IBM

Using ICU Workshop

Steven R. Loomis <srloomis@us.ibm.com>
Software Engineer,
IBM San José Globalization Center of Competency

35th Internationalization and Unicode Conference
October 2011 • San José, California, USA

Agenda

- What is ICU?
- Getting & setting up ICU
- Testing it for yourself
- Using the conversion engine
- Using the break iterator engine
- Using resource bundles
- Using the collation engine
- Using message formats

What is ICU?

- International Components for Unicode
- Globalization / Unicode / Locales
- Mature, widely used set of C/C++ and Java libraries
  - Basis for Java 1.1 internationalization, but goes far beyond Java 1.1
- Very portable – identical results on all platforms / programming languages
  - C/C++: 30+ platforms/compilers (ICU4C)
  - Java: IBM JDK, Oracle (Sun) JDK, OpenJDK
- Full threading model
- Conformant with the latest version of Unicode
- Data updated with the latest CLDR
- Customizable
- Modular
- Non-viral Open Source: MIT/X License

ICU4C

Getting ICU4C

Recommended: Download the latest stable release

- **`icu-project.org/download`**
- Binary package: pre-built for your OS/compiler
- Source download: other platforms, modifying build options
- API Docs: for off-line reading

Bleeding edge development:

- Download from Subversion repository:
  icu-project.org/repository/

Setting up ICU4C

Download & unpack binaries

If you need to build from source, read ICU's readme.html (in source package)

- Windows:
  - MSVC .Net 2010 project files
  - Cygwin or MinGW (MSVC, gcc, Intel and so on)
    - Follow Unix readme.html instructions
    - Some advanced options may work differently
- Unix & Unix like operating systems:
  - **runConfigureICU** … (or just **configure** …)
  - **make** install
  - **make** check

Commonly Used Configure Options

`--prefix=`*`directory`*

- Set to where you want to install ICU

`--with-library-bits=64`

- Build 64-bit libraries instead of 32-bit, or vice versa

`--with-library-suffix=`*`name`*

- Allows you to customize the library name
- Highly recommended when not using the default configure options

`--enable-static`

- Build static libraries
- Make sure you get your very own ICU
- Minimize footprint when using a small amount of ICU
- Beware large data, stale dependencies
- If you're building on Windows, read the readme.html

Commonly Used Configure Options (Part II)

`--with-data-packaging=`*`type`*

- Specify the type of data that ICU's large data library should be packaged
- Specify **files**, **archive** or **library**

`--disable-renaming`

- Disable the ICU version renaming
  ucnv_open() → ucnv_open_48()
- Not normally recommended

`--enable-debug`

- Enable building debuggable versions of ICU
- Use with runConfigureICU before you specify the platform target

`--disable-release`

- Disable building optimized versions of ICU
- Use with runConfigureICU before you specify the platform target

Testing ICU4C

Windows

- C:\ICU**source\allinone\icucheck.bat** x86 Debug

(**x86** or **x64**, **Debug** or **Release**)

Unixes

- **gmake check**

Testing for Yourself: code

```
#include <stdio.h>
#include "unicode/uclean.h"

void main() {
    UErrorCode status = U_ZERO_ERROR;
    u_init(&status);
    printf("This is ICU %s!\n", U_ICU_VERSION);
    if (U_SUCCESS(status)) {
        printf("everything is OK\n");
    } else {
        printf("error %s initializing.\n", u_errorName(status));
    }
}
```

ICU for C First Look

```
#include "unicode/…"
```

- All ICU headers are in unicode/

```
UErrorCode status = U_ZERO_ERROR;
```

- Error code is a Fill-in – **must** be initialized.

```
u_init(&status);
```

- Returns success if ICU data loads OK.

```
If ( U_SUCCESS(status) ) …
```

- TRUE if no error.

Testing for Yourself: Windows

Project Properties

- C/C++»General»Additional Include Directories:
  - `icu\`**`include`**
- Linker»Input»Additional Dependencies:
  - **`icuuc.lib;icuin.lib;icuio.lib`**

Add `icu\bin` to your `PATH`

Testing for Yourself: UNIX

```
CPPFLAGS+= -I/path/to/icu/include
LDFLAGS+=-L/path/to/icu/lib -licuuc -licui18n -licuio
```

Set `LD_LIBRARY_PATH` or `DYLD_LIBRARY_PATH`, etc to `/path/to/icu/lib`

Add `/path/to/icu/bin` to your `PATH` for tools

HelloWorld (ICU Style)

```
#include <unicode/ustdio.h>
int main(int argc, const char *argv[]) {
  UFILE *out;
  UErrorCode status = U_ZERO_ERROR;
  out = u_finit(stdout, NULL, NULL);
  UChar world[256];
  uloc_getDisplayCountry("und_001", NULL, world,
                256, &status);
  if(U_FAILURE(status)) { puts("Fail!"); return 1; }
  u_fprintf(out, "Hello, %S!\n", world);
  u_fclose(out);
  return 0;
}
```

»"Hello, World!"

%S: world[ ] is UTF-16 (Unicode)

und_001 = "Language=Unknown, **Region=World**"

Hello*Welt*

```
#include <unicode/ustdio.h>
int main(int argc, const char *argv[]) {
  UFILE *out;
  UErrorCode status = U_ZERO_ERROR;
  const char *locale = "de"; // Get the user's locale from somewhere
  out = u_finit(stdout, locale, NULL);
  UChar world[256];
  uloc_getDisplayCountry("und_001", locale, world,
                256, &status);
  if(U_FAILURE(status)) { puts("Fail!"); return 1; }
  u_fprintf(out, "%s: Hello, %S!\n", locale, world);
  u_fclose(out);
  return 0;
}
```

»"de: Hello, Welt!"

Changing "de" to "zh-Hans" gives:

»zh-Hans: Hello, 世界！

Conversion Engine - Opening

ICU Service objects use an open/close model.

Here is a simplified example with a converter:

```
UErrorCode status = U_ZERO_ERROR;
UConverter *cnv = ucnv_open(encoding, &status);
if(U_FAILURE(status)) {
        /* process the error situation, die gracefully */
}
… /* Use the converter */
/* then call close */
ucnv_close(cnv);
```

- Can't share service objects across threads.
- Many services have a clone() facility available.

What Converters are Available?

`ucnv_countAvailable()` – get the number of available converters

`ucnv_getAvailable` – get the name of a particular converter

Many frameworks allow this type of examination.

Converting Text Chunk by Chunk

Quick example of using the converter API

```
char buffer[DEFAULT_BUFFER_SIZE];
char *bufP = buffer;
int32_t len = ucnv_fromUChars(cnv, bufP, DEFAULT_BUFFER_SIZE,
                              source, sourceLen, &status);
if(U_FAILURE(status)) {
        if(status == U_BUFFER_OVERFLOW_ERROR) {
                status = U_ZERO_ERROR;
                bufP = (UChar *)malloc((len + 1) * sizeof(char));
                len = ucnv_fromUChars(cnv, bufP, DEFAULT_BUFFER_SIZE,
                                      source, sourceLen, &status);
        } else {
                /* other error, die gracefully */
        }
}
/* do interesting stuff with the converted text */
```

Converting Text Character by Character

```
UChar32 result;
char *source = start;
char *sourceLimit = start + len;
while(source < sourceLimit) {
        result = ucnv_getNextUChar(cnv, &source, sourceLimit, &status);
        if(U_FAILURE(status)) {
                /* die gracefully */
        }
        /* do interesting stuff with the converted text */
}
```

Works only from code page to Unicode

Less efficient than converting a whole buffer

Doesn't require managing a target buffer

Converting Text Piece by Piece From a File

```
while((!feof(f)) && ((count=fread(inBuf, 1, BUFFER_SIZE , f)) > 0) ) {
    source = inBuf;
    sourceLimit = inBuf + count;
    do {
        target = uBuf;
        targetLimit = uBuf + uBufSize;
        ucnv_toUnicode(conv, &target, targetLimit,
                    &source, sourceLimit, NULL,
                    feof(f)?TRUE:FALSE, /* pass 'flush' when eof */
                    /* is true (when no more data will come) */
                    &status);
        if(status == U_BUFFER_OVERFLOW_ERROR) {
            // simply ran out of space - we'll reset the
            // target ptr the next time through the loop.
            status = U_ZERO_ERROR;
        } else {
            //  Check other errors here and act appropriately
        }
        text.append(uBuf, target-uBuf);
        count += target-uBuf;
    } while (source < sourceLimit); // while simply out of space
}
```

Clean up!

Whatever is opened, needs to be closed

Converters use **`ucnv_close`**`()`

Other C APIs that have an open function also have a close function

Allocated C++ objects require delete

Can clean up all of ICU with `u_cleanup(…)`

(opposite of `u_init()`)

Break Iteration - Introduction

Four types of boundaries:

- Character, word, line, sentence

Points to a boundary between two characters

Index of character following the boundary

Use `current()` to get the boundary

Use `first()` to set iterator to start of text

Use `last()` to set iterator to end of text

Break Iteration - Navigation

Use `next()` to move to next boundary

Use `previous()` to move to previous boundary

Returns `BreakIterator::DONE` if can't move boundary

Break Iteration - Checking a Position

Use `isBoundary()` to see if position is boundary

Use `preceding()` to find boundary at or before

Use `following()` to find boundary at or after

Break Iteration - Opening

Use the factory methods:

```
Locale     locale("th"); // locale to use for break iterators
UErrorCode status = U_ZERO_ERROR;

BreakIterator *characterIterator =
               BreakIterator::createCharacterInstance(locale, status);

BreakIterator *wordIterator =
               BreakIterator::createWordInstance(locale, status);

BreakIterator *lineIterator =
               BreakIterator::createLineInstance(locale, status);

BreakIterator *sentenceIterator =
               BreakIterator::createSentenceInstance(locale, status);
```

Don't forget to check the status!

Set the text

We need to tell the iterator what text to use:

```
UnicodeString text;

readFile(file, text);
wordIterator->setText(text);
```

Reuse iterators by calling `setText()` again.

Break Iteration - Counting Words in a File

```
int32_t countWords(BreakIterator *wordIterator, UnicodeString &text)
{
    U_ERROR_CODE status = U_ZERO_ERROR;
    UnicodeString word;
    UnicodeSet letters(UnicodeString("[:letter:]"), status);

    int32_t wordCount = 0;
    int32_t start = wordIterator->first();

    for(int32_t end = wordIterator->next();
        end != BreakIterator::DONE;
        start = end, end = wordIterator->next())
    {
        text->extractBetween(start, end, word);

        if(letters.containsSome(word)) {
            wordCount += 1;
        }
    }

    return wordCount;
}
```

Break Iteration - Breaking Lines

```
int32_t previousBreak(BreakIterator *breakIterator, UnicodeString &text,
                      int32_t location)
{
    int32_t len = text.length();

    while(location < len) {
        UChar c = text[location];

        if(!u_isWhitespace(c) && !u_iscntrl(c)) {
            break;
        }

        location += 1;
    }

    return breakIterator->previous(location + 1);
}
```

Break Iteration - Cleaning Up

Use `delete` to delete the iterators

```
delete characterIterator;
delete wordIterator;
delete lineIterator;
delete sentenceIterator;
```

Using Resource Bundles

Provides a way to separate translatable text from code

Provides an easy way to update and add localizations to your product

Your application must be internationalized before it can be localized

It's best to encode the files as UTF-8 with a BOM

Bundles can be converted to and from XLIFF format for translation interchange

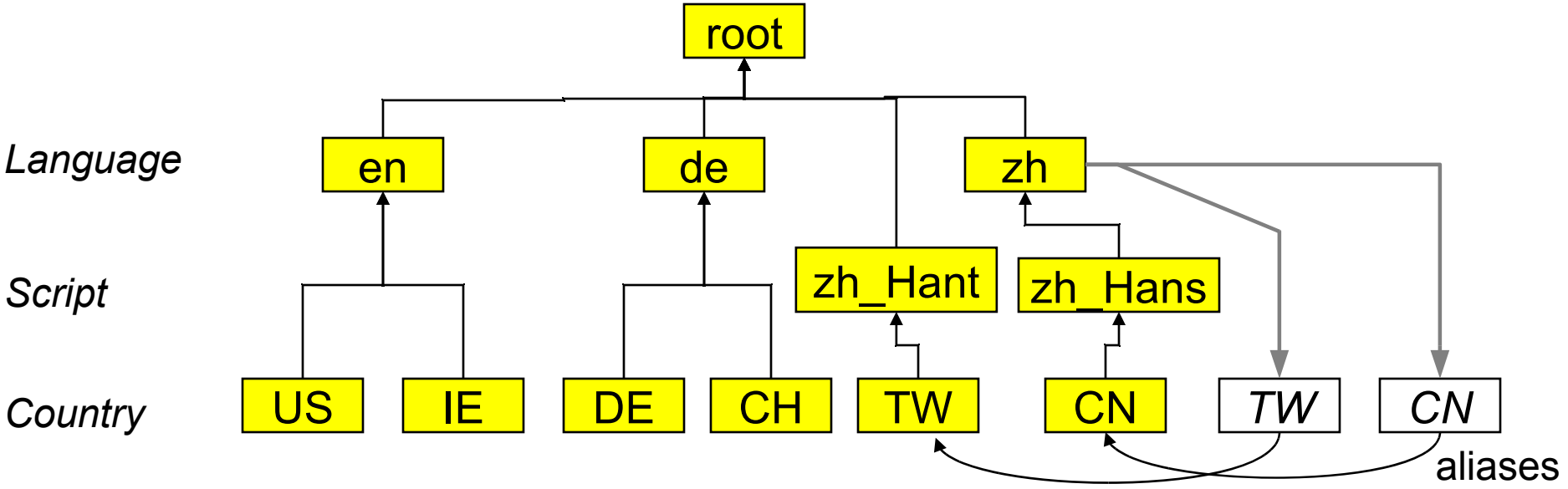
Resource Bundle Overview

Locale Based Services

- Locale is an **identifier**, not a container

Resource inheritance: shared resources

zh_Hant (Traditional Chinese)
does **not** inherit from zh (Simplified Chinese)



Creating a Resource Bundle

Create files like the following:

root.txt:

```
root {
    Aunt { "My Aunt" }
    table { "on the table" }
    pen { "pen" }
    personPlaceThing { "{0}''s {2} is {1}." }
}
```

es.txt:

```
es {
    Aunt { "mi tía" }
    table { "en la tabla" }
    pen { "La pluma" }
    personPlaceThing { "{2} de {0} está {1}." }
}
```

Building a Resource Bundle

Create a file called pkgdatain.txt with these contents

pkgdatain.txt:

```
myapp/es.res
myapp/root.res
```

Execute these commands where the files are located

```
$ mkdir myapp
$ genrb -d myapp root.txt
$ genrb -d myapp es.txt
$ pkgdata -m archive -p myapp pkgdatain.txt
```

This results in a **`myapp.dat`** archive file being created

Accessing a Resource Bundle

Here is a C++ and then a C example:

```
UErrorCode status = U_ZERO_ERROR;
ResourceBundle resourceBundle("myapp", Locale::getDefault(), status);
if(U_FAILURE(status)) {
    printf("Can't open resource bundle. Error is %s\n", u_errorName(status));
    return;
}

// thing will be “pen” or “La pluma”
UnicodeString thing = resourceBundle.getStringEx("pen", status);
```

```
  UErrorCode status = U_ZERO_ERROR;
  int32_t length;
  UResourceBundle *resourceBundle = NULL;
  const UChar *thing;
  resourceBundle = ures_open("myapp", NULL, &status);
  if(U_FAILURE(status)) {
    printf("Can't open resource bundle. Error is %s\n", u_errorName(status));
    return;
  }

  thing = ures_getStringByKey(resourceBundle, "pen", &length, &status);
  printString(thing);   /* thing will be “pen” or “La pluma” */
  ures_close(resourceBundle);  /* 'thing' is no longer valid */
```

Collation Engine

Used for comparing strings in a culturally sensitive way

Instantiation:

C:

```
UErrorCode status = U_ZERO_ERROR;
UCollator *coll = ucol_open("en_US", &status);
if(U_SUCCESS(status)) {
     /* do useful things with a collator */
     ucol_close(coll);
}
```

C++:

```
UErrorCode status = U_ZERO_ERROR;
Collator *coll = Collator::createInstance(Locale("en", "US"), status);
if(U_SUCCESS(status)) {
     // do useful things with a collator
     delete coll;
}
```

String Comparison

Works fast

You get the result of the comparison as soon as it is known

Use when you don't need to compare the same strings

```
C: UCollationResult ucol_strcoll (const UCollator *coll,
                const UChar *source, int32_t sourceLength,
                const UChar *target, int32_t targetLength)

C++: EComparisonResult Collator::compare (
       const UnicodeString &source, const UnicodeString &target) const
```

Using Comparison (C)

```
UChar *s [] = { /* list of Unicode strings */ };
uint32_t listSize = sizeof(s)/sizeof(s[0]);
UErrorCode status = U_ZERO_ERROR;
UCollator *coll = ucol_open("en_US", &status);
uint32_t i, j;
if(U_SUCCESS(status)) {
  for(i=listSize-1; i>=1; i--) {
    for(j=0; j<i; j++) {
      if(ucol_strcoll(s[j], -1, s[j+1], -1) == UCOL_LESS) {
        swap(s[j], s[j+1]);
      }
    }
  }
  ucol_close(coll);
}
```

Using Comparison (C++)

```
UnicodeString s[] = { /* list of Unicode strings */ };
uint32_t listSize = sizeof(s)/sizeof(s[0]);
UErrorCode status = U_ZERO_ERROR;
Collator *coll = Collator::createInstance(Locale("en", "US"), status);
uint32_t i, j;
if(U_SUCCESS(status)) {
  for(i=listSize-1; i>=1; i--) {
    for(j=0; j<i; j++) {
      if(coll->compare(s[j], s[j+1]) == UCOL_LESS) {
        swap(s[j], s[j+1]);
      }
    }
  }
  delete coll;
}
```

Sort Keys

- Array of bytes representing a string
- Used when multiple comparisons are required
- Indexes in databases
- Compare only sort keys generated by the same type of a collator
- You need a receiving buffer
- Use preflighting to estimate memory needed

Message Format - Introduction

Assembles a user message from parts both fixed (static), and dynamic.

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

- English: My Aunt's pen is on the table.
- Spanish: La pluma de mi tía está en la tabla.

Pattern string (from Resource Bundle) defines how to assemble the parts:

- English: `{0}''s {2} is on the {1}.`
- Spanish: `{2} de {0} está en la {1}.`

Message Format - Example

```
UErrorCode status = U_ZERO_ERROR;
ResourceBundle resourceBundle("myapp", Locale::getDefault(), status);

if(U_FAILURE(status)) {
    printf("Can't open resource bundle. Error is %s\n", u_errorName(status));
    return;
}

Formattable arguments[3];
arguments[0].setString(resourceBundle.getStringEx(”Aunt", status)); // “My Aunt”
arguments[1].setString(resourceBundle.getStringEx(”table", status)); // “on the table”
arguments[2].setString(resourceBundle.getStringEx("pen", status)); //  “pen”

UnicodeString pattern = resourceBundle.getStringEx("personPlaceThing", status);
UnicodeString result;

MessageFormat::format(pattern, arguments, 3, result, status);
```

Message Format - Different Data Types

We can also format other data types, like dates

We do this by adding a format type:

```
UnicodeString pattern = "On {0, date} at {0, time} there was {1}.";
Calendar *c = icu::Calendar::createInstance(status);
Formattable args[] = {
  c->getTime(status), // 0
  "a power failure"   // 1
};
UnicodeString result;
MessageFormat::format(pattern, args, 2, result, status);
```

Result will contain:

```
On Oct 15, 2007 at 10:15:08 AM there was a power failure.
```

Message Format - Format Styles

Add a format style:

```
UnicodeString pattern =
    "On {0, date, full} at {0, time, long} there was {1}.";
Calendar *c = icu::Calendar::createInstance(status);
Formattable args[] = {
  c->getTime(status), // 0
  "a power failure"   // 1
};
UnicodeString result;
MessageFormat::format(pattern, args, 2, result, status);
```

Result will contain:

```
On Monday, October 15, 2007 at 10:15:08 AM PDT there was a power
failure.
```

Message Format – Named Arguments

```
UnicodeString pattern = "On {when, date, full} at {when, time,
long} there was {what}.";
Calendar *c = icu::Calendar::createInstance(status);
Formattable args[] = {
  c->getTime(status), // when
  "a power failure"   // what
};
UnicodeString names[] = {
      "when",
       "what"
};

UnicodeString result;
MessageFormat fmt(pattern, status);
fmt.format(names, pattern, args, 2, result, status);
```

Result is the same:

```
On Monday, October 15, 2007 at 10:15:08 AM PDT there was a power
failure.
```

Message Format - Format Style

Format Type	Format Style	Sample Output
number	*(none)*	123,456.789
	integer	123,457
	currency	$123,456.79
	percent	12%
date	*(none)*	Jul 16, 2011
	short	7/16/11
	medium	Jul 16, 2004
	long	July 16, 2011
	full	Saturday, July 16, 2011
time	*(none)*	2:15:08 PM
	short	2:15 PM
	medium	2:14:08 PM
	long	2:15:08 PM PDT
	full	2:15:08 PM PDT

Message Format - Counting Files

Plural pattern to display number of files:

```
There {1, plural, one{is one file} other{are # files}} in {0}.
```

Code to use the pattern:

```
UnicodeString pattern = resourceBundle.getStringEx("fileCount");
UnicodeString directoryName = ...;
int fileCount = ...;
Formattable args[] = {
  directoryName, // 0
  fileCount      // 1
};
UnicodeString result;
MessageFormat::format(pattern, args, 2, result, status);
```

This could be used to output messages like:

```
There are 1,234 files in myDirectory.
There is one file in myDirectory.
There are 0 files in myDirectory.
```

Message Format - Other Details

Format style can be a pattern string

- Format type `number`: use `DecimalFormat` pattern (e.g. #,#00.00)
- Format type `date`, `time`: use `SimpleDateFormat` pattern (e.g. MM/yy)

Quoting in patterns

- Enclose special characters in single quotes
- Use two consecutive single quotes to represent one

```
The '{' character, the '#' character and the '' character.
```

ICU4J

Getting ICU4J

- **Recommended:** Download a stable release
  - **Easiest** – pre-compiled `.jar` from

    icu-project.org/download
  - Use the latest stable version if possible
  - For source, download the source `.jar`
- Bleeding edge development:
  - Download from Subversion

    icu-project.org/repository

Setting up ICU4J

- JDK/JRE version 5.0 or later (J2SE 5.0)
- Try the test code:

```
import com.ibm.icu.util.ULocale;
import com.ibm.icu.util.VersionInfo;
public class TestICU {
    public static void main(String[] args) {
        System.out.println("Hello, "
       + new Ulocale("und_001")
          .getDisplayCountry(ULocale.GERMAN)
       + "! ICU " + VersionInfo.ICU_VERSION);
    }
}
```

- Add ICU's jar to classpath and run TestICU:
  "Hello, Welt! ICU #.#.#…"
- Can also test ICU as jar: "**java -jar icu4j.jar**"

Building ICU4J

- Use Apache Ant to build
  - "**`ant all`**" (to build all)
  - "**`ant jar`**" (to just build icu4j.jar)
  - "**`ant check`**" (to run tests)
  - Other targets, see the Readme and homepage
- Can also build from Eclipse (preferred)

Collation Engine

- Used for comparing strings
- Instantiation:

```
ULocale locale = new ULocale("fr");
Collator coll = Collator.getInstance(locale);
// do useful things with the collator
```

- Package: **com.ibm.icu.text.Collator**

String Comparison

- Works fast
- You get the result as soon as it is ready
- Use when you don't need to compare same strings multiple times

- is-a **`Comparator`**

```
int coll.compare(String source, String target);
```

```
new TreeSet<String>(coll)
```

Sort Keys

- Used when multiple comparisons are required
- Indexes in data bases
- Only sort keys generated by the same type of collator should be compared
- ICU4J has two classes
  - **`CollationKey`**
  - **`RawCollationKey`**

CollationKey class

- JDK API compatible
- Saves the original string
- Compare keys with **`compareTo`**`()` method
- Get the bytes with **`toByteArray`**`()` method

RawCollationKey class

- Does not store the original string
- Get with the **`getRawCollationKey`**`()` method
- Mutable class, can be reused
- Simple and lightweight

Collation Key Example

```
Collator coll…

String strs[] = { “bad”, “baz”, “bat” };

TreeMap<RawCollationKey, String> rawMap =
        new TreeMap<RawCollationKey, String>();

for(String s : strs ) {

        rawMap.put(coll.getRawCollationKey(s,null));

}

//  'rawMap' is in collated order, because the keys are comparable.


RawCollationKey bazKey = coll.getRawCollationKey(“baz”,null);

String out = rawMap.get(bazKey); // == “baz”
```

Collation Key Table

Key	Value
bad	2F 2D 33 01 07 01 07 00
baz	2F 2D 5F 01 07 01 07 00
Bat	2F 2D 53 01 07 01 8F 06 00
BAD	2F 2D 33 01 07 01 8F 8F 8F 00
bat	2F 2D 53 01 07 01 07 00

BazKey = "2F 2D 5F 01 07 01 07 00", matches "baz"

Message Format - Example

```
UResourceBundle resourceBundle
    = UResourceBundle.getBundleInstance("myapp", ULocale.getDefault());

String person = resourceBundle.getString("Aunt"); // e.g. "My Aunt"
String place = resourceBundle.getString("table"); // e.g. "on the table"
String thing = resourceBundle.getString("pen"); // e.g. "pen"
Object arguments[] = {person, place, thing};
String pattern = resourceBundle.getString("personPlaceThing");
MessageFormat msgFmt = new MessageFormat(pattern);
String message = msgFmt.format(arguments);

System.out.println(message);
```

ICU4J: Message Format - Different Data Types

- Uses normal Java types such as Date and String.

```
String pattern = "On {0, date} at {0, time} there was {1}.";
MessageFormat fmt = new MessageFormat(pattern);
Object args[] = {new Date(System.currentTimeMillis()), // 0
                   "a power failure"                    // 1
                 };

System.out.println(fmt.format(args));
```

- **This will output:**

```
On Jul 16, 2011 at 2:15:08 PM there was a power failure.
```

Thank You / Useful Links

IBM

Homepage: icu-project.org

API documents: icu-project.org/apiref

User guide: icu-project.org/userguide

Latest Version/Sample Code:
icu-project.org/docs

IBM Globalization:
ibm.com/software/globalization