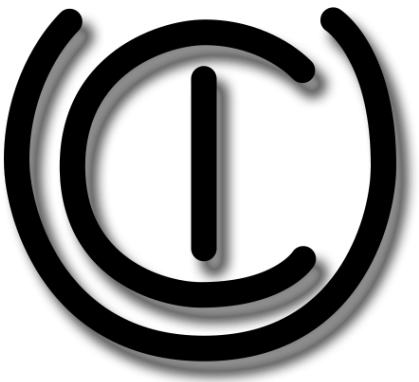


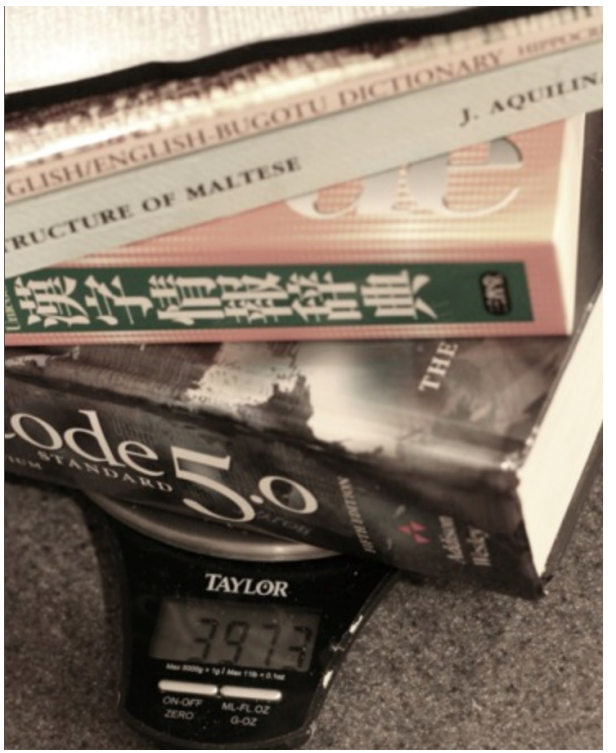
Put ICU to Work!

Steven R. Loomis, IBM

Shane Carr, Google

Can’t I just “use Unicode” and be done?

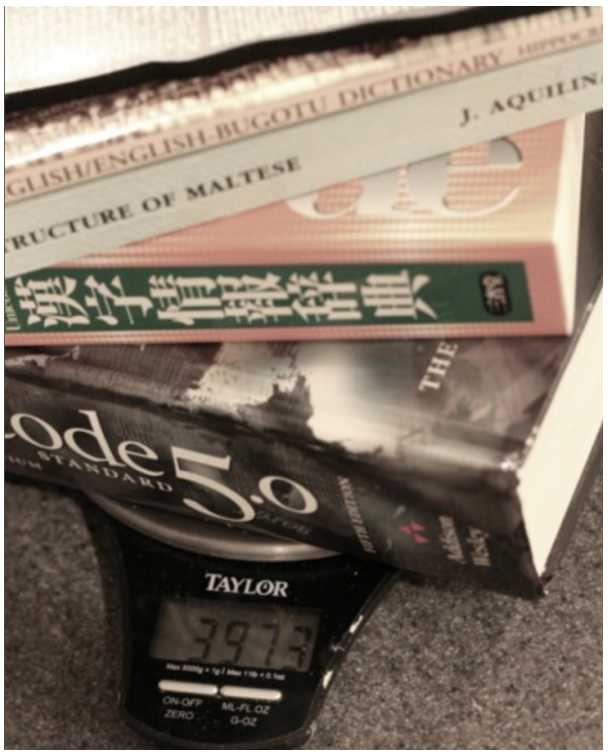
Can’t I just “use Unicode” and be done?



Put ICU to Work!

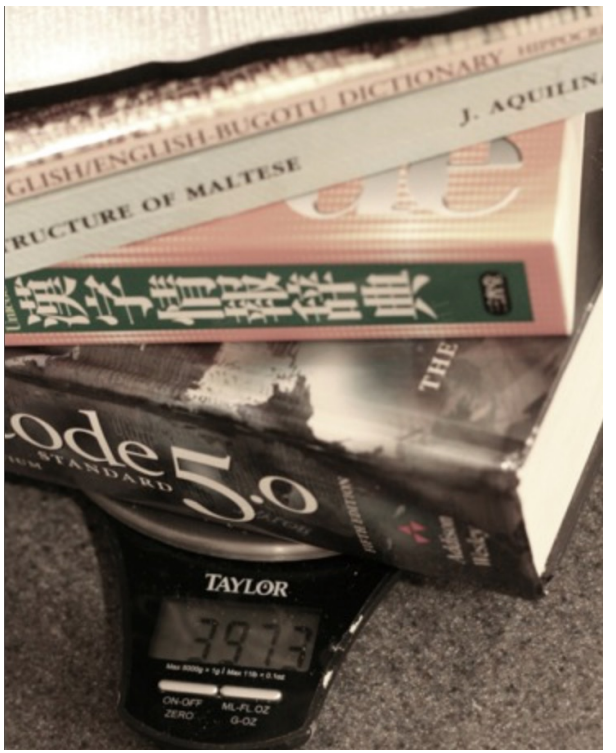
Can’t I just “use Unicode” and be done?

- 3,639 pages (core+charts)



Can’t I just “use Unicode” and be done?

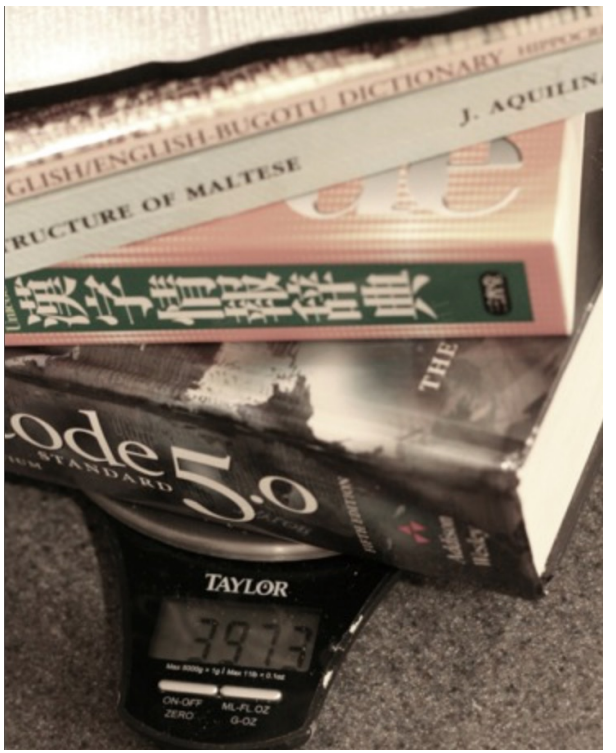
- 3,639 pages (core+charts) + Annexes



Put ICU to Work!

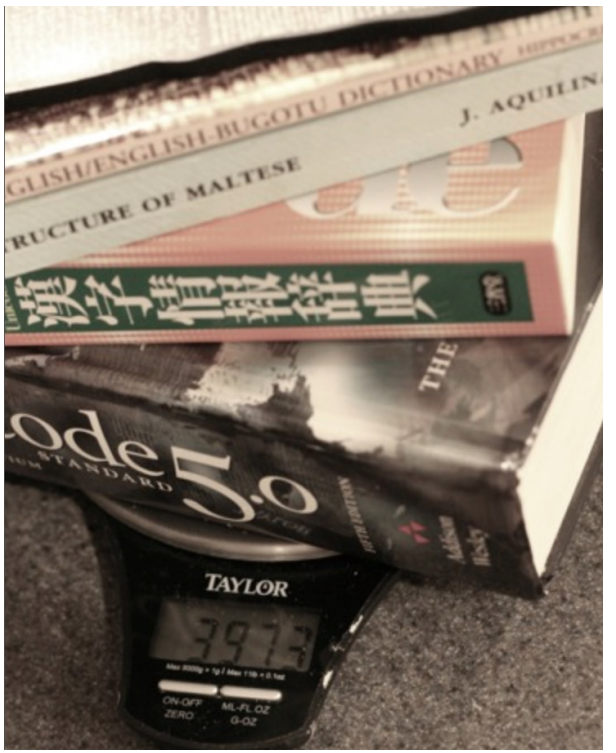
Can’t I just “use Unicode” and be done?

- 3,639 pages (core+charts) + Annexes
- More than 143,000 characters



Can’t I just “use Unicode” and be done?

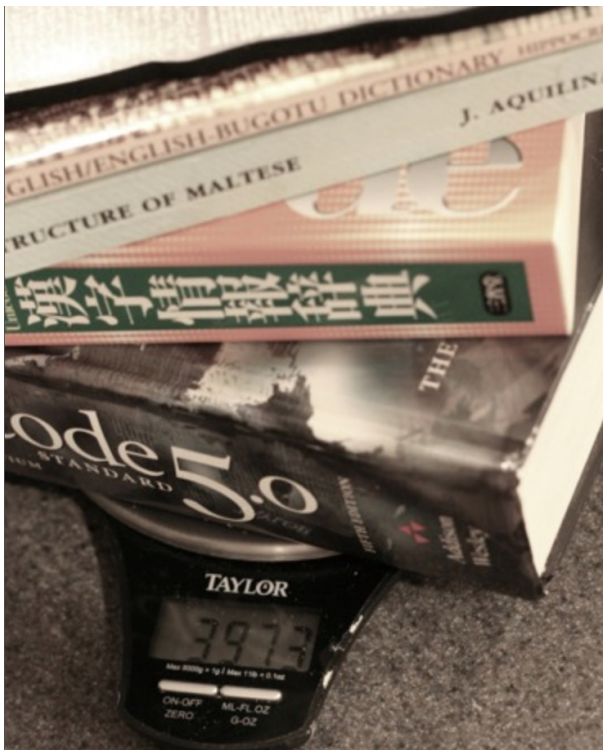
- 3,639 pages (core+charts) + Annexes
- More than 143,000 characters
- Significant update about once a year



Put ICU to Work!

Can’t I just “use Unicode” and be done?

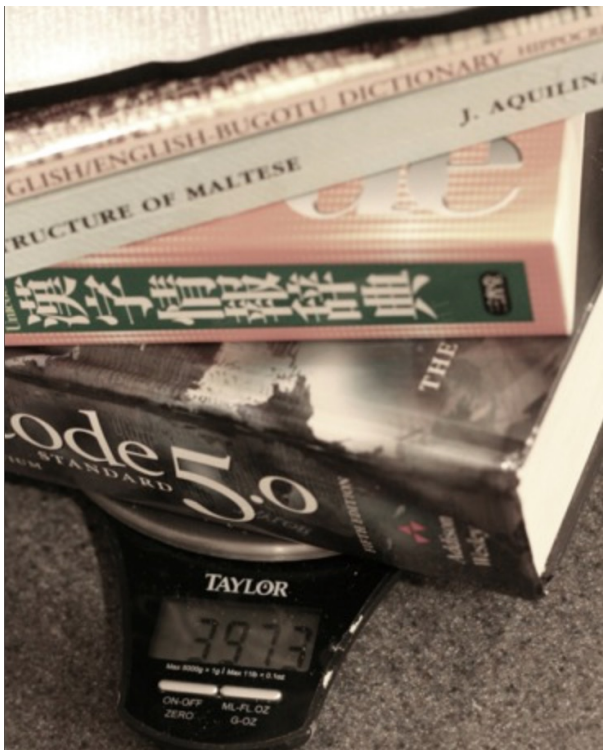
- 3,639 pages (core+charts) + Annexes
- More than 143,000 characters
- Significant update about once a year
- 80+ character properties, many multi-valued



Put ICU to Work!

Can’t I just “use Unicode” and be done?

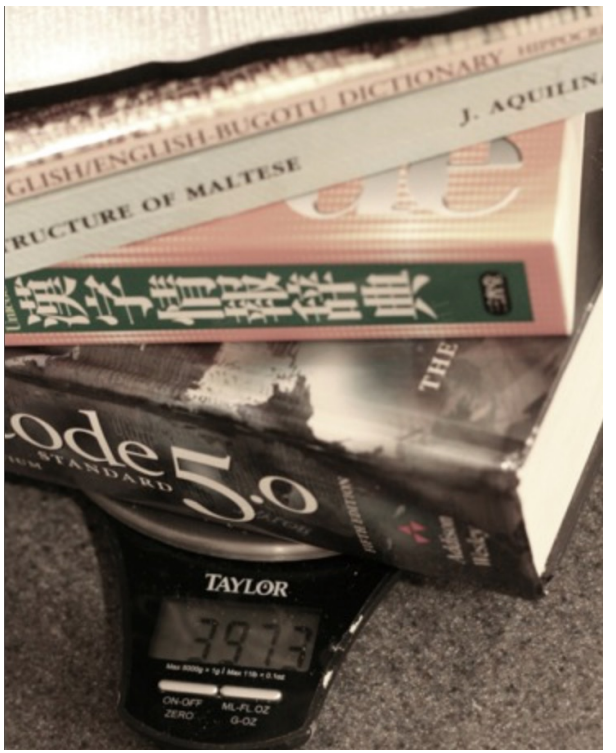
- 3,639 pages (core+charts) + Annexes
- More than 143,000 characters
- Significant update about once a year
- 80+ character properties, many multi-valued
- Additional standards and geopolitical requirements
  - CLDR, ISO, TZ, …



Put ICU to Work!

Can’t I just “use Unicode” and be done?

- 3,639 pages (core+charts) + Annexes
- More than 143,000 characters
- Significant update about once a year
- 80+ character properties, many multi-valued
- Additional standards and geopolitical requirements
  - CLDR, ISO, TZ, …
- Good vs. fast



Put ICU to Work!

Unicode covers the world 🌍

- *“Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.”* (unicode.org)

Unicode covers the world 🌍

- *“Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.”* (unicode.org)

ICU brings you home 🏡

- Requirements vary widely across languages & countries
- Sorting
- Text searching
- Bidirectional text processing
- Date/time/number/currency formatting
- Codepage conversion
- …many more

I See Unicode

- 1999: *IBM Classes for Unicode* open-sourced as the *International Components for Unicode*
- 2016: ICU joins Unicode as ICU-TC
- 2018: Development now on GitHub and Jira

ICU's Laundry List

- Breaks: word, line, …
- Formatting
  - Date & time
  - Durations
  - Messages
  - Numbers & currencies
  - Plurals
- Transforms
  - Normalization
  - Casing
- Transliterations
- Unicode text handling
- Charset conversions (200+)
- Charset detection
- Collation & Searching
- Locales from CLDR (640+)
- Resource Bundles
- Calendar & Time zones
- Unicode Regular Expressions …

Benefits of ICU

- Mature, widely used, up-to-date set of C/C++ and Java libraries
  - Basis for Java 1.1 internationalization, but goes far beyond Java 1.1
  - Team continues to work on improving and monitoring performance.
- Very portable – identical results on all platforms/programming languages
  - C/C++ (ICU4C): many platforms/compilers
  - Java (ICU4J): Oracle Java SE, IBM JRE, OpenJDK, Android
  - Wrappers: D/C#/PHP/Python/…
- Customizable & Modular
  - Open source (since 1999) – but non-restrictive
  - Contributions from many parties (IBM, Google, Apple, Microsoft, …)
- Sponsored by Unicode

Where do I get ICU?

Main site: http://icu-project.org/

- Downloads, API references, Mailing list, Bug tracking
- Userguide: http://userguide.icu-project.org
  - Moving soon to GitHub Pages
  - User's guide with examples

Prepackaged ICU

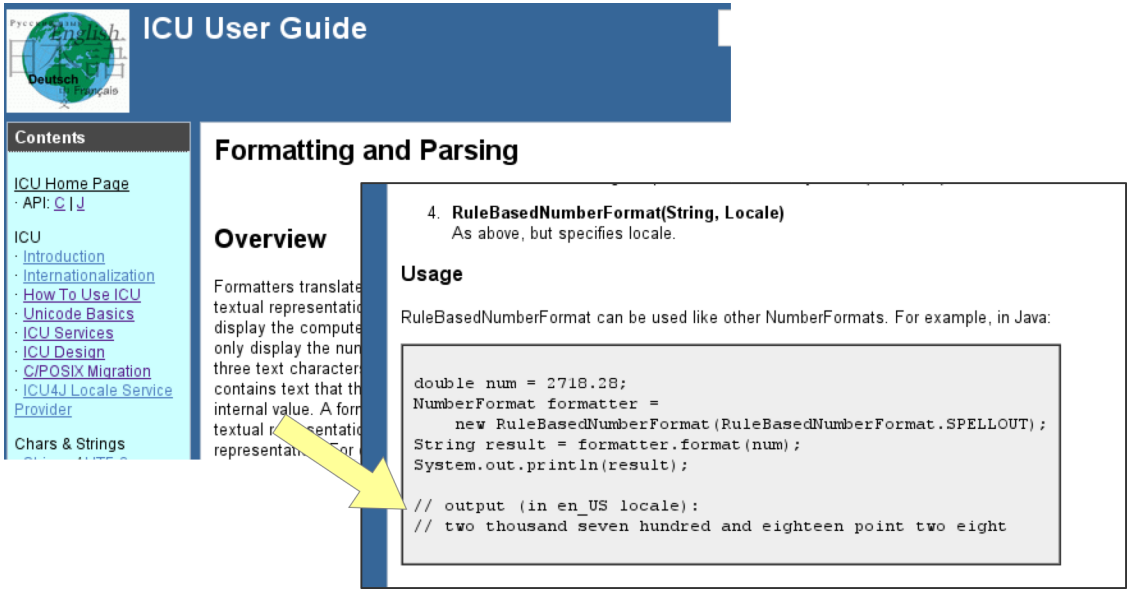
Package Managers (C)

- `brew install icu4c`
- `apt-get install libicu-dev`
- `dnf install libicu-devel`

Maven and friends: (J)

- group: *`com.ibm.icu`*
- artifactId: *`icu4j`*

ICU Userguide



Put ICU to Work!

API Docs

RuleBasedNumberFormat

```
public RuleBasedNumberFormat(String description,
                             String[][] localizations)
```

Creates a RuleBasedNumberFormat that behaves according to the d
formatter uses th

The localizations
for different loca
element in this ar
localizations of t
the first String be
names, in the sar

Parameters:

RuleBasedNumberFormat::RuleBasedNumberFormat (const UnicodeString & rules,
UParseError & perror,
UErrorCode & status
)

Creates a RuleBasedNumberFormat that behaves according to the description passed in.

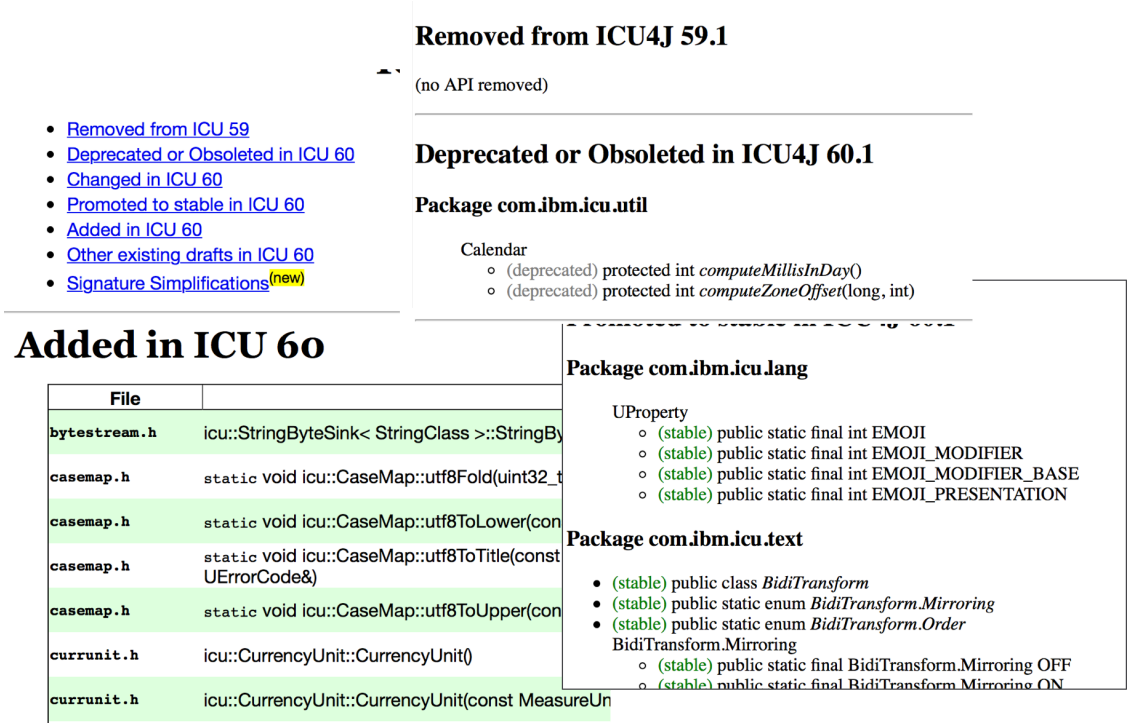
The formatter uses the default locale.

Parameters:

rules	A description of the formatter's desired behavior. See the class documentation for a complete explanation of syntax.
perror	The parse error if an error was encountered.
status	The status indicating whether the constructor succeeded.

Stable:
ICU 3.2

API Change Report



- Removed from ICU 59
- Deprecated or Obsoleted in ICU 60
- Changed in ICU 60
- Promoted to stable in ICU 60
- Added in ICU 60
- Other existing drafts in ICU 60
- Signature Simplifications(new)

Added in ICU 60

File	
bytestream.h	icu::StringByteSink< StringClass >::StringBy
casemap.h	static void icu::CaseMap::utf8Fold(uint32_t
casemap.h	static void icu::CaseMap::utf8ToLower(con
casemap.h	static void icu::CaseMap::utf8ToTitle(const UErrorCode&)
casemap.h	static void icu::CaseMap::utf8ToUpper(con
currunit.h	icu::CurrencyUnit::CurrencyUnit()
currunit.h	icu::CurrencyUnit::CurrencyUnit(const MeasureUn

Removed from ICU4J 59.1

(no API removed)

Deprecated or Obsoleted in ICU4J 60.1

Package com.ibm.icu.util

Calendar
- (deprecated) protected int *computeMillisInDay*()
- (deprecated) protected int *computeZoneOffset*(long, int)

Package com.ibm.icu.lang

UProperty
- (stable) public static final int EMOJI
- (stable) public static final int EMOJI_MODIFIER
- (stable) public static final int EMOJI_MODIFIER_BASE
- (stable) public static final int EMOJI_PRESENTATION

Package com.ibm.icu.text

- (stable) public class *BidiTransform*
- (stable) public static enum *BidiTransform.Mirroring*
- (stable) public static enum *BidiTransform.Order*

BidiTransform.Mirroring
- (stable) public static final BidiTransform.Mirroring OFF
- (stable) public static final BidiTransform.Mirroring ON

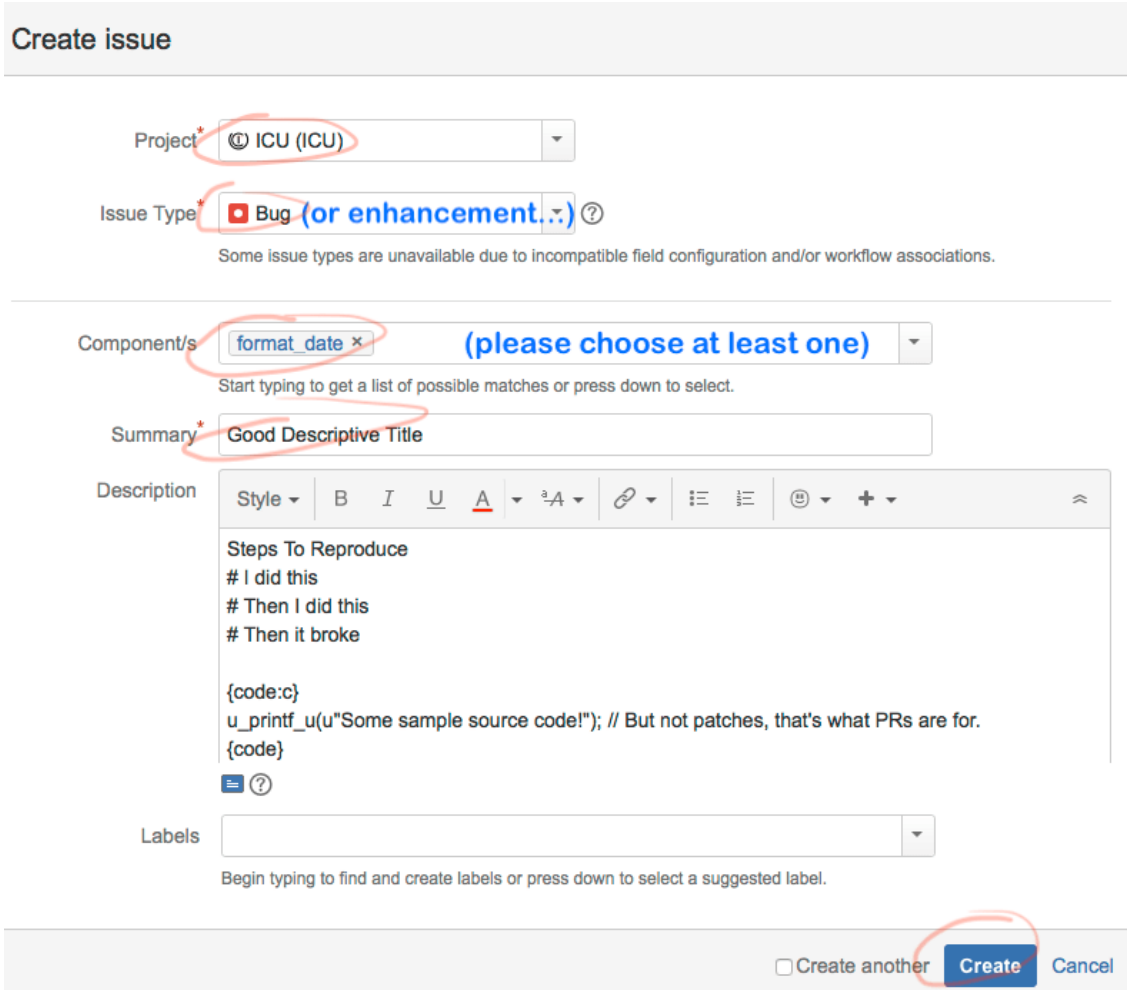
Put ICU to Work!

Mailing Lists

http://site.icu-project.org/contacts

- `icu-support` – Technical support and discussion
- `icu-design` – API proposal discussion
- `icu-announce` – Low-volume announcements list

Issues (Jira)



Put ICU to Work!

Contributing

1. Open an issue in Jira
2. Fork the ICU repo
3. Write and test your code
4. Commit your change to your fork
5. Open a new Pull Request
6. Sign the CLA when prompted 

Contributing

1. Open an issue in Jira
2. Fork the ICU repo
3. Write and test your code
4. Commit your change to your fork
5. Open a new Pull Request
6. Sign the CLA when prompted 

7. Bask in your newfound fame and fortune!

And now, code

Task at Hand

- *Display a list of world regions, with their population figures*

Task at Hand

- *Display a list of world regions, with their population figures*

Example

- 150,000: Ceuta and Melilla
- 38,087,800: Algeria
- 15,439,400: Ecuador

ICU4C First Look

```
#include <unicode/…>

void func() {
  UErrorCode status = U_ZERO_ERROR;
  u_init(&status);
  if ( U_SUCCESS(status) ) { /* … */ }
}
```

ICU4C First Look

```
#include <unicode/…>

void func() {
  UErrorCode status = U_ZERO_ERROR;
  u_init(&status);
  if ( U_SUCCESS(status) ) { /* … */ }
}
```

#include <unicode/…>

- All ICU headers are in the `unicode/` subdirectory

ICU4C First Look

```
#include <unicode/…>

void func() {
  UErrorCode status = U_ZERO_ERRORκ
  u_init(&status)κ
  if ( U_SUCCESS(status) ) { /* … */ }
}
```

`UErrorCode status = U_ZERO_ERRORκ`

- Error code is a fill-in, but must be initialized
- If in C++, `icu::ErrorCode` is available (example on next slide)

ICU4C First Look

```
#include <unicode/…>

void func() {
  UErrorCode status = U_ZERO_ERRORκ
  u_init(&status)κ
  if ( U_SUCCESS(status) ) { /* … */ }
}
```

u_init(&status)κ

- Returns successful `status` if ICU data loaded OK

ICU4C First Look

```
#include <unicode/…>

void func() {
  UErrorCode status = U_ZERO_ERROR;
  u_init(&status);
  if ( U_SUCCESS(status) ) { /* … */ }
}
```

`if ( U_SUCCESS(status) )`

- TRUE if there was no error

Error codes in C++

No need to initialize! Less prone to error:

```
#include <unicode/…>

int main() {
  icu::ErrorCode statusκ
  u_init(status)κ
  if (status.isFailure()) {
    return 1κ
  }
  return 0κ
}
```

ASSERT_OK()

C++ version:

```
#define ASSERT_OK(status) \
 if(status.isFailure()) { \
     puts(status.errorName())κ \
     return 1κ \
 }
```

Plain C version:

```
#define ASSERT_OK(status) \
 if(U_FAILURE(status)) { \
     puts(u_errorName(status))κ \
     return 1κ \
 }
```

ASSERT_OK()

C++ version:

```
#define ASSERT_OK(status) \
 if(status.isFailure()) { \
     puts(status.errorName())κ \
     return 1κ \
 }
```

Plain C version:

```
#define ASSERT_OK(status) \
 if(U_FAILURE(status)) { \
     puts(u_errorName(status))κ \
     return 1κ \
 }
```

- always check for failure

ASSERT_OK()

C++ version:

```
#define ASSERT_OK(status) \
 if(status.isFailure()) { \
     puts(status.errorName())κ \
     return 1κ \
 }
```

Plain C version:

```
#define ASSERT_OK(status) \
 if(U_FAILURE(status)) { \
     puts(u_errorName(status))κ \
     return 1κ \
 }
```

- always check for failure
- (We will use this macro to keep test code more compact)

so9_test.c

```c
#include <unicode/ustdio.h>

int main(int argc, const char *argv[]) {
    u_printf_u(u"This is ICU %s! 😸 \n", U_ICU_VERSION)ĸ
    return 0ĸ
}
```

so9_test.c

```c
#include <unicode/ustdio.h>

int main(int argc, const char *argv[]) {
    u_printf_u(u"This is ICU %s! 😸 \n", U_ICU_VERSION)ĸ
    return 0ĸ
}
```

This is ICU 64.2! 

so9_test.c

```c
#include <unicode/ustdio.h>

int main(int argc, const char *argv[]) {
    u_printf_u(u"This is ICU %s! 😸 \n", U_ICU_VERSION)ĸ
    return 0ĸ
}
```

This is ICU 64.2! 

- *but, let's actually build this*

Building `s09_test.c`

```
$ brew install icu4c pkg-config
```

Building `s09_test.c`

```
$ brew install icu4c pkg-config
```

```
$ git clone https://github.com/unicode-org/icu-demos.git -b iuc43
```

Building `s09_test.c`

```
$ brew install icu4c pkg-config
```

```
$ git clone https://github.com/unicode-org/icu-demos.git -b iuc43
```

```
$ cd iucsamples/c/s09_test
$ make check
This is ICU 64.2! 😺
everything is OK 🎉
```

Building `s09_test.c`

```
$ brew install icu4c pkg-config
```

```
$ git clone https://github.com/unicode-org/icu-demos.git -b iuc43
```

```
$ cd iucsamples/c/s09_test
$ make check
This is ICU 64.2! 😺
everything is OK 🎉
```

under the hood:

- paths detected via pkg-config

```
cc -I/usr/local/Cellar/icu4c/64.2/include -o s09_test s09_test.c -L/usr/local/
```

s13a_hello.cpp

```cpp
#include <unicode/errorcode.h>
#include <unicode/locid.h>
#include <unicode/ustdio.h>
#include <unicode/ustream.h>
#include <iostream>

int main() {
  icu::ErrorCode status;
  icu::Locale locale("und_001");
  icu::UnicodeString world;
  locale.getDisplayCountry(world);
  ASSERT_OK(status);

  std::cout << "Hello, " << world << "!" << std::endl;
  return 0;
}
```

s13a_hello.cpp

```cpp
#include <unicode/errorcode.h>
#include <unicode/locid.h>
#include <unicode/ustdio.h>
#include <unicode/ustream.h>
#include <iostream>

int main() {
  icu::ErrorCode status;
  icu::Locale locale("und_001");
  icu::UnicodeString world;
  locale.getDisplayCountry(world);
  ASSERT_OK(status);

  std::cout << "Hello, " << world << "!" << std::endl;
  return 0;
}
```

Hello, World!

```
$ LC_ALL=es ./s13a_hello
```

Hello, Mundo!

icuhelloworld.cpp

```
$ LC_ALL=mt ./s13a_hello
```

Hello, Dinja!

```
$ LC_ALL=zh ./s13a_hello
```

Hello, 世界!

What if we want to change **“Hello”**?

String Concatenation 🙀

- Order is different for different languages, can't just concatenate strings.

String Concatenation 🙀

- Order is different for different languages, can't just concatenate strings.

My **Aunt's** **pen** is on the table.

String Concatenation 🙀

- Order is different for different languages, can't just concatenate strings.

My **Aunt's** **pen** is on the table.

```
whom + "'s " + what + " is on the " + where
```

String Concatenation 🙀

- Order is different for different languages, can't just concatenate strings.

My **Aunt's** **pen** is on the table.

```
whom + "'s " + what + " is on the " + where
```

La **pluma** de **mi tía** está sobre la mesa.

Pattern Syntax

Pattern Syntax

```
en: {whom}'s {what} is on the {where}.
```

Pattern Syntax

```
en: {whom}'s {what} is on the {where}.

es: {what} de {whom} está sobre la {where}.
```

Pattern Syntax

```
en: {whom}'s {what} is on the {where}.

es: {what} de {whom} está sobre la {where}.
```

Or, avoid sentences entirely

"Location: table, Object: pen, Owner: Aunt"

hellomsg.cpp

```
    const int kArgCount = 1κ
    Formattable   arguments[kArgCount] = { world }κ
    UnicodeString argnames[kArgCount]  = {"world"}κ
    FieldPosition fpos = 0κ
```

hellomsg.cpp

```
    const int kArgCount = 1κ
    Formattable   arguments[kArgCount] = { world }κ
    UnicodeString argnames[kArgCount]  = {"world"}κ
    FieldPosition fpos = 0κ
```

```
    MessageFormat msg_en("Hello, {world}",
        Locale("en"), status)κ
    UnicodeString result_enκ
    msg_en.format(argnames, arguments, kArgCount, result_en, status)κ
    ASSERT_OK(status)κ
    std::cout << "en: " << result_en << std::endlκ
```

```
en: Hello, World
```

hellomsg.cpp

```
    const int kArgCount = 1;
    Formattable   arguments[kArgCount] = { world };
    UnicodeString argnames[kArgCount]  = {"world"};
    FieldPosition fpos = 0;
```

```
    MessageFormat msg_es("¡Hola, {world}!",
        Locale("es"), status);
    UnicodeString result_es;
    msg_es.format(argnames, arguments, kArgCount, result_es, status);
    ASSERT_OK(status);
    std::cout << "es: " << result_es << std::endl;
```

```
es: ¡Hola, Mundo!
```

Java

Java(ICU4J)

ICU4J : Hello, Maven

```
        <dependency>
            <groupId>com.ibm.icu</groupId>
            <artifactId>icu4j</artifactId>
            <version>64.2</version>
        </dependency>
```

Hello.java

```
Locale locale = Locale.getDefault()⏎
String world = LocaleDisplayNames
        .getInstance(ULocale.forLocale(locale))
        .regionDisplayName("001")⏎
System.out.println("Hello, " + world + "\u2603")⏎
```

Hello, World☃

`Hello.java` (español)

```
Locale locale = Locale.forLanguageTag("es")⏎
String world = LocaleDisplayNames
        .getInstance(ULocale.forLocale(locale))
        .regionDisplayName("001")⏎
System.out.println("Hello, " + world + "\u2603")⏎
```

```
Hello, Mundo☃
```

Hello.java (español)

```
Locale locale = Locale.forLanguageTag("es")⏎
String world = LocaleDisplayNames
        .getInstance(ULocale.forLocale(locale))
        .regionDisplayName("001")⏎
System.out.println("Hello, " + world + "\u2603")⏎
```

Hello, Mundo☃

- use `java.util.Locale`

`Hello.java` (español)

```
Locale locale = Locale.forLanguageTag("es")⏎
String world = LocaleDisplayNames
        .getInstance(ULocale.forLocale(locale))
        .regionDisplayName("001")⏎
System.out.println("Hello, " + world + "\u2603")⏎
```

`Hello, Mundo☃`

- use `java.util.Locale`
- …except for some ICU4J APIs that still use ICU's `ULocale`

BadMessage.properties

```
population=The territory of {territory} has {population} persons.
```

BadMessage.java

```
        final Locale locale = Locale.getDefault()ĸ
        ResourceBundle rb = ResourceBundle.getBundle(BadMessage.class.getName())ĸ
        String popmsg = rb.getString("population")ĸ
        System.out.println("Message: " + popmsg)ĸ

        for(final PopulationData.TerritoryEntry entry :
           PopulationData.getTerritoryEntries(locale)) {
            MessageFormat m = new MessageFormat(popmsg, locale)ĸ
            Map msgArgs = new HashMap<String,Object>()ĸ
            msgArgs.put("territory", entry.territoryName())ĸ
            msgArgs.put("population", entry.population())ĸ
            System.out.println(m.format(msgArgs))ĸ
        }
```

BadMessage.java

```
        final Locale locale = Locale.getDefault()ĸ
        ResourceBundle rb = ResourceBundle.getBundle(BadMessage.class.getName())ĸ
        String popmsg = rb.getString("population")ĸ
        System.out.println("Message: " + popmsg)ĸ

        for(final PopulationData.TerritoryEntry entry :
           PopulationData.getTerritoryEntries(locale)) {
            MessageFormat m = new MessageFormat(popmsg, locale)ĸ
            Map msgArgs = new HashMap<String,Object>()ĸ
            msgArgs.put("territory", entry.territoryName())ĸ
            msgArgs.put("population", entry.population())ĸ
            System.out.println(m.format(msgArgs))ĸ
        }
```

```
Message: The territory of {territory} has {population} persons.
The territory of Afghanistan has 34,124,800 persons.
The territory of Albania has 3,047,990 persons.
The territory of Algeria has 40,969,400 persons.
```

- ok so far

BadMessage.java

```
        final Locale locale = Locale.getDefault()⏎
        ResourceBundle rb = ResourceBundle.getBundle(BadMessage.class.getName())⏎
        String popmsg = rb.getString("population")⏎
        System.out.println("Message: " + popmsg)⏎

        for(final PopulationData.TerritoryEntry entry :
           PopulationData.getTerritoryEntries(locale)) {
            MessageFormat m = new MessageFormat(popmsg, locale)⏎
            Map msgArgs = new HashMap<String,Object>()⏎
            msgArgs.put("territory", entry.territoryName())⏎
            msgArgs.put("population", entry.population())⏎
            System.out.println(m.format(msgArgs))⏎
        }
```

```
The territory of Bouvet Island has 1 persons.
The territory of Unknown Region has 0 persons.
```

- Not so OK!

BadMessage.java

```
        final Locale locale = Locale.getDefault()⏎
        ResourceBundle rb = ResourceBundle.getBundle(BadMessage.class.getName())⏎
        String popmsg = rb.getString("population")⏎
        System.out.println("Message: " + popmsg)⏎

        for(final PopulationData.TerritoryEntry entry :
           PopulationData.getTerritoryEntries(locale)) {
            MessageFormat m = new MessageFormat(popmsg, locale)⏎
            Map msgArgs = new HashMap<String,Object>()⏎
            msgArgs.put("territory", entry.territoryName())⏎
            msgArgs.put("population", entry.population())⏎
            System.out.println(m.format(msgArgs))⏎
        }
```

```
The territory of Bouvet Island has 1 persons.
The territory of Unknown Region has 0 persons.
```

- Not so OK!

CLDR Plurals

CLDR Plurals

- English: 0 dogs, 1 dog, 2 dogs, 3 dogs, 4 dogs

CLDR Plurals

- English: 0 dogs, 1 dog, 2 dogs, 3 dogs, 4 dogs
- Welsh: 0 cŵn,

CLDR Plurals

- English: 0 dogs, 1 dog, 2 dogs, 3 dogs, 4 dogs
- Welsh: 0 cŵn, 1 ci,

CLDR Plurals

- English: 0 dogs, 1 dog, 2 dogs, 3 dogs, 4 dogs
- Welsh: 0 cŵn, 1 ci, 2 gi,

CLDR Plurals

- English: 0 dogs, 1 dog, 2 dogs, 3 dogs, 4 dogs
- Welsh: 0 cŵn, 1 ci, 2 gi, 3 ci,

CLDR Plurals

- English: 0 dogs, 1 dog, 2 dogs, 3 dogs, 4 dogs
- Welsh: 0 cŵn, 1 ci, 2 gi, 3 ci, 4 ci

CLDR Plurals

- English: 0 dogs, 1 dog, 2 dogs, 3 dogs, 4 dogs
- Welsh: 0 cŵn, 1 ci, 2 gi, 3 ci, 4 ci

CLDR Plurals

Put ICU to Work!

GoodMessage.properties

```
population={population, plural,
  one{The territory of {territory} has # person}
  other{The territory of {territory} has # persons}}
```

GoodMessage.properties

```
population={population, plural,
  one{The territory of {territory} has # person}
  other{The territory of {territory} has # persons}}
```

- no code change

GoodMessage.properties

```
population={population, plural,
  one{The territory of {territory} has # person}
  other{The territory of {territory} has # persons}}
```

- no code change

```
The territory of United States has 323,996,000 persons
The territory of Unknown Region has 0 persons
The territory of Uruguay has 3,351,020 persons
The territory of Botswana has 2,209,210 persons
The territory of Bouvet Island has 1 person
The territory of Brazil has 205,824,000 persons
```

Units and Currencies

```
The room measures
{0, plural, one{1 meter} other{# meters}}
wide.
```

```
The room measures 0 meters wide.
The room measures 1 meter wide.
The room measures 0 meters wide.
```

But with ICU message strings, ICU can handle measurement units without having to enumerate all the plural forms yourself!

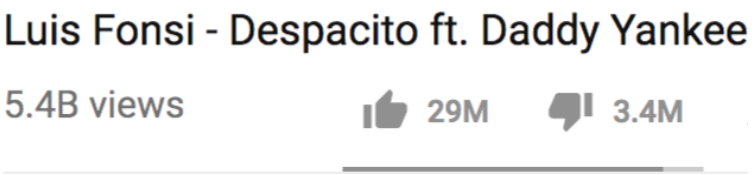
Use the "number" type instead of "plural" type and pass a number skeleton:

```
The room measures
{0, number, ::measure-unit/length-meter unit-width-full-name}
wide.
```

Also works for currencies.

Sample code: s88_units.cpp

Compact Notation



Programmatically:

```
std::cout
  << icu::number::NumberFormatter::with()
    .notation(icu::number::Notation::compactShort())
    .locale("en-us")
    .formatDouble(quantity, status)
    .toString(status)
  << std::endl
```

Via Message String:

```
{0, number, ::compact-short}
```

Sample code: s99_compact.cpp

Break Iteration

- Unicode standards + tailoring
- UAX#14 line breaking
- UAX#29 sentence, grapheme cluster, word

Break Iteration Sample

```
BreakIterator *wordIterator = BreakIterator::createWordInstance(locale, status)ĸ
breakIterator->setText(u"Hello World")ĸ
breakIterator->current()ĸ // 0
breakIterator->next()ĸ // 5
breakIterator->next()ĸ // 6
breakIterator->next()ĸ // 11
breakIterator->next()ĸ // -1 == DONE
```

Sample: `s23_brki.cpp`

Collators (Text Sorting)

- binary comparison inadequate
- order varies by language (Danish ‘aa…’ follows ‘z…’)
- need multiple-level collation

Uses:

- comparing
- sorting
- searching

Options:

- case sensitive?
- ignore punctuation?
- UPPERCASE first?
- which variant collator?
- which locale?
- custom tailorings?
- time vs. memory tradeoff?

CollateMessage.java

```
        Collator col = Collator.getInstance(locale)κ
        for(final PopulationData.TerritoryEntry entry :
         PopulationData.getTerritoryEntries(locale,
                new TreeSet<>((o1, o2)
                -> col.compare (o1.territoryName(), o2.territoryName()))))) {
                    …
```

CollateMessage.java

```
        Collator col = Collator.getInstance(locale)κ
        for(final PopulationData.TerritoryEntry entry :
         PopulationData.getTerritoryEntries(locale,
                new TreeSet<>((o1, o2)
                -> col.compare (o1.territoryName(), o2.territoryName()))))) {
                    …
```

- No Lambda function needed if `Set<String>`

Multilingual

Russian

```
The territory of Mxmzq.wur {-om has 26 200 persons in it.
The territory of Mxnmzuя has 3 011 410 persons in it.
```

Japanese

```
アイスランドには、315,281人います。
アイルランドには、4,775,980人います。
```

Spanish

```
En la región de “Afganistán” hay 31.108.100 personas.
En la región de “Albania” hay 3.011.410 personas.
En la región de “Angola” hay 18.565.300 personas.
```

API Stability

- `Internal`: Used by ICU implementation or Technology Preview.

API Stability

- `Internal`: Used by ICU implementation or Technology Preview.
- `Draft`: New API, reviewed and approved by ICU project team. The API might be still changed.

API Stability

- `Internal`: Used by ICU implementation or Technology Preview.
- `Draft`: New API, reviewed and approved by ICU project team. The API might be still changed.
- `Stable`: For public use, the API signature won't be changed in future releases.

API Stability

- `Internal`: Used by ICU implementation or Technology Preview.
- `Draft`: New API, reviewed and approved by ICU project team. The API might be still changed.
- `Stable`: For public use, the API signature won't be changed in future releases.
- `Deprecated`: Previously Stable, but no longer recommended. The API might be removed after a few releases.

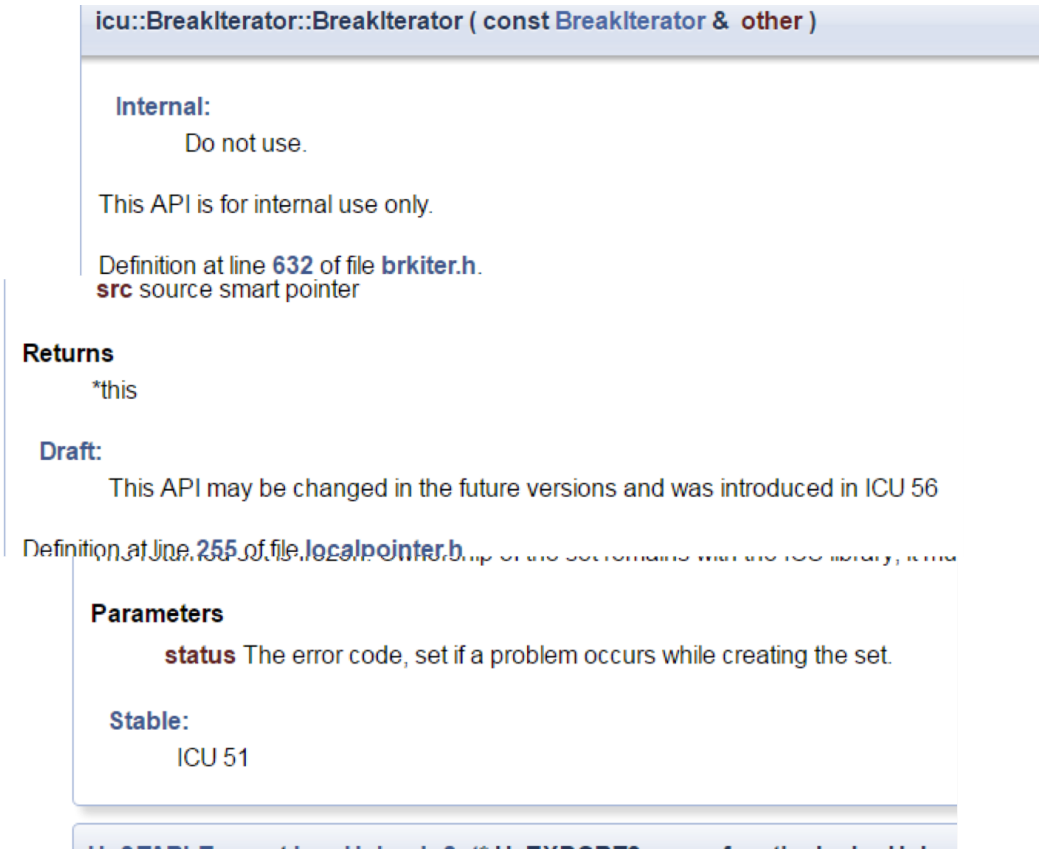
API Stability

- `Internal`: Used by ICU implementation or Technology Preview.
- `Draft`: New API, reviewed and approved by ICU project team. The API might be still changed.
- `Stable`: For public use, the API signature won't be changed in future releases.
- `Deprecated`: Previously Stable, but no longer recommended. The API might be removed after a few releases.

More details:

- userguide.icu-project.org/design

API Stability in docs



icu::BreakIterator::BreakIterator (const BreakIterator & other)

Internal:

Do not use.

This API is for internal use only.

Definition at line **632** of file **brkiter.h**.

src source smart pointer

Returns

*this

Draft:

This API may be changed in the future versions and was introduced in ICU 56

Definition at line 255 of file localpointer.h

Parameters

status The error code, set if a problem occurs while creating the set.

Stable:

ICU 51

Put ICU to Work!

Binary Stability

Source code compatible

- Consumer program should be compiled successfully without changes.
- Rare exceptions, documented in readme.

Serialization compatible (ICU4J)

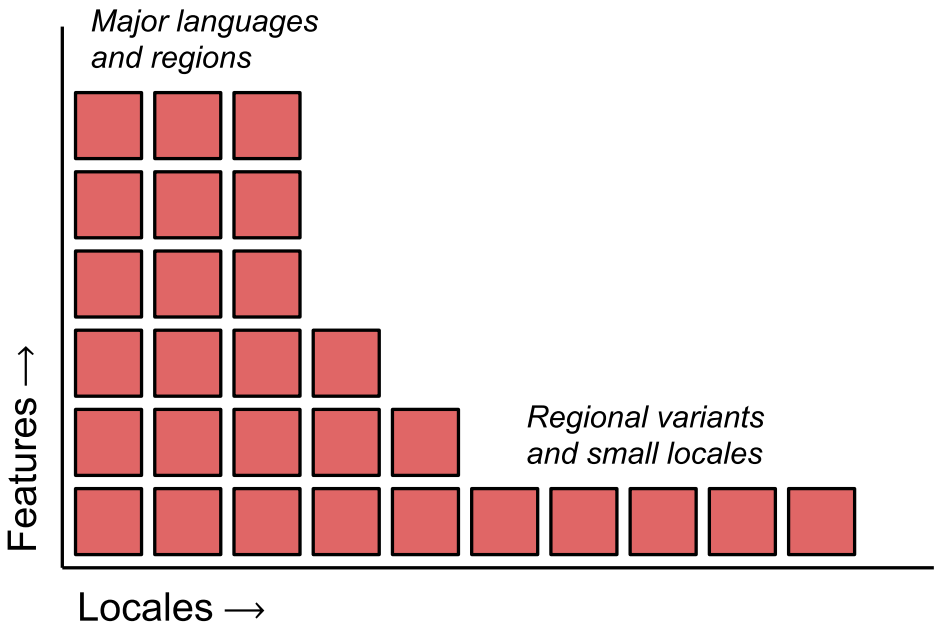
- Newer ICU version should be able to deserialize object data serialized by older ICU version.
- (see docs for limited exceptions)

ICU Data File

- aka, `icudt64l.dat`
- 20MB+ of data to support ICU's features

ICU Data File

- aka, `icudt64l.dat`
- 20MB+ of data to support ICU's features



Packaging

Packaging *"It's too big"*

Packaging *"It's too big"*

ICU 64 Data Build Tool

- Slice data by feature and locale
- Read the Docs, and attend Shane's session tomorrow

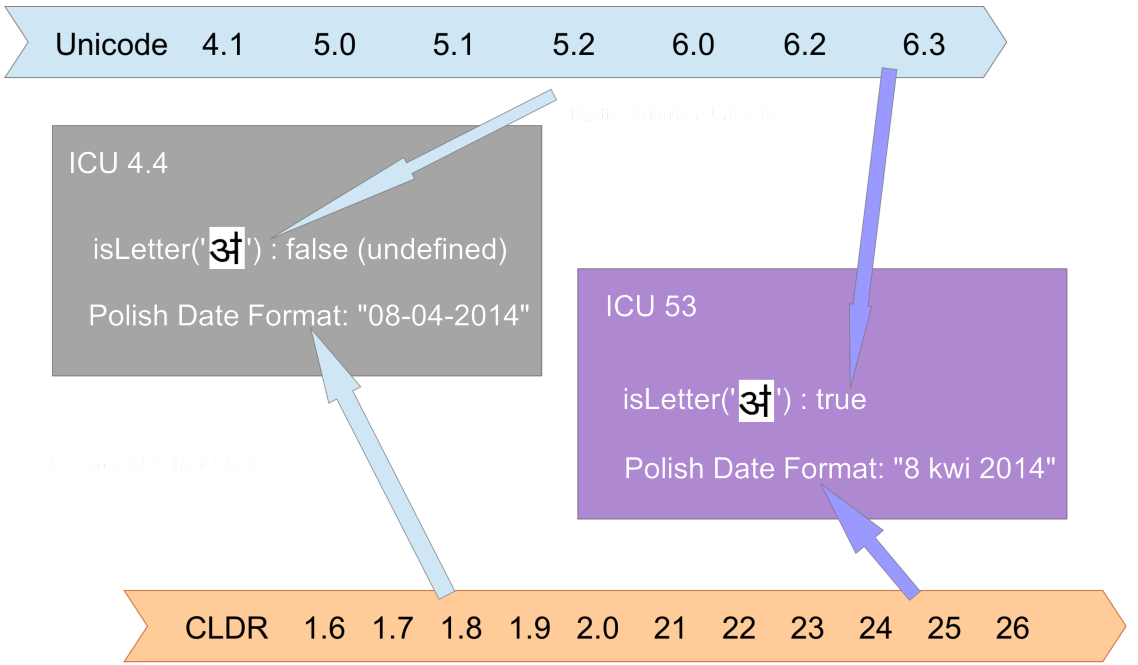
Other Customization

- Repackage ICU data http://userguide.icu-project.org/icudata
- Repackage ICU4C Code http://userguide.icu-project.org/packaging

  Example: `#define UCONFIG_NO_LEGACY_CONVERSION`

  - (Note: removes code but not data)

Data Changes



Put ICU to Work!

Data Stability

Unicode stability

Data Stability

Unicode stability

- character type, upper/lower case, normalization, text direction, sorting order…

Data Stability

Unicode stability

- character type, upper/lower case, normalization, text direction, sorting order…
- policy http://www.unicode.org/policies/stability_policy.html

Data Stability

Unicode stability

- character type, upper/lower case, normalization, text direction, sorting order…
- policy http://www.unicode.org/policies/stability_policy.html
- Unicode is still growing.

Data Stability

Unicode stability

- character type, upper/lower case, normalization, text direction, sorting order…
- policy http://www.unicode.org/policies/stability_policy.html
- Unicode is still growing.

Locale data

Data Stability

Unicode stability

- character type, upper/lower case, normalization, text direction, sorting order…
- policy http://www.unicode.org/policies/stability_policy.html
- Unicode is still growing.

Locale data

- cultural data can be updated based on community voting

Data Stability

Unicode stability

- character type, upper/lower case, normalization, text direction, sorting order…
- policy http://www.unicode.org/policies/stability_policy.html
- Unicode is still growing.

Locale data

- cultural data can be updated based on community voting
- cultural format results are not suited for serializing data, application protocols and storage

Stability Problems

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs(other side may parse/format differently)

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs(other side may parse/format differently)
- DON’T store localized data on disk

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs(other side may parse/format differently)
- DON’T store localized data on disk(later app version may parse/format differently)]

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs(other side may parse/format differently)
- DON’T store localized data on disk(later app version may parse/format differently)]
- DO send and store non-localized format
  - Binary: 0x12345678
  - “Neutral” - ISO 8601 - “`2019-04-08`”

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs(other side may parse/format differently)
- DON’T store localized data on disk(later app version may parse/format differently)]
- DO send and store non-localized format
  - Binary: 0x12345678
  - “Neutral” - ISO 8601 - “`2019-04-08`”
- REMEMBER अ may not be a letter

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs(other side may parse/format differently)
- DON’T store localized data on disk(later app version may parse/format differently)]
- DO send and store non-localized format
  - Binary: 0x12345678
  - “Neutral” - ISO 8601 - “`2019-04-08`”
- REMEMBER अ may not be a letter(isLetter()) in one Unicode version, but may later be defined.

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs(other side may parse/format differently)
- DON’T store localized data on disk(later app version may parse/format differently)]
- DO send and store non-localized format
  - Binary: 0x12345678
  - “Neutral” - ISO 8601 - “`2019-04-08`”
- REMEMBER अ may not be a letter(isLetter()) in one Unicode version, but may later be defined.Could cause difficulties if used to validate account names, …

Stability Problems

- “`08-04-2019`” may not parse as “`8 kwi 2019`”
- DON’T send localized data across the network between programs(other side may parse/format differently)
- DON’T store localized data on disk(later app version may parse/format differently)]
- DO send and store non-localized format
  - Binary: 0x12345678
  - “Neutral” - ISO 8601 - “`2019-04-08`”
- REMEMBER अ may not be a letter(isLetter()) in one Unicode version, but may later be defined.Could cause difficulties if used to validate account names, …
- DO Think carefully about where Unicode properties are used.

ICU4J vs JDK (0/2)

- ICU has functionality beyond JDK - See userguide.

ICU4J vs JDK (0/2)

- ICU has functionality beyond JDK - See userguide.
- Where there is overlap, in some cases JDK may be used instead of ICU.

ICU4J vs JDK (0/2)

- ICU has functionality beyond JDK - See userguide.
- Where there is overlap, in some cases JDK may be used instead of ICU. Example: `Locale` instead of ICU's ULocale

ICU4J vs JDK (1/2)

JDK class	ICU class	ICU Benefits	Suggestion
java.lang.**Character**	com.ibm.icu.lang.**UCharacter**	•Latest Unicode standard •More character properties support	JDK OK: ICU as-needed
java.math.**BigDecimal**	com.ibm.icu.math.**BigDecimal**	•For backward compatibility only	ICU not recommended in new code
java.text.**Bidi**	com.ibm.icu.text.**Bidi**	•Latest Unicode bidi algorithm (UAX#9)	JDK OK: ICU as-needed
java.text.**BreakIterator**	com.ibm.icu.text.**BreakIterator**	•Latest Unicode standard (UAX#29) •Dictionary based word break (Thai, Lao, Chinese/Japanese)	JDK OK: ICU as-needed
java.text.**Collator** java.text.**RuleBasedCollator**	com.ibm.icu.text.**Collator** com.ibm.icu.text.**RuleBasedCollator**	•Unicode collation algorithm (UTS#10) •Faster comparison	ICU recommended

Put ICU to Work!

ICU4J vs JDK (2/2)

JDK class	ICU class	ICU Benefits	Suggestion
java.text.DateFormat java.text.SimpleDateFormat	com.ibm.icu.text.DateFormat com.ibm.icu.text.SimpleDateFormat	• Abstract (skeleton) pattern (e.g. year-month only format) • Patterns for additional calendar types • More field format types (e.g. narrow weekday, standalone month) • Capitalization control • Slower service object creation, format & parse than JDK	JDK OK, ICU as-needed
java.text.MessageFormat	com.ibm.icu.text.MessageFormat	• Plural formatting • Gender formatting (social applications) • Named arguments (“{filename}” vs “{4}”) • Auto apostrophe mode	JDK OK, ICU as-needed
java.text.NumberFormat java.text.DecimalFormat	com.ibm.icu.text.NumberFormat com.ibm.icu.text.DecimalFormat + RuleBasedNumberFormat	• More styles (e.g. scientific, currency spell out) • Parse currency • Algorithmic numbering systems • Slower service object creation, format & parse than JDK	JDK OK, ICU as-needed

Put ICU to Work!

Action for You: Join our mailing lists!

http://site.icu-project.org/contacts

Action for You: Join our mailing lists!

http://site.icu-project.org/contacts

Sample Code: bit.ly/iuc43-icu-samples

Presenter: Steven Loomis

- Social: @srl295
- Web site: git.io/srl295
- Email: `srloomis@us.ibm.com`

Presenter: Shane Carr

- Social: @sffc or @_sffc
- Web site: https://sffc.xyz
- Email: `sffc@google.com` / `shane@unicode.org`

Have a nice day!