

Recommendation: Test with syntax-significant characters everywhere

& + / \ " ' ( ) ? . # @ _ - ~

Escape/Unescape example Ampersand "&"

Character	HTML Escape	URI Escape
&	&amp;	%26

Example: M&M	Step 1	Step 2
Encode M&M HTML followed by URI	M&M	M%26amp;M
Decode M%26amp;M URI first	M&M	M&M
Decode M%26amp;M HTML first	M%26amp;M	M&M

The Invisible Man and Invisible Girl

White space (space, tab, return, et al) often needs trimming or other special handling

1. No-Break Space (U+00A0,) often missed
   - Is Group separator in some locales (e.g. **1 234 567,89**)
2. Full width (ideographic) space (U+3000)
3. Other spaces

En Space U+2002
Em Space U+2003
Three-Per-Em Space U+2004
Four-Per-Em Space U+2005
Six-Per-Em Space U+2006
Figure Space U+2007
Punctuation Space U+2008
Thin Space U+2009
Hair Space U+200A
Zero Width Space U+200B

The Invisible Man and Invisible Girl

- Test Data NBSP, Full Width Space

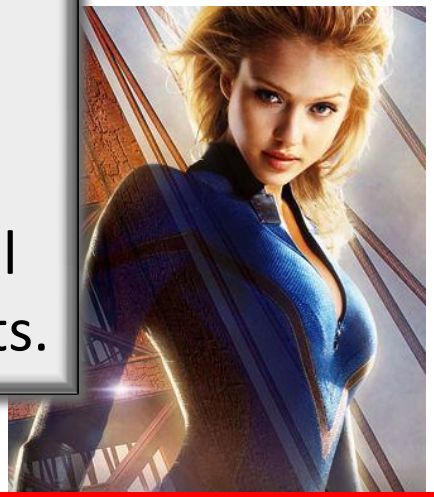
Although No-Break Space is not usually typed in by users, it is often copy/paste into fields.

If NBSP or Full Width Space are not trimmed, searches can fail, or duplicate entries can occur

Duplicate Entries	String	String+NBSP	String+FWSP	NBSP+String

Search for "String", won't match "String+NBSP".

Create User "String+NBSP" won't fail (as expected) if "String" already exists.



Wide Loads: Full Width Characters

ASCII (et al) characters are duplicated

Full Width: ！ " # ＄ % ＆ ' （ ） * ＋ , － . ／ ０ １ ２ ３ ４ ５ ６ ７ ８ ９ ： ； ＜ ＝ ＞ ？
＠ Ａ Ｂ Ｃ Ｄ Ｅ Ｆ Ｇ Ｈ Ｉ Ｊ Ｋ Ｌ Ｍ Ｎ Ｏ Ｐ Ｑ Ｒ Ｓ Ｔ Ｕ Ｖ Ｗ Ｘ Ｙ Ｚ ［ ＼ ］ ＾ ＿ ｀ ａ ｂ
ｃ ｄ ｅ ｆ ｇ ｈ ｉ ｊ ｋ ｌ ｍ ｎ ｏ ｐ ｑ ｒ ｓ ｔ ｕ ｖ ｗ ｘ ｙ ｚ ｛ ｜ ｝ ～

ASCII: !"#$%&'()*+,-./0123456789:;< = >?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijkl
mnopqrstuvwxyz{|}~

- Test Case: Use full width (Ideographic) "**＊**" and other characters in keywords and operators
  - Full Width characters may be syntax equivalents.
    - Inconsistent across products, even where standards exist

```
SELECT ＊ FROM Employees WHERE LastName='龍' AND
(FirstName='陳元' OR FirstName='Jackie')
```

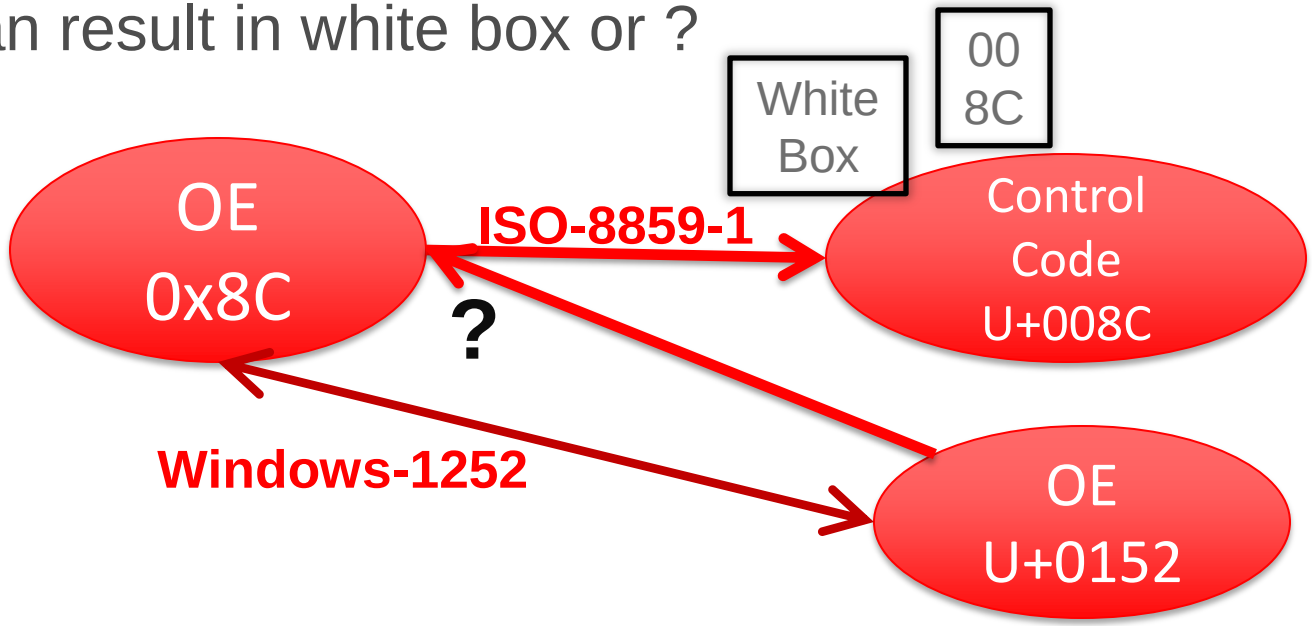
ISO-8859-1 vs Windows-1252

Very common error: incorrect charset name

- For CSV, HTML, XML, e-mail, and other file formats, DB, HTTP and other protocols

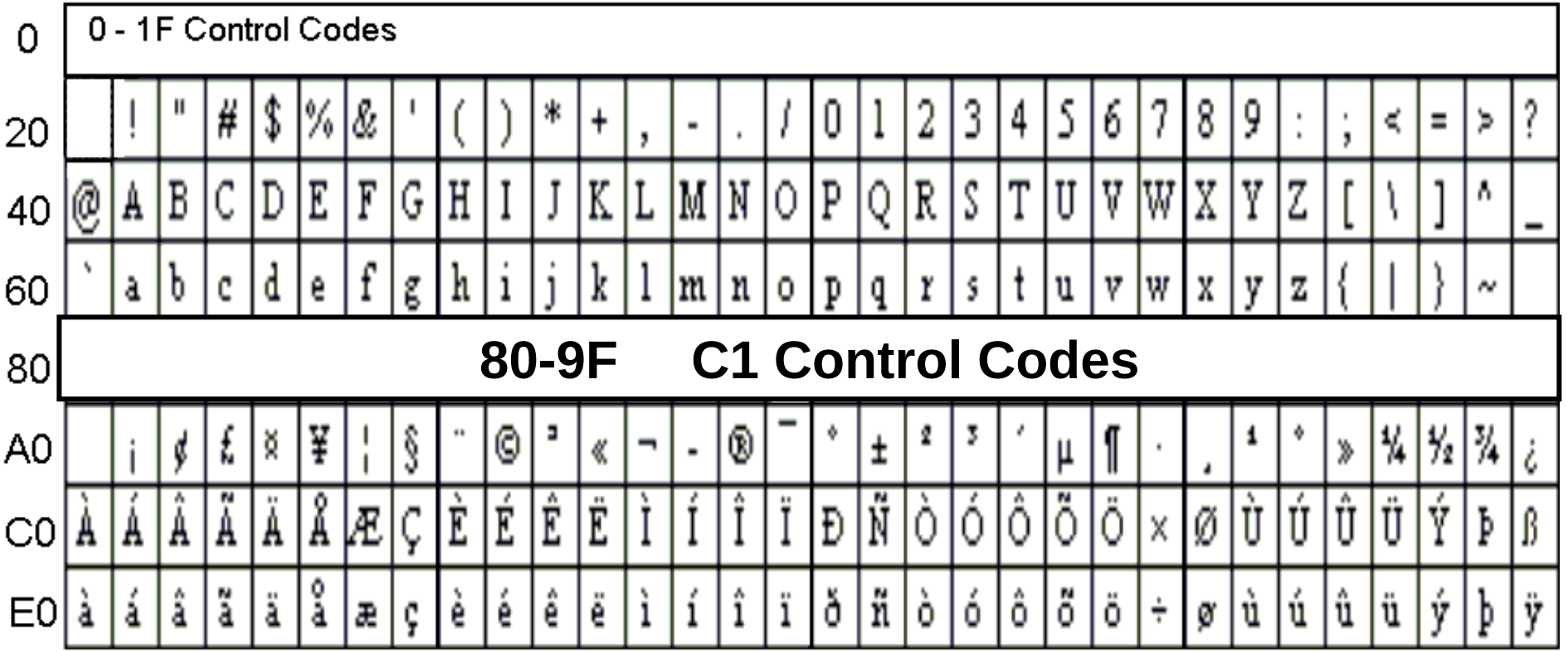
Background

- Convert "ISO-8859-1" from/to Unicode is incorrect
  - Can result in white box or ?



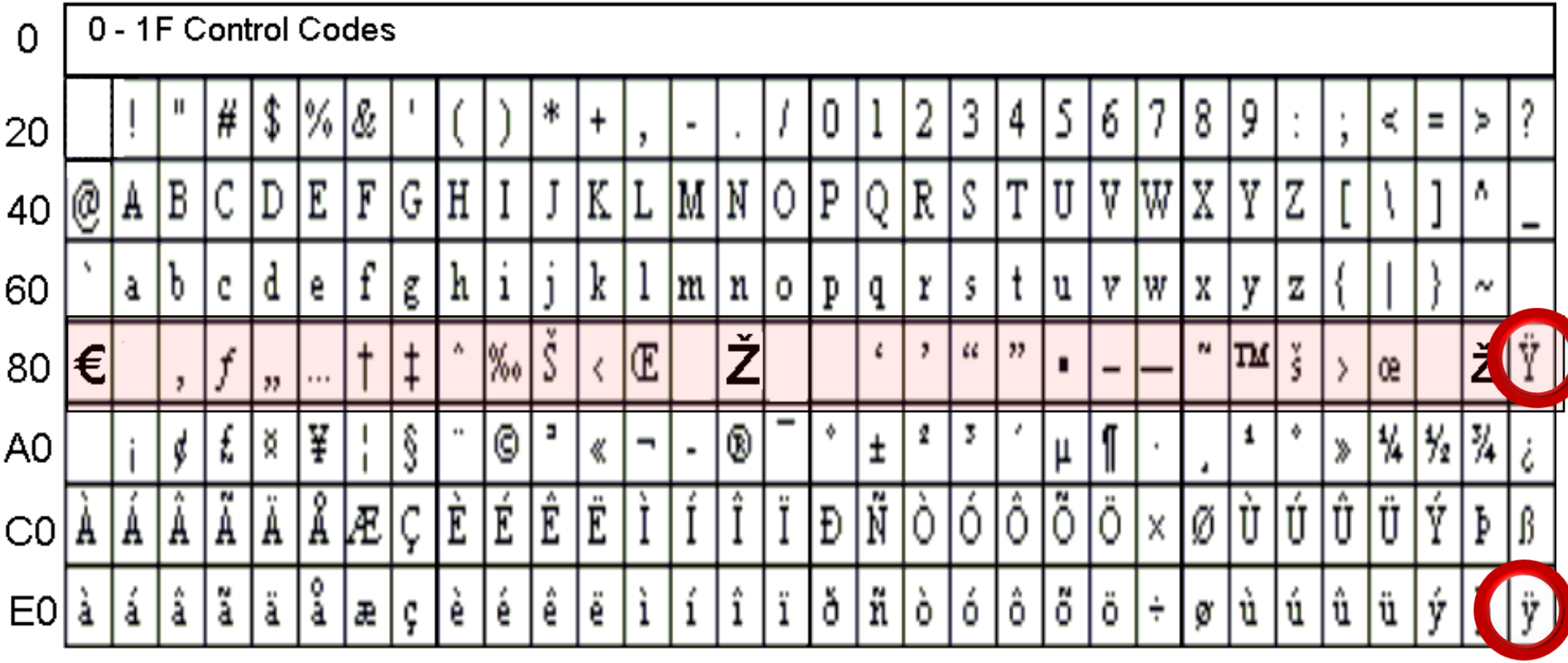
ISO-8859-1

Every code point is used. C1 control codes are legacy of mainframe and minicomputers



Windows 1252

C1 control codes replaced by characters. Some code points are not (yet) assigned (81, 8D, 8F, 90, 9D)



ISO-8859-1 vs Windows-1252

Test Recommendations

- Use characters in 0x80-0x9F (128-159)
  - Dashes: – —
  - Punctuation: Smart quotes ‘’“”
  - Ellipses …
  - Left/Right Single Pointing Quotation Marks: ‹ ›
  - Characters: OE Ligature "Œ œ", Trademark "™"

1. Œ easy to use in text data & pseudolocalization
2. Use Ÿ and ÿ for case tests
   - ÿ U+00FF is in both encodings
   - Ÿ U+0178 0x9F (159) is only in Windows-1252

Double Decode Detector

Common error is double conversion to UTF-8

- Often data goes in and out without error being noticed

	Start	Convert Windows-1252 to UTF-8		Convert Back to Windows-1252	
		1st time	2nd time	1st time	2nd time
Most characters succeed	Â 0xC3	Âƒ 0xC3 0x83	Âƒ‚ 0xC3 0x83 0xC2 0x83	Âƒ 0xC3 0x83	Â 0xC3
Test Data showing error	Á 0xC1	Â? 0xC3 0x81	**Error 0x81 doesn't exist**		

- Recommend Test Character Á (U+00C1) A-acute
  - Or others with byte value of 81, 8D, 8F, 90, 9D

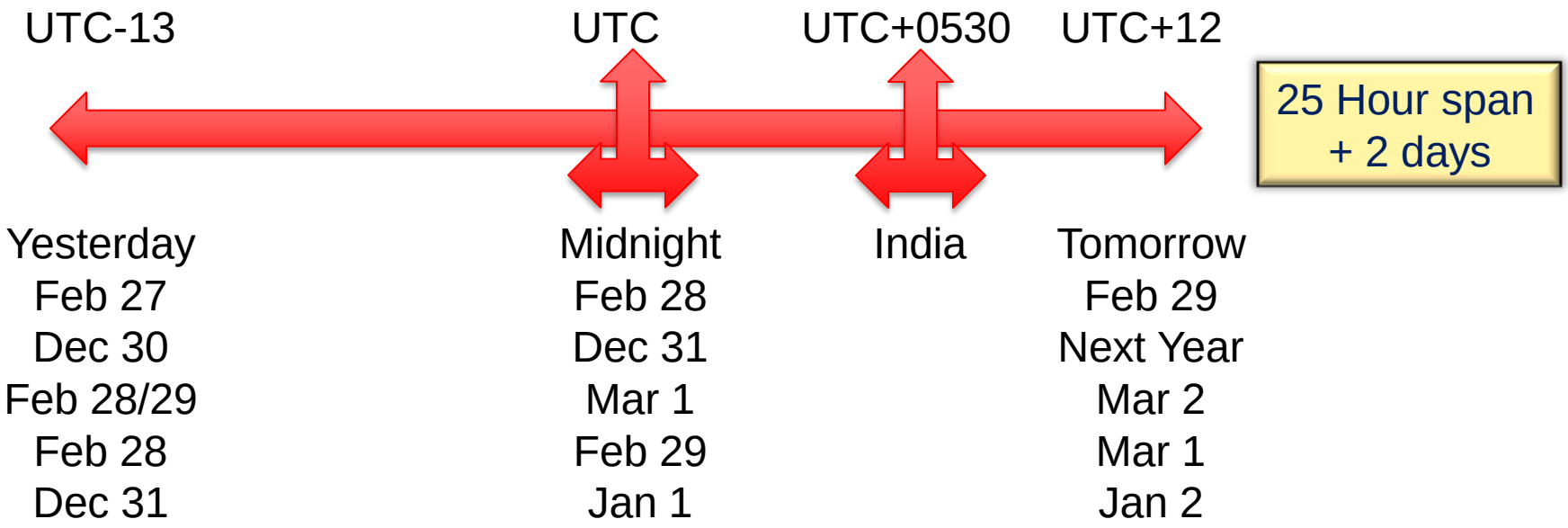
From time out of mind

Date-time Tests

The Date-Time Continuum

Time zone affects date as well as time

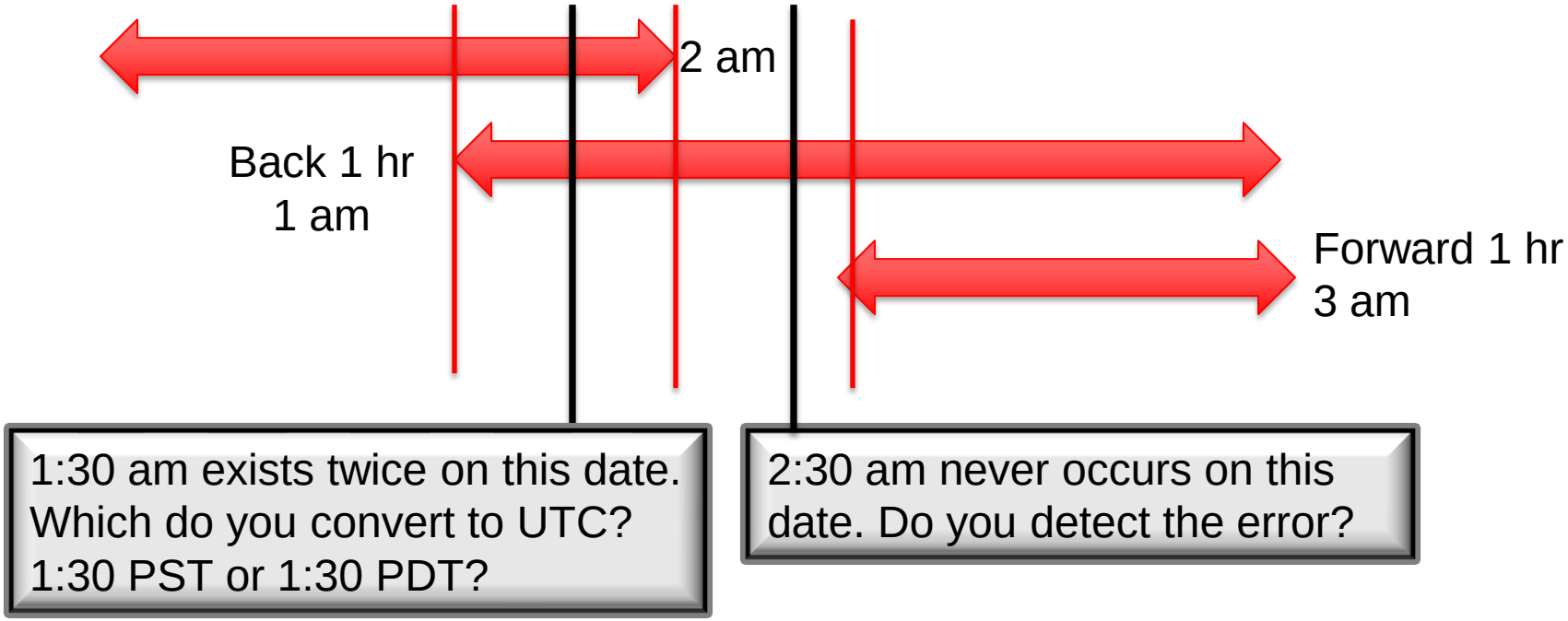
- Often overlooked, but important to test
- Use data that crosses datelines, dates, months
- Use time zones with offsets other than 00



The Date-Time Continuum

Daylight Savings

- Test gaps in single time zone
- Test gaps crossing multiple time zones
  - E.g. PST-BST, PST-BDT, PDT-BST, PDT-BDT



The Date-Time Continuum

- Time zones change
- Test changing periods of Daylight Savings
  - How do you find dates that need updating?

Knock Knock! Who’s there?

Identity Tests

Identify yourself! Names, Postal Addresses

Vast differences in conventions

- Standards are not consistent
  - Due to differences in applications and use cases

Recommended Tests

- Country Code: UK
  - ISO 3166 code is GB Postal code is UK

    Source of errors
    - ISO Code GB causes mail to be rejected.
    - Postal Code UK will fail validations using ISO Codes
  - DB & XML Identifiers should distinguish them
    - NOT <Country Code> but <PostalCountryCode> <ISOCountryCode>
- Countries with no post (zip) code
  - Ireland, Somalia, Syria (Although these may be changing)

Identify yourself! Names, Postal Addresses

Field sizes are often inadequate

- Longest Surname MacGhilleseatheanaich
- Longest Street Name (Poland)

  Dwudziestego Pierwszego Praskiego Pulku Piechoty imienia Dzieci Warszawy
- Long Street Name with no spaces

  Carl-Philipp-Emanuel-Bach-Straße Frankfurt (Oder), Germany
- Longest City Name (Anglesey, Wales 58 characters)
  - Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch

E-mail

E-mail tests

- Test syntax characters in email
  - "T**é**x" <Tex**+**1@xencraft.com>
- Test maximum size (254 bytes)
- Test International domain names in mail
  - TEX@globalização.biz (Need your own domain)
  - tex@xn--globalizao-n5a1c.biz
  - Verify equivalence for search

Web Address / International Domain Names

Test Cases for International Domain Names (IDN)

- Max length (255-2048)
  - Mobile and Personal device browsers have lower limits
    - http://thelongestlistofthelongeststuffatthelongestdomainnameatlonglast.com
    - http://www.thequickbrownfoxjumpsoveralazydog.com
- Syntax characters, escapes (&, /, + # ? . % :)
- International TLD, domain and subdomain names
  - http://www.xn--globalizao-n5a1c.biz
  - http://www.globalização.biz
  - http://globalização.globalização.biz
    - Testing subdomains are useful if you don't have an IDN
  - Test search equivalence of international and ASCII-fied names
  - Evaluate display
  - Test international domain names as parameters in URLs
    - google.com/search?q=http%3A%2F%2Fglobaliza%C3%A7%C3%A3o.biz

Web Addresses

Test Cases

- International path

  www.xencraft.com/globalização/globalização

  - Verify links in non-UTF-8 pages first convert to UTF-8

    Ç is %C3%A7 not %C7

- International query and fragment, including non-UTF-8

  www.xencraft.com/globalização/?a=çç#ã%C3%A7

Questions?

Tex Texin

TexTexin@Xencraft.com

Tex is an industry thought leader specializing in business and software globalization services. His expertise includes global product strategy, Unicode and internationalization architecture, and cost-effective implementation and testing. Over the past two decades, Tex has created numerous global products, led internationalization development teams, and guided companies in taking business to new regional markets.

Tex is a contributor to internationalization standards for software and on the Web.

Tex is a popular speaker at conferences around the world and provides on-site training on Unicode, internationalization, and globalization QA worldwide.

Tex is the author of the popular, instructional web site www.I18nGuy.com

Tex is founder and Chief Globalization Architect for XenCraft. XenCraft provides global business consulting and software design, implementation, test and training services on globalization product strategy and software internationalization architecture.