

**TABLE OF CONTENTS**

1. [INTRODUCTION](#)
  1. [Copyright Notice](#)
  2. [About BEQ Technologies](#)
  3. [Target Audience](#)
  4. [System Requirements](#)
    1. [Usage Requirements](#)
    2. [Build Requirements](#)
2. [DISTRIBUTION STRUCTURE](#)
  1. [BIN Directory](#)
  2. [DOC Directory](#)
  3. [SRC Directory](#)
    1. [BUILD Directory](#)
    2. [JAVA Directory](#)
    3. [JNI Directory](#)
3. [BUILDING](#)
4. [USAGE](#)
  1. [Examples](#)
    1. [Creating A New Key](#)
    2. [Creating A Subkey](#)
    3. [Deleting An Existing Key](#)
    4. [Enumerating Subkeys](#)
    5. [Reading A Value](#)
    6. [Writing A Value](#)
    7. [Enumerating Values](#)
5. [DOCUMENTATION](#)
  1. [Design Documentation](#)
  2. [API Documentation](#)

---

**1. INTRODUCTION**

***jRegistryKey*** is a [Java](#)<sup>™</sup> Native Interface (JNI) wrapper around the Microsoft® Windows® Win32® application programming interface (API) [registry](#)

[functions](#), designed to facilitate Windows® registry access for Java™ developers.

Originally, *BEQ Technologies* required access to the system registry to read the MIME types and associated shell commands to replicate the native behaviour of Windows® Explorer when operating on files residing on a Windows® system. After testing a number of open-source products it was decided that none were full-featured nor robust enough to satisfy requirements. Therefore, *jRegistryKey* was developed.

Where possible, *BEQ Technologies* uses open-source software, and it was eventually decided that, for good karma, we should release *jRegistryKey* as an open-source product under the LGPL. We, the *BEQ Technologies* development team, hope that you find *jRegistryKey* to be a useful library, and that you, too, will consider supporting the open-source software movement -- nice to meet GNU!

## 1. Copyright Notice

*jRegistryKey* - A JNI wrapper of the Windows® Registry functions.

Copyright © 2001, BEQ Technologies Inc.

#205, 3132 Parsons Road

Edmonton, Alberta

T6N 1L6 Canada

(780) 430-0056

(780) 437-6121 (fax)

<http://www.beq.ca>

This library is free software; you can redistribute it and/or modify it under the terms of the [GNU Lesser General Public License](#) as published by the [Free Software Foundation](#); either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

## 2. About BEQ Technologies

*BEQ Technologies Inc.* is the research-and-development division of [Bay Equities Inc.](#), a publicly traded company on the [Canadian Venture Exchange](#) (CDNX) traded under the symbol '[BEQ](#)'.

*BEQ Technologies* is a leader in high-performance real-time Java™ software design and development, and provides a host of comprehensive, client-focused technology solutions.

## 3. Target Audience

This document is intended for software developers, and, therefore, assumes that the reader is at least somewhat familiar with the following topics:

- the Java™ programming language,
- the [Unified Modelling Language](#) (UML), and
- the Windows® registry

Additionally, individuals wishing to build the *jRegistryKey* project should be familiar with the following topics:

- [Jakarta](#) Ant,
- Gamma and Beck's [JUnit](#), and
- Microsoft® Visual C++®

#### 4. System Requirements

Because the system registry is specific to the Windows® family of operating systems, *jRegistryKey* is an atypical Java™ library in that it will only operate correctly on Windows® platforms. *jRegistryKey* should, but has not necessarily been tested to, operate on the following Microsoft® platforms:

- Windows® 95,
- Windows® 98,
- Windows® ME,
- Windows® NT,
- Windows® 2000, and
- Windows® XP

##### 1. Usage Requirements

*jRegistryKey* is a JNI library. To use *jRegistryKey*, the following files are required:

- jRegistryKey.jar
- jRegistryKey.dll

jRegistryKey.jar is the Java™ Archive (JAR) file containing the packaged Java™ class files, whereas jRegistryKey.dll is a Windows® dynamically linked library (DLL) that contains the native (C/C++) code required to access the registry.

jRegistryKey.jar must be included in the CLASSPATH available to the Java™ Virtual Machine (JVM); jRegistryKey.dll must be located in a directory included in the Windows® PATH environment variable or java.lang.UnsatisfiedLinkError's will be generated.

##### 2. Build Requirements

Building *jRegistryKey* is significantly more involved than simply using *jRegistryKey*. To successfully build *jRegistryKey*, the following packages are required to be installed and properly configured:

- Jakarta Ant 1.4 (or later),
- Java™ 2 Software Development Kit (J2SDK) 1.3.1 (or later),

- JUnit 3.7 (or later),
- Microsoft® Visual C++® 6.0 SP3 (or later)

**Important Note:** As specified in the Ant documentation, the environment variables ANT\_HOME, and JAVA\_HOME must be correctly defined to build the *jRegistryKey* project. Furthermore, in addition to the standard required Visual C++® environment variables, the environment variable MSVC\_HOME must be defined to point to the Visual C++® root directory.

---

## 2. DISTRIBUTION STRUCTURE

The following list shows the hierarchy and locations of key files in the *jRegistryKey* distribution package:

```
\jRegistryKey
  \1.0
    \bin
      * jRegistryKey.dll
      * jRegistryKey.jar
      * jRegistryKeyTest.jar
    \doc
      \api
      \manual
      \uml
    \src
      \build
        \dist
          \classes
          \javadoc
          \reports
        \lib
          * ant.jar
          * jakarta-ant-1.4-optional.jar
          * junit.jar
        * build.bat
        * build.xml
      \java
        \api
          \ca
            \beq
              \util
                \win32
                  \registry
                    * KeyIterator.java
```

```

        * package.html
        * RegistryException.java
        * RegistryKey.java
        * RegistryValue.java
        * RootKey.java
        * ValueIterator.java
        * ValueType.java

\test
  \ca
    \beq
      \util
        \win32
          \registry
            * AllTests.java
            * RegistryKeyTest.java

\jni
  \include
    * ca_beq_util_win32_registry_KeyIterator.h
    * ca_beq_util_win32_registry_RegistryKey.h
    * ca_beq_util_win32_registry_ValueIterator.h
    * jniClassDescriptor.h
    * jniFieldDescriptor.h
    * jRegistryKey.h
  \jRegistryKey.dll
    * build.xml
    * KeyIterator.cpp
    * RegistryKey.cpp
    * ValueIterator.cpp

```

## 1. BIN Directory

The BIN (BINARY) directory contains a copy of the compiled ***jRegistryKey*** binaries. Binaries are placed in this directory simply for ease of access for those who wish to use, but not necessarily rebuild, ***jRegistryKey***. Note that these binaries are **not** modified when the ***jRegistryKey*** project is rebuilt.

## 2. DOC Directory

The DOC (DOCUMENTATION) directory is the root of the ***jRegistryKey*** project documentation and contains the following sub-directories:

- API - a copy of the Javadoc API documentation,
- MANUAL - the User Manual, and
- UML - the Argo/UML design documentation

### 3. SRC Directory

The SRC (SOURCE) directory is the root of the *jRegistryKey* project source code tree. The following sections detail each of the three sub-directories.

#### 1. BUILD Directory

The BUILD directory contains exactly two sub-directories and three files.

The DIST (DISTRIBUTION) directory contains the output files of generated by rebuilding the project. The resultant binaries (jRegistryKey.dll, jRegistryKey.jar, and jRegistryKeyTest.jar) are automatically placed in the build directory after a rebuild.

The LIB (LIBRARY) directory contains those Java™ libraries required to build the *jRegistryKey* project.

The BUILD.BAT file is a Windows® command-script (batch file) that adds all Java™ archives in the LIB directory to the CLASSPATH and then executes Ant.

The BUILD.XML file is the master Ant build-file for the *jRegistryKey* project. By default, the Ant build compiles the Java™ source code (Javac), compiles the native C/C++ code (Visual C++®), constructs the API documentation (Javadoc), and finally executes the unit tests (JUnit).

#### 2. JAVA Directory

The JAVA directory contains exactly two sub-directories: API, and TEST.

The API directory contains the Java™ source code in the appropriate package hierarchy.

The TEST directory exactly mirrors the package directory hierarchy, but contains the Java™ source code for the JUnit unit tests.

#### 3. JNI Directory

The JNI directory contains two directories required for building the jRegistryKey.dll.

The INCLUDE directory contains header files required by the Visual C++® command-line compiler (CL.EXE).

The JREGISTRYKEY.DLL directory contains the C/C++ source code files that implement the native portion (jRegistryKey.dll) of the *jRegistryKey* project.

---

### 3. BUILDING

Assuming you've read the [Build Requirements](#) section and the applications listed therein are properly installed and configured on your system, rebuilding the *jRegistryKey* binaries is as simple as executing Ant on the master build file (you did actually \*read\* the [Distribution Structure](#) section, so you know

where the master build file is, don't you?)

---

## 4. USAGE

As specified in the [Usage Requirements](#) section, before attempting to use *jRegistryKey* in a Java™ project, ensure that:

- jRegistryKey.jar is in the CLASSPATH, and
- jRegistryKey.dll is either a) in the current directory, or b) in the PATH

To use *jRegistryKey* in a Java™ project, simply import the classes from the `ca.beq.util.win32.registry` package, as shown in the following example:

```
import ca.beq.util.win32.registry.*;
```

There are two main classes in the package: `RegistryKey`, and `RegistryValue`.

`RegistryKey` is the Java™ representation of a registry key, and provides methods to create and delete keys, enumerate subkeys and values, and set and retrieve values.

`RegistryValue` is the Java™ representation of a registry value (defined as a name, a type, and data).

### 1. Examples

The following sections provides sample code snippets designed to aid developers in understanding the design and usage of *jRegistryKey*.

#### 1. Creating A New Key

The following code snippet creates the new key "BEQ Technologies" under the key "HKEY\_CURRENT\_USER\Software":

```
RegistryKey r = new RegistryKey(RootKey.HKEY_CURRENT_USER, "Software\\BEQ  
Technologies");  
r.create();
```

#### 2. Creating A Subkey

The following code snippet is a variant of the above code, and creates the "BEQ Technologies" key using the `createSubkey( )` method:

```
RegistryKey r = new RegistryKey(RootKey.HKEY_CURRENT_USER, "Software");  
r.createSubkey("BEQ Technologies");
```

### 3. Deleting An Existing Key

The following code snippet deletes the key "BEQ Technologies" from the key "HKEY\_CURRENT\_USER\Software" (if the key doesn't already exist, a `RegistryException` will be thrown):

```
try {  
    RegistryKey r = new RegistryKey(RootKey.HKEY_CURRENT_USER, "Software\\BEQ  
Technologies");  
    r.delete();  
} // try  
catch(RegistryException re) {  
    re.printStackTrace();  
} // catch
```

### 4. Enumerating Subkeys

The following code snippet demonstrates how to use the `java.util.Iterator` class to enumerate and iterate over the subkeys of the key "HKEY\_CURRENT\_USER\Software":

```
RegistryKey r = new RegistryKey(RootKey.HKEY_CURRENT_USER, "Software");  
if(r.hasSubkeys()) {  
    Iterator i = r.subkeys();  
    while(i.hasNext()) {  
        RegistryKey x = (RegistryKey)i.next();  
        System.out.println(x.toString());  
    } // while  
} // if
```

### 5. Reading A Value

The following code snippet demonstrates how to read a value from a registry key:



```

RegistryKey r = new RegistryKey(RootKey.HKEY_CURRENT_USER, "Software\\BEQ
Technologies");
if(r.hasValue("myValue")) {
    RegistryValue v = r.getValue("myValue");
    System.out.println(v.toString());
} // if

```

## 6. Writing A Value

The following code snippet demonstrates how to write a value to a registry key:

```

RegistryKey r = new RegistryKey(RootKey.HKEY_CURRENT_USER, "Software\\BEQ
Technologies");
RegistryValue v = new RegistryValue("myValue", ValueType.REG_SZ, "This is my data");
r.setValue(v);

```

## 7. Enumerating Values

The following code snippet demonstrates how to use the `java.util.Iterator` class to enumerate and iterate over the values of the key "HKEY\_CURRENT\_USER\Software":

```

RegistryKey r = new RegistryKey(RootKey.HKEY_CURRENT_USER, "Software");
if(r.hasValues()) {
    Iterator i = r.values();
    while(i.hasNext()) {
        RegistryValue v = (RegistryValue)i.next();
        System.out.println(v.toString());
    } // while
} // if

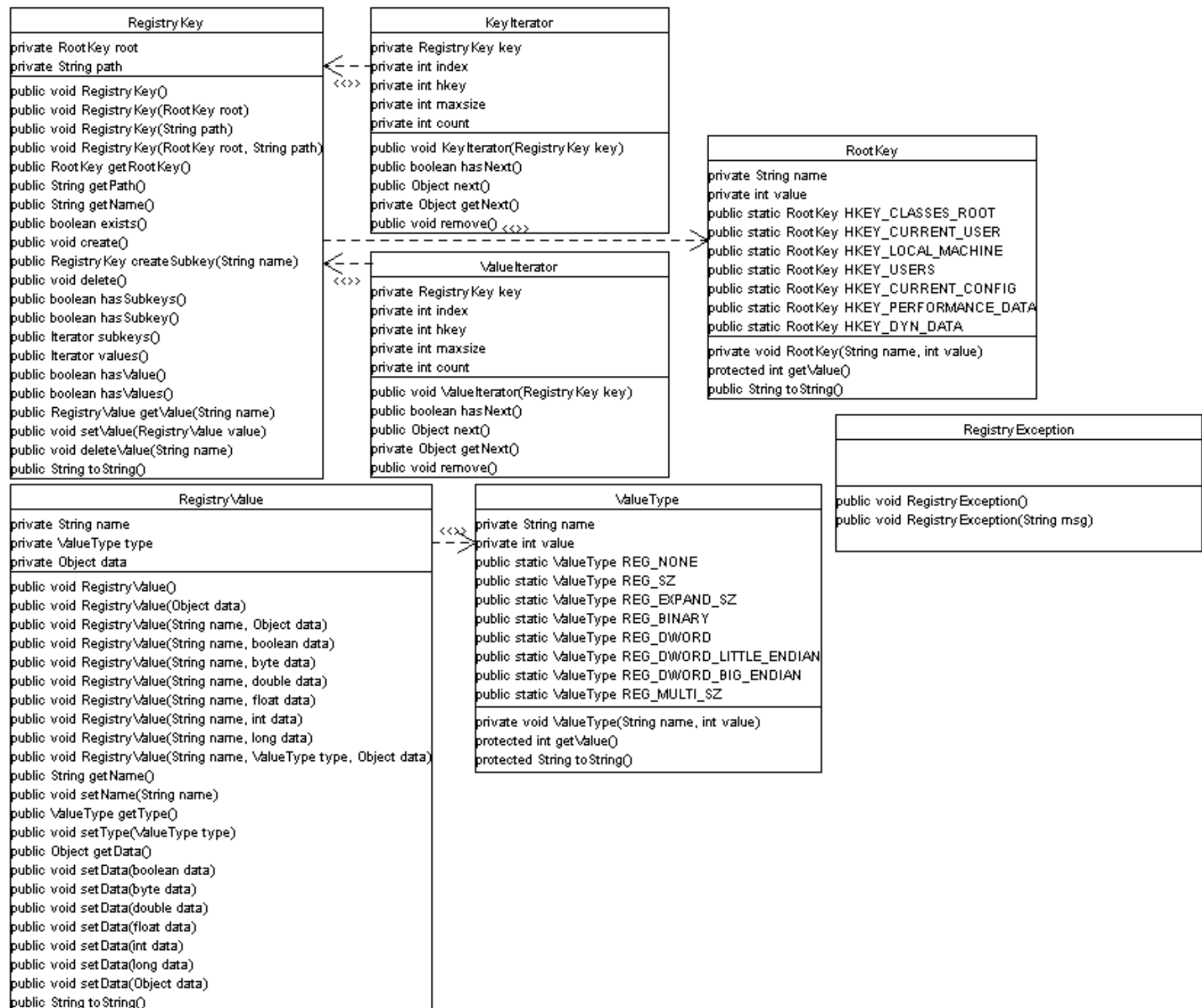
```

---

## 5. DOCUMENTATION

### 1. Design Documentation

*BEQ Technologies* uses [Argo/UML](#) for UML design documentation. The Argo project documentation can be found in the `jRegistryKey/1.0/doc/uml` directory, and is displayed below as figure 1.



*Figure 1. jRegistryKey UML Class Diagrams.*

## **2. API Documentation**

*BEQ Technologies uses the [Javadoc Tool](#) to generate API documentation from source code. The [API documentation](#) is available in the `jRegistryKey/1.0/doc/api` directory.*