International Components for Unicode for Java (ICU4J)

Read Me for ICU4J 56 Release Candidate

(Last Update: 2015-September-23)

Note: This is a release candidate of ICU4J 56. The contents of this document may not reflect the recent changes done for ICU 56 development. This release candidate is intended for those wishing to verify ICU 56 integration before final release. It is not recommended for production use.

For the most recent release, see the ICU4J download site.

Contents

Introduction to ICU4J

The International Components for Unicode (ICU) library provides robust and full-featured Unicode services on a wide variety of platforms. ICU supports the most current version of the Unicode standard, including support for supplementary characters (needed for GB 18030 repertoire support).

Java provides a strong foundation for global programs, and IBM and the ICU team played a key role in providing globalization technology to Java. But because of its long release schedule, Java cannot always keep up with evolving standards. The ICU team continues to extend Java's Unicode and internationalization support, focusing on improving performance, keeping current with the Unicode standard, and providing richer APIs, while remaining as compatible as possible with the original Java text and internationalization API design.

ICU4J is an add-on to the regular JRE that provides:

- **Collation** – rule-based, up-to-date Unicode Collation Algorithm (UCA) sorting order
  For fast multilingual string comparison; faster and more complete than the J2SE implementation
- **Charset Detection** – Recognition of various single and multibyte charsets
  Useful for recognizing untagged text data
- **UnicodeSet** – standard set operations optimized for sets of Unicode characters
  UnicodeSets can be built from string patterns using any Unicode properties.
- **Transforms** – a flexible mechanism for Unicode text conversions
  Including Full/Halfwidth conversions, Normalization, Case conversions, Hex conversions, and transliterations between scripts (50+ pairs)
- **Unicode Normalization** – NFC, NFD, NFKD, NFKC
  For canonical text representations, needed for XML and the net

- **International Calendars** – Arabic, Buddhist, Chinese, Hebrew, Japanese, Ethiopic, Islamic, Coptic and other calendars
  Required for correct presentation of dates in certain countries
- **Number Format Enhancements** – Scientific Notation, Spelled-out, etc.
  Enhancements to the normal Java number formatting. The spell-out format is used for checks and similar documents
- **Enhanced Word-Break Detection** – Rule-based, supports Thai
  Required for correct support of Thai
- **Unicode Text Compression** – Standard compression of Unicode text
  Suitable for large numbers of small fields, where LZW and similar schemes do not apply
- **Charset Conversion** – Conversion to and from different charsets.
  Plugs into Java CharsetProvider Service Provider Interface (SPI)

Note: We continue to provide assistance to Java, and in some cases, ICU4J support has been rolled into a later release of Java. For example, BCP47 language tag support including Unicode locale extensions is now in Java 7. However, the most current and complete version is always found in ICU4J.

Changes In This Release

See the ICU 56 download page about new features in this release. The list of API changes since the previous ICU4J release is available here.

License Information

The ICU projects (ICU4C and ICU4J) use the X license. The X license is **suitable for commercial use** and is a recommended free software license that is compatible with the GNU GPL license. This became effective with release 1.8.1 of ICU4C and release 1.3.1 of ICU4J in mid-2001. All new ICU releases will adopt the X license; previous ICU releases continue to utilize the IPL (IBM Public License). Users of previous releases of ICU who want to adopt new ICU releases will need to accept the terms and conditions of the X license.

The main effect of the change is to provide GPL compatibility. The X license is listed as GPL compatible, see the GNU page at http://www.gnu.org/licenses/license-list.html#GPLCompatibleLicenses. This means that GPL projects can now use ICU code, it does **not** mean that projects using ICU become subject to GPL.

The IBM version contains the essential text of the license, omitting the X-specific trademarks and copyright notices. The full copy of ICU's license is included in the download package.

Platform Dependencies

ICU4J 56 depends on J2SE 5.0 functionality. Therefore, ICU4J only runs on JRE version 5.0 or later. The table below shows the operating systems and JRE/VM versions currently used by the ICU development team to test ICU4J.

Operating System	JRE 8		JRE 7		JRE 6		JRE 5	
	32bit	64bit	32bit	64bit	32bit	64bit	32bit	64bit
AIX 6.1	-	-	-	Regularly tested	-	Regularly tested	-	Regularly tested
AIX 7.1	-	-	-	***Reference platform***	-	Regularly tested	-	Regularly tested
HP-UX 11 (IA64)	-	-	-	Regularly tested	-	Regularly tested	-	Regularly tested
Mac OS X 10.6	-	-	-	-	-	Regularly tested	-	-
Redhat Enterprise Linux 6 (x86)	-	-	Regularly tested	-	Regularly tested	-	Regularly tested	-

Redhat Enterprise Linux 6 (x86_64)	-	Regularly tested	-	***Reference platform***	-	Regularly tested	-	Regularly tested
Solaris 10 (SPARC)	-	-	Regularly tested	Regularly tested	Regularly tested	Regularly tested	Regularly tested	Regularly tested
Solaris 11 (SPARC)	-	-	-	Regularly tested	-	Regularly tested	-	Regularly tested
Windows XP	-	-	Regularly tested	-	Regularly tested	-	Regularly tested	-
Windows Vista	-	-	Regularly tested	-	Regularly tested	-	Regularly tested	-
Windows 7	-	Regularly tested	***Reference platform***	-	Regularly tested	-	Regularly tested	-
Windows 2008 Server	-	Regularly tested	-	Regularly tested	-	Regularly tested	-	Regularly tested

How to Download ICU4J

There are two ways to download the ICU4J releases.

- **Official Release:**
  If you want to use ICU4J (as opposed to developing it), your best bet is to download an official, packaged version of the ICU4J library files. These versions are tested more thoroughly than day-to-day development builds, and they are packaged in jar files for convenient download. These packaged files can be found at the ICU Download page.

- **Subversion Source Repository:**
  If you are interested in developing features, patches, or bug fixes for ICU4J, you should probably be working with the latest version of the ICU4J source code. You will need to check the code out of our Subversion repository to ensure that you have the most recent version of all of the files. There are several ways to do this. Please follow the directions that are contained on the Source Repository page for details.

For more details on how to download ICU4J directly from the web site, please see the ICU download page at http://www.icu-project.org/download/

The Structure and Contents of ICU4J

Below, all directory pathes are relative to the directory where the ICU4J source archive is extracted.

Information and build files:

Path	Description
readme.html	A description of ICU4J (International Components for Unicode for Java)
build.html	The main Ant build file for ICU4J. See How to Install and Build for more information
main/shared/licenses/license.html	The X license, used by ICU4J

ICU4J runtime class files:

Path	Sub-component Name	Build Dependencies	Public API Packages	Description
main/classes/charset	icu4j-charset	icu4j-core	com.ibm.icu.charset	Implementation of `java.nio.charset.spi.CharsetProvider`. This sub-component is shipped as icu4j-charset.jar along with ICU charset converter data files.

main/classes/collate	icu4j-collate	icu4j-core	com.ibm.icu.text com.ibm.icu.util	Collator APIs and implementation. Also includes some public API classes that depend on Collator. This sub-component is packaged as a part of icu4j.jar.
main/classes/core	icu4j-core	n/a	com.ibm.icu.lang com.ibm.icu.math com.ibm.icu.text com.ibm.icu.util	ICU core API classes and implementation. This sub-component is packaged as a part of icu4j.jar.
main/classes/currdata	icu4j-currdata	icu4j-core	n/a	No public API classes. Provides access to currency display data. This sub-component is packaged as a part of icu4j.jar.
main/classes/langdata	icu4j-langdata	icu4j-core	n/a	No public API classes. Provides access to language display data. This sub-component is packaged as a part of icu4j.jar.
main/classes/localespi	icu4j-localespi	icu4j-core icu4j-collate	n/a	Implementation of various locale-sensitive service providers defined in `java.text.spi` and `java.util.spi` in J2SE 6.0 or later Java releases. This sub-component is shipped as icu4j-localespi.jar.
main/classes/regiondata	icu4j-regiondata	icu4j-core	n/a	No public API classes. Provides access to region display data. This sub-component is packaged as a part of icu4j.jar.
main/classes/translit	icu4j-translit	icu4j-core	com.ibm.icu.text	Transliterator APIs and implementation. This sub-component is packaged as a part of icu4j.jar.

ICU4J unit test files:

Path	Sub-component Name	Runtime Dependencies	Description
main/tests/charset	icu4j-charset-tests	icu4j-charset icu4j-core icu4j-test-framework	Test suite for charset sub-component.
main/tests/collate	icu4j-collate-tests	icu4j-collate icu4j-core icu4j-test-framework	Test suite for collate sub-component.
main/tests/core	icu4j-core-tests	icu4j-core icu4j-currdata icu4j-langdata icu4j-regiondata icu4j-test-framework	Test suite for core sub-component.
main/tests/framework	icu4j-test-framework	icu4j-core	Common ICU4J unit test framework and utilities.
main/tests/localespi	icu4j-localespi-tests	icu4j-core icu4j-collate icu4j-currdata icu4j-langdata icu4j-localespi icu4j-regiondata icu4j-test-framework	Test suite for localespi sub-component.
main/tests/packaging	icu4j-packaging-tests	icu4j-core icu4j-test-framework	Test suite for sub-component packaging.

main/tests/translit	icu4j-translit-tests	icu4j-core icu4j-translit icu4j-test-framework	Test suite for translit sub-component.

Others:

Path	Description
main/shared	Files shared by ICU4J sub-components under the `main` directory including: • ICU4J runtime data archive (icudata.jar). • ICU4J unit test data archive (testdata.jar). • Shared Ant build script and configuration files. • License files.
demos	ICU4J demo programs.
perf-tests	ICU4J performance test files.
tools	ICU4J tools including: • Custom JavaDoc taglets used for generating ICU4J API references. • API report tool and data. • Other independent utilities used for ICU4J development.

Where to get Documentation

The ICU user's guide contains lots of general information about ICU, in its C, C++, and Java incarnations.

The complete API documentation for ICU4J (javadoc) is available on the ICU4J web site, and can be built from the sources:

- Index to all ICU4J API
- Charset Detector – Detection of charset from a byte stream
- International Calendars – Buddhist, Chinese, Coptic, Ethiopic, Gregorian, Hebrew, Indian, Islamic, Japanese, Persian, Dangi.
- Time Zone Enhancements – Time zone transition and rule detection, iCalendar VTIMEZONE formatting and parsing, Custom time zones constructed by user defined rules.
- Date Format Enhancements – Date/Time Pattern Generator, Date Interval Format, Duration Format.
- Unicode Normalization – Canonical text representation for W3C.
- Number Format Enhancements – Scientific Notation, Spelled out.
- Enhanced word-break detection – Rule-based, supports Thai
- Transliteration – A general framework for converting text from one format to another, e.g. Cyrillic to Latin, or Hex to Unicode.
- Unicode Text Compression & Decompression – 2:1 compression on English Unicode text.
- Collation – Rule-based sorting, Efficient multi-lingual searching, Alphabetic indexing

How to Install and Build

To install ICU4J, simply place the prebuilt jar file **icu4j.jar** on your Java CLASSPATH. If you need Charset API support please also place **icu4j-charset.jar** on your class path along with **icu4j.jar**.

To build ICU4J, you will need J2SE SDK 5.0 or later (ICU4J locale SPI provider sub-components require J2SE SDK 6.0 or later) and the Ant build system version 1.7 or later. It's recommended to install both the J2SE SDK and Ant somewhere *outside* the ICU4J directory. For example, on Linux you might install these in `/usr/local`.

- Install J2SE SDK 7.
- Install the **Ant** build system. Ant is a portable, Java-based build system similar to make. ICU4J uses Ant because it introduces no other dependencies, it's portable, and it's easier to manage than a collection of makefiles. We currently build ICU4J using a single makefile on all platforms Ant. The build system requires

Ant 1.7 or later.

Installing Ant is straightforward. Download it (see http://ant.apache.org/bindownload.cgi), extract it onto your system, set some environment variables, and add its bin directory to your path. For example:

```
set JAVA_HOME=C:\jdk1.7.0
set ANT_HOME=C:\ant
set PATH=%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
```

See the current Ant documentation for details.

Once the J2SE SDK and Ant are installed, building is just a matter of typing **ant** in the ICU4J root directory. This causes the Ant build system to perform the build target **jar** as specified by the file **build.xml**, located in the ICU4J root directory. You can give Ant options like -verbose, and you can specify other targets. For example:

```
C:\icu4j>ant
Buildfile: C:\icu4j\build.xml

info:
     [echo] ----- Build Environment Information -------------------
     [echo] Java Home:    C:\jdk1.7.0\jre
     [echo] Java Version: 1.7.0
     [echo] Ant Home:     C:\ant
     [echo] Ant Version:  Apache Ant(TM) version 1.9.4 compiled on April 29 2014
     [echo] OS:           Windows 7
     [echo] OS Version:   6.1
     [echo] OS Arch:      amd64
     [echo] Host:         ICUDEV
     [echo] ------------------------------------------------------

core:

@compile:
     [echo] build-local:     ../../shared/../../build-local.properties
     [echo] ${java5.bootclasspath}
     [echo] --- java compiler arguments ------------------------
     [echo] source dir:     C:\icu4j\main\classes\core/src
     [echo] output dir:     C:\icu4j\main\classes\core/out/bin
     [echo] bootclasspath:
     [echo] classpath:
     [echo] source:         1.5
     [echo] target:         1.5
     [echo] debug:          on
     [echo] encoding:       UTF-8
     [echo] compiler arg:   -Xlint:all,-deprecation,-dep-ann,-options
     [echo] ------------------------------------------------------
    [mkdir] Created dir: C:\icu4j\main\classes\core\out\bin
    [javac] Compiling 365 source files to C:\icu4j\main\classes\core\out\bin
    [javac] Note: Some input files use or override a deprecated API.
    [javac] Note: Recompile with -Xlint:deprecation for details.

compile:

@copy:
     [copy] Copying 23 files to C:\icu4j\main\classes\core\out\bin

set-icuconfig-datapath:

copy-data:
    [unjar] Expanding: C:\icu4j\main\shared\data\icudata.jar into C:\icu4j\main\
classes\core\out\bin
    [unjar] Expanding: C:\icu4j\main\shared\data\icutzdata.jar into C:\icu4j\mai
n\classes\core\out\bin

...
...
...

_build-localespi:

@compile:
```

```
     [echo] build-local:     ../../shared/../../build-local.properties
     [echo] ${java5.bootclasspath}
     [echo] --- java compiler arguments ------------------------
     [echo] source dir:     C:\icu4j\main\classes\localespi/src
     [echo] output dir:     C:\icu4j\main\classes\localespi/out/bin
     [echo] classpath:      C:\icu4j\main\classes\core\out\lib\icu4j-core.jar;C:
\icu4j\main\classes\collate\out\lib\icu4j-collate.jar
     [echo] source:         1.6
     [echo] target:         1.6
     [echo] debug:          on
     [echo] encoding:       UTF-8
     [echo] compiler arg:   -Xlint:all,-deprecation,-dep-ann,-options
     [echo] ----------------------------------------------------
    [mkdir] Created dir: C:\icu4j\main\classes\localespi\out\bin
    [javac] Compiling 22 source files to C:\icu4j\main\classes\localespi\out\bin


compile:

@copy:
     [copy] Copying 10 files to C:\icu4j\main\classes\localespi\out\bin

copy:

@jar:
    [mkdir] Created dir: C:\icu4j\main\classes\localespi\out\lib
     [copy] Copying 1 file to C:\icu4j\main\classes\localespi\out
      [jar] Building jar: C:\icu4j\main\classes\localespi\out\lib\icu4j-localesp
i.jar

jar:

@src-jar:
      [jar] Building jar: C:\icu4j\main\classes\localespi\out\lib\icu4j-localesp
i-src.jar

src-jar:

build:

jar:
     [copy] Copying 1 file to C:\icu4j
     [copy] Copying 1 file to C:\icu4j

BUILD SUCCESSFUL
Total time: 50 seconds
```

Note: The above output is an example. The numbers are likely to be different with the current version ICU4J.

The following are some targets that you can provide to **ant**. For more targets run `ant -projecthelp` or see the build.xml file.

jar (default)	Create ICU4J runtime library jar archives (`icu4j.jar`, `icu4j-charset.jar` and `icu4j-localespi.jar`) in the root ICU4J directory.
check	Build all ICU4J runtime library classes and corresponding unit test cases, then run the tests.
clean	Remove all build output files.
main	Build all ICU4J runtime library sub-components (under the directory `main/classes`).
tests	Build all ICU4J unit test sub-components (under the directory `main/tests`) and their dependencies.
tools	Build the tools.
docs	Run javadoc over the ICU4J runtime library files, generating an HTML documentation tree in the subdirectory `doc`.
jarDocs	Create ICU4J doc jar archive (`icu4jdocs.jar`) containing API reference docs in the root ICU4J directory.
jarDemos	Create ICU4J demo jar archive (`icu4jdemos.jar`) in the root ICU4J directory.

For more information, read the Ant documentation and the **build.xml** file.

Note: If you get an OutOfMemoryError when you are running `"ant check"`, you can set the heap size of the jvm by setting the environment variable JVM_OPTIONS to the appropriate java options.

Eclipse users: See the ICU4J site for information on how to configure Eclipse to build and develop ICU4J on Eclipse IDE.

Note: To install and configure ICU4J Locale Service Provider, please refer the user guide page ICU4J Locale Service Provider.

How to modularize ICU4J

Some clients may not wish to ship all of ICU4J with their application, since the application might only use a small part of ICU4J. ICU4J release 2.6 and later provide build options to build individual ICU4J 'modules' for a more compact distribution. For more details, please refer to the section *Modularization of ICU4J* in the ICU user's guide article Packaging ICU4J.

Trying Out ICU4J

Note: the demos provided with ICU4J are for the most part undocumented. This list can show you where to look, but you'll have to experiment a bit. The demos are **unsupported** and may change or disappear without notice.

The icu4j.jar file contains only the ICU4J runtime library classes, not the demo classes, so unless you build ICU4J there is little to try out.

Charset

To try out the **Charset** package, build **icu4j.jar** and **icu4j-charset.jar** using the 'jar' target. You can use the charsets by placing these files on your classpath.

```
java -cp $icu4j_root/icu4j.jar:$icu4j_root/icu4j-charset.jar <your program>
```

Other demos

The other demo programs are **not supported** and exist only to let you experiment with the ICU4J classes. First, build ICU4J using `ant jarDemos`. Then launch the demos as below:

```
java -jar $icu4j_root/icu4jdemos.jar
```

ICU4J Resource Information

Starting with release 2.1, ICU4J includes its own resource information which is completely independent of the JRE resource information. (Note, ICU4J 2.8 to 3.4, time zone information depends on the underlying JRE). The ICU4J resource information is equivalent to the information in ICU4C and many resources are, in fact, the same binary files that ICU4C uses.

By default the ICU4J distribution includes all of the standard resource information. It is located under the directory com/ibm/icu/impl/data. Depending on the service, the data is in different locations and in different formats. **Note:** This will continue to change from release to release, so clients should not depend on the exact organization of the data in ICU4J.

- The primary **locale data** is under the directory `icudt56b`, as a set of `".res"` files whose names are the locale identifiers. Locale naming is documented the `com.ibm.icu.util.ULocale` class, and the use of these names in searching for resources is documented in `com.ibm.icu.util.UResourceBundle`.
- The **break iterator data** is under the directory `icudt56b/brkitr`, as a set of `".res"`, `".brk"` and `".dict"` files.
- The **collation data** is under the directory `icudt56b/coll`, as a set of `".res"` files.
- The **currency display name data** is under the directory `icudt56b/curr`, as a set of `".res"` files.
- The **language display name data** is under the directory `icudt56b/lang`, as a set of `".res"` files.

- The **rule-based number format data** is under the directory `icudt56b/rbnf`, as a set of `".res"` files.
- The **region display name data** is under the directory `icudt56b/region`, as a set of `".res"` files.
- The **rule-based transliterator data** is under the directory `icudt56b/translit`, as a set of `".res"` files.
- The **measurement unit data** is under the directory `icudt56b/unit`, as a set of `".res"` files.
- The **time zone display name data** is under the directory `icudt56b/zone`, as a set of `".res"` files.
- The **character property data** and default **unicode collation algorithm (UCA) data** is found under the directory `icudt56b`, as a set of `".icu"` files.
- The **normalization data** is found under the directory `icudt56b`, as a set of `".nrm"` files.
- The **character set converter data** is under the directory `icudt56b`, as a set of `".cnv"` files. These files are currently included only in icu-charset.jar.
- The **time zone rule data** is under the directory `icudt56b`, as `zoneinfo64.res`.
- The **holiday data** is under the directory `icudt56b`, as a set of `".class"` files, named `"HolidayBundle_"` followed by the locale ID.

Some of the data files alias or otherwise reference data from other data files. One reason for this is because some locale names have changed. For example, `he_IL` used to be `iw_IL`. In order to support both names but not duplicate the data, one of the resource files refers to the other file's data. In other cases, a file may alias a portion of another file's data in order to save space. Currently ICU4J provides no tool for revealing these dependencies.

> **Note:** Java's `Locale` class silently converts the language code `"he"` to `"iw"` when you construct the Locale (for versions of Java through Java 5). Thus Java cannot be used to locate resources that use the `"he"` language code. ICU, on the other hand, does not perform this conversion in ULocale, and instead uses aliasing in the locale data to represent the same set of data under different locale ids.

Resource files that use locale ids form a hierarchy, with up to four levels: a root, language, region (country), and variant. Searches for locale data attempt to match as far down the hierarchy as possible, for example, `"he_IL"` will match `he_IL`, but `"he_US"` will match `he` (since there is no `US` variant for he, and `"xx_YY` will match root (the default fallback locale) since there is no `xx` language code in the locale hierarchy. Again, see `java.util.ResourceBundle` for more information.

Currently ICU4J provides no tool for revealing these dependencies between data files, so trimming the data directly in the ICU4J project is a hit-or-miss affair. The key point when you remove data is to make sure to remove all dependencies on that data as well. For example, if you remove `he.res`, you need to remove `he_IL.res`, since it is lower in the hierarchy, and you must remove iw.res, since it references `he.res`, and `iw_IL.res`, since it depends on it (and also references `he_IL.res`).

Unfortunately, the jar tool in the JDK provides no way to remove items from a jar file. Thus you have to extract the resources, remove the ones you don't want, and then create a new jar file with the remining resources. See the jar tool information for how to do this. Before 'rejaring' the files, be sure to thoroughly test your application with the remaining resources, making sure each required resource is present.

Using additional resource files with ICU4J

> **Warning: Resource file formats can change across releases of ICU4J!**
>
> The format of ICU4J resources is not part of the API. Clients who develop their own resources for use with ICU4J should be prepared to regenerate them when they move to new releases of ICU4J.

We are still developing ICU4J's resource mechanism. Currently it is not possible to mix icu's new binary `.res` resources with traditional java-style `.class` or `.txt` resources. We might allow for this in a future release, but since the resource data and format is not formally supported, you run the risk of incompatibilities with future releases of ICU4J.

Resource data in ICU4J is checked in to the repository as a jar file containing the resource binaries, `$icu4j_root/main/shared/data/icudata.jar`. This means that inspecting the contents of these resources is difficult. They currently are compiled from ICU4C `.txt` file data. You can view the contents of the ICU4C text resource files to understand the contents of the ICU4J resources.

The files in `icudata.jar` get extracted to `com/ibm/icu/impl/data` in the build output directory by some build targets.

Building ICU4J Resources from ICU4C

ICU4J data is built by ICU4C tools. Please see "icu4j-readme.txt" in *$icu4c_root*/source/data for the procedures.

Generating Data from CLDR

Note: This procedure assumes that all 3 sources are present

1. Checkout or download CLDR version 'release-28'
2. Checkout ICU4C with tag 'release-56-rc'
3. Checkout ICU4J with tag 'release-56-rc'
4. cd to *$icu4c_root*/source/data directory
5. Follow the instructions in *$icu4c_root*/source/data/cldr-icu-readme.txt
6. Rebuild ICU4C with the newly generated data.
7. Run ICU4C tests to verify that the new data is good.
8. Build ICU4J data from ICU4C data by following the procedures in *$icu4c_root*/source/data/icu4j-readme.txt
9. cd to *$icu4j_root* dir
10. Build and test icu4j

About ICU4J Time Zone

ICU4J 56 Release Candidate includes time zone data version 2015f, which is the latest one as of the release date. However, time zone data is frequently updated in response to changes made by local governments around the world. If you need to update the time zone data, please refer the ICU user guide topic Updating the Time Zone Data.

Starting with ICU4J 4.0, you can optionally configure ICU4J date and time service classes to use underlying JDK TimeZone implementation (see the ICU4J API reference TimeZone for the details). When this configuration is enabled, ICU's own time zone data won't be used and you have to get time zone data patches from the JRE vendor.

Where to Find More Information

http://www.icu-project.org/ is the home page of International Components for Unicode development project

http://www.ibm.com/software/globalization/icu/ is a pointer to general information about the International Components for Unicode hosted by IBM

http://www.ibm.com/software/globalization/ is a pointer to information on how to make applications global.

Submitting Comments, Requesting Features and Reporting Bugs

Your comments are important to making ICU4J successful. We are committed to investigate any bug reports or suggestions, and will use your feedback to help plan future releases.

To submit comments, request features and report bugs, please see ICU bug database information or contact us through the ICU Support mailing list. While we are not able to respond individually to each comment, we do review all comments.

Thank you for your interest in ICU4J!