Table Of Content

Package com.netcommwireless.javardb

Interface Summary

RDBSubscriber

Used as a callback when a subscribed variable changes.

Class Summary

AsyncRDB

Implements RDB subscriptions with a callback mechanism.

RDB

Manages the RDB session and interfaces to librdb.

RDBException

Thrown by the RDB.*E() functions to pass the library's error code.

RDBVar

Stores all information about a single RDB variable.

---

com.netcommwireless.javardb

Class AsyncRDB

```
java.lang.Object
   |
   +--RDB
      |
      +--com.netcommwireless.javardb.AsyncRDB
```

< Constructors > < Methods >

public class **AsyncRDB**
extends RDB

Implements RDB subscriptions with a callback mechanism. This uses generics so it requires at least Java 1.5 - the other classes work fine with Java 1.4. It should be possible to make it backward compatible with 'javac -source 1.5 -target jsr14'

Constructors

AsyncRDB

```
public  AsyncRDB()
        throws RDBException
```

Create session using default device. Equivalent to AsyncRDB("")

Throws:

com.netcommwireless.javardb.RDBException - if session can't be created

---

AsyncRDB

```
public  AsyncRDB(java.lang.String dev)
        throws RDBException
```

Create RDB session using given device and start select loop.

Parameters:

dev - Path to device file, "" means default

Throws:

com.netcommwireless.javardb.RDBException - if session can't be created

Methods

closeLib

```
public synchronized void closeLib()
```

Stop select loop then close session.

Overrides:

closeLib in class RDB

---

subscribe

```
public synchronized int subscribe(java.lang.String varName)
```

NOT AVAILABLE - will throw Error(). Use RDB instead if you want this interface.

Overrides:

subscribe in class RDB

---

subscribe

```
public synchronized int subscribe(java.lang.String varName,
                                  RDBSubscriber sub)
```

Subscribes for notifications if the given EXISTING variable is written or deleted. A process can only subscribe to notifications if the variable is readable by that process.

The callback will only be run (at most) once for each write. In the case of multiple quick writes to a single var it's possible that some of the intermediate values won't cause callbacks, but the final value always will.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
sub - Object to be notified when the variable is written

Returns:

0 on success, or a negative value (-ENOENT, -EBUSY, -EPERM)

---

unsubscribe

```
public synchronized void unsubscribe(java.lang.String varName,
                                     RDBSubscriber sub)
```

Undo a previous call to subscribe() with the given varName and sub. If there's no match with a previous call nothing is done. If there's more than one match, only one of them is removed.

PERFORMANCE NOTE: technically the underlying library has no unsubscibe; this method simply removes the mapping to a callback, which causes the notification to be ignored.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
sub - Object to be notified when the variable is written

---

unsubscribeAll

```
public synchronized void unsubscribeAll(java.lang.String varName)
```

Undo all previous calls to subscribe().

PERFORMANCE NOTE: technically the underlying library has no unsubscibe; this method simply removes the mapping to the callbacks, which causes the notification to be ignored.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
sub - Object to be notified when the variable is written

---

waitForTriggers

```
public com.netcommwireless.javardb.RDBVar[] waitForTriggers(long milliseconds)
```

NOT AVAILABLE - will throw Error(). Use RDB instead if you want this interface.

Overrides:

waitForTriggers in class RDB

---

com.netcommwireless.javardb

Class RDB

```
java.lang.Object
    |
    +--com.netcommwireless.javardb.RDB
```

Direct Known Subclasses:

AsyncRDB

< Fields > < Constructors > < Methods >

---

public class **RDB**
extends java.lang.Object

Manages the RDB session and interfaces to librdb. * Multithread-safe (only one thread can waitForTriggers()) * Supports binary values * Can choose to use either Java Exceptions or errno return codes to handle errors

Author:

Bill Bennett william.bennett@netcommwireless.com

Fields

CRYPT

```
public static int CRYPT
```

Flags (bitmask constants) used for getNames(), getVars() and RDBVar.flags. Combine them with the bitwise OR operator, e.g. PERSIST | CRYPT. These can't be final because the JNI code sets them at load.

EBUSY

```
public static int EBUSY
```

Error codes to compare with function return values. Note the return values are negative to compare like: if (get() == -ENOENT) These can't be final because the JNI code sets them at load.

EFAULT

`public static int` **`EFAULT`**

Error codes to compare with function return values. Note the return values are negative to compare like: if (get() == -ENOENT) These can't be final because the JNI code sets them at load.

ENOENT

`public static int` **`ENOENT`**

Error codes to compare with function return values. Note the return values are negative to compare like: if (get() == -ENOENT) These can't be final because the JNI code sets them at load.

EOVERFLOW

`public static int` **`EOVERFLOW`**

Error codes to compare with function return values. Note the return values are negative to compare like: if (get() == -ENOENT) These can't be final because the JNI code sets them at load.

EPERM

`public static int` **`EPERM`**

Error codes to compare with function return values. Note the return values are negative to compare like: if (get() == -ENOENT) These can't be final because the JNI code sets them at load.

HASH

`public static int` **`HASH`**

Flags (bitmask constants) used for getNames(), getVars() and RDBVar.flags. Combine them with the bitwise OR operator, e.g. PERSIST | CRYPT. These can't be final because the JNI code sets them at load.

PERSIST

`public static int` **`PERSIST`**

Flags (bitmask constants) used for getNames(), getVars() and RDBVar.flags. Combine them with the bitwise OR operator, e.g. PERSIST | CRYPT. These can't be final because the JNI code sets them at load.

READ_ONCE

`public static int` **`READ_ONCE`**

Flags (bitmask constants) used for getNames(), getVars() and RDBVar.flags. Combine them with the bitwise OR operator, e.g. PERSIST | CRYPT. These can't be final because the JNI code sets them at load.

Constructors

RDB

```
public  RDB()
        throws RDBException
```

Create session using default device. Equivalent to RDB("")

Throws:

com.netcommwireless.javardb.RDBException - if session can't be created

---

RDB

```
public  RDB(java.lang.String dev)
        throws RDBException
```

Create RDB session using given device.

Parameters:

dev - Path to device file, "" means default

Throws:

com.netcommwireless.javardb.RDBException - if session can't be created

Methods

closeLib

```
public synchronized native void closeLib()
```

Close RDB session - must not make any more native calls after this.

---

create

```
public synchronized native int create(RDBVar var)
```

Creates a new variable in database with given flags and perms. Variable must NOT exist.

Parameters:

var - Data to write. In particular the .name must be an ASCII variable name, less than NAMESIZE characters long

Returns:

0 on success, or a negative value (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

createE

```
public void createE(java.lang.String varName,
                    java.lang.String varValue)
             throws RDBException
```

Creates a new variable in database with 0 flags and perms. Variable must NOT exist.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
varValue - value to write - will be converted to (modified) UTF8

Throws:

com.netcommwireless.javardb.RDBException - on failure (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

delete

```
public synchronized native int delete(java.lang.String varName)
```

Deletes an existing variable from database.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long

Returns:

0 on success, or a negative value (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

get

```
public synchronized native int get(RDBVar var)
```

Reads a single EXISTING variable from the database. NOTE: var is used for both input and output - the .name field is used for lookup and then the other fields are filled with data.

Parameters:

var - Must have .name set to an ASCII variable name (less than NAMESIZE characters long), other fields are updated in method

Returns:

0 on success, or a negative value (-ENOENT, -EBUSY, -EPERM, -ENOMEM)

getE

```
public RDBVar getE(java.lang.String varName)
                                          throws RDBException
```

Reads a single EXISTING variable from the database.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long

Returns:

retrieved data

Throws:

com.netcommwireless.javardb.RDBException - on failure (-ENOENT, -EBUSY, -EPERM, -ENOMEM)

---

getFlagsE

```
public int getFlagsE(java.lang.String varName)
           throws RDBException
```

Read a single variable's flags from the database.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long

Returns:

The flags

Throws:

com.netcommwireless.javardb.RDBException - on failure (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

---

getIntDef

```
public int getIntDef(java.lang.String varName,
                     int def)
```

Reads a single variable from the database, or the given default value if it doesn't exist, it's not an integer or some other error occurs.

Parameters:

var - ASCII variable name, less than NAMESIZE characters long

Returns:

The read value or the default value

---

getNames

```
public synchronized native java.lang.String[] getNames(java.lang.String
varNamePart,
                                                       int flagsSet,
                                                       int flagsClear)
```

Searches all variable names from database that have a specific flag.

Parameters:

varNamePart - part of an ASCII variable name
flagsSet - specific flags that must be set ORed together
flagsClear - specific flags that must be clear ORed together

Returns:

array of ASCII variable names that match

---

getStringDef

```
public java.lang.String getStringDef(java.lang.String varName,
                                     java.lang.String def)
```

Reads a single variable from the database, or the given default value if it doesn't exist or some other error occurs.

Parameters:

var - ASCII variable name, less than NAMESIZE characters long

Returns:

The read value or the default value

---

getVars

```
public com.netcommwireless.javardb.RDBVar[] getVars(java.lang.String
varNamePart,
                                                    int flagsSet,
                                                    int flagsClear)
```

Searches all variable names from database that have a specific flag.

Parameters:

varNamePart - part of an ASCII variable name
flagsSet - specific flags that must be set ORed together
flagsClear - specific flags that must be clear ORed together

Returns:

array of matched variables

---

lock

```
public synchronized native int lock()
```

Acquires the database lock. Any RDB operations between lock and unlock are atomic for other RDB users. Be sure to call unlock() ASAP. **WARNING: Do not perform any extensive computation or blocking operations (e.g. file access) while the lock is held.** The driver may kill the process if the lock is held for an excessive amount of time, and that will take out the whole Java VM!

Parameters:

nFlag - Database flags. e.g. NONBLOCK

Returns:

0 on success, or a negative value (-EBUSY)

---

set

```
public synchronized native int set(RDBVar var,
                                   boolean doValue,
                                   boolean doFlags)
```

Writes to a single EXISTING variable. Can write value, flags or both.

Parameters:

var - Data to write. In particular the .name must be an ASCII variable name, less than NAMESIZE characters long
doValue - True if var.value is valid and should be updated
doFlags - True if var.flags is valid and should be updated

Returns:

0 on success, or a negative value (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

---

setFlagsE

```
public void setFlagsE(java.lang.String varName,
                      int nFlags)
           throws RDBException
```

Writes a single variable's flags in database.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
nFlags - flags to write

Throws:

com.netcommwireless.javardb.RDBException - on failure (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

---

setValueE

```
public void setValueE(java.lang.String varName,
                      java.lang.String varValue)
               throws RDBException
```

Writes a single EXISTING variable's value.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
varValue - value to write - will be converted to (modified) UTF8

Throws:

com.netcommwireless.javardb.RDBException - on failure (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

---

subscribe

```
public synchronized native int subscribe(java.lang.String varName)
```

Subscribes for notifications if the given EXISTING variable is written or deleted. A process can only subscribe to notifications if the variable is readable by that process.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
callback - Object to be notified when the variable is written, or null if none

Returns:

0 on success, or a negative value (-ENOENT, -EBUSY, -EPERM)

---

unlock

```
public synchronized native void unlock()
```

Release the database lock obtained with lock().

---

update

```
public synchronized native int update(RDBVar var)
```

Write a variable's value if it already exists or creates a new variable if it doesn't. If created the flags and perms will also be set, but not if it exists.

Parameters:

var - Data to write. In particular the .name must be an ASCII variable name, less than NAMESIZE characters long

Returns:

0 if existing, 1 if created or a negative value (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

updateValue

```
public int updateValue(java.lang.String varName,
                       int varValue)
```

Write a variable's value if it already exists or creates a new variable if it doesn't. If created the flags and perms will be set to 0, but not if it exists.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
varValue - value to write - will be converted to a string

Returns:

0 if existing, 1 if created or a negative value (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

updateValue

```
public int updateValue(java.lang.String varName,
                       java.lang.String varValue)
```

Write a variable's value if it already exists or creates a new variable if it doesn't. If created the flags and perms will be set to 0, but not if it exists.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
varValue - value to write - will be converted to (modified) UTF8

Returns:

0 if existing, 1 if created or a negative value (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

updateValueE

```
public int updateValueE(java.lang.String varName,
                        int varValue)
           throws RDBException
```

Write a variable's value if it already exists or creates a new variable if it doesn't. If created the flags and perms will be set to 0, but not if it exists.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
varValue - value to write - will be converted to a string

Throws:

com.netcommwireless.javardb.RDBException - on failure (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

updateValueE

```
public int updateValueE(java.lang.String varName,
                        java.lang.String varValue)
           throws RDBException
```

Write a variable's value if it already exists or creates a new variable if it doesn't. If created the flags and perms will be set to 0, but not if it exists.

Parameters:

varName - ASCII variable name, less than NAMESIZE characters long
varValue - value to write - will be converted to (modified) UTF8

Throws:

com.netcommwireless.javardb.RDBException - on failure (-ENOENT, -EBUSY, -EPERM, -EOVERFLOW)

---

waitForTriggers

```
public com.netcommwireless.javardb.RDBVar[] waitForTriggers(long milliseconds)
```

Waits for at least one subscribed var to trigger and returns that list. If not using 'clear' then you must call complete() yourself.

Parameters:

clear - if true the returned vars will be marked complete() automatically
milliseconds - Time to wait - 0 means forever

Returns:

Vars that triggered (could be none or a few depending on scheduling and timeout)

---

com.netcommwireless.javardb

Class RDBException

```
java.lang.Object
    |
    +--java.lang.Throwable
        |
        +--java.lang.Exception
            |
            +--com.netcommwireless.javardb.RDBException
```

All Implemented Interfaces:
java.io.Serializable

< Fields > < Constructors > < Methods >

---

public class **RDBException**
extends java.lang.Exception

Thrown by the RDB.*E() functions to pass the library's error code.

Author:

Bill Bennett william.bennett@netcommwireless.com

Fields

errorCode

```
public int errorCode
```

The POSITIVE error code from the C library; compare with RDB.E* vars. The values actually come from errno.h.

---

generator

```
public RDB generator
```

The instance/session that generated the error.

Constructors

RDBException

```
public  RDBException(RDB from,
                     java.lang.String message,
                     int code)
```

Methods

toString

```
public java.lang.String toString()
```

Overrides:

toString in class java.lang.Throwable

---

com.netcommwireless.javardb

Interface RDBSubscriber

< Methods >

---

public interface **RDBSubscriber**

Used as a callback when a subscribed variable changes. Use AsyncRDB.subscribe() and

AsyncRDB.unsubscribe() to manage callbacks.

Each RDB object runs subscribed callbacks sequentially (though in an inderterminate order), so if there's only 1 RDB object in an application it's impossible for more than 1 to be running at a time. You wont need synchronisation if all your application does is wait for a callback, but be careful if you have other threads working in the background and you access their objects.

The callback will only be run (at most) once for each write. In the case of multiple quick writes to a single var it's possible that some of the intermediate values won't cause callbacks, but the final value always will.

Author:

Bill Bennett william.bennett@netcommwireless.com

Methods

callback

```
public void callback(RDBVar newVar,
                     byte[] oldValue,
                     boolean hasChanged)
```

Called when a subscribed variable changes.

Parameters:

newVar - The variable that changed, filled out with the new value.
oldValue - The variable's last known value - not necessarily the most recent value - see discussion in the interface desc
hasChanged - True if the old value and new value differ

---

com.netcommwireless.javardb

Class RDBVar

```
java.lang.Object
    |
    +--com.netcommwireless.javardb.RDBVar
```

< Fields > < Constructors > < Methods >

public class **RDBVar**
extends java.lang.Object

Stores all information about a single RDB variable.

Author:

Bill Bennett william.bennett@netcommwireless.com

Fields

flags

```
public int flags
```

Variable flags - filled before RDB.setFlags() or during RDB.get()

---

name

```
public java.lang.String name
```

Variable name - always filled out before use

---

perms

```
public int perms
```

Variable perms - filled during RDB.get()

---

value

```
public byte[] value
```

Variable value - filled before RDB.set() or during RDB.get()

Constructors

RDBVar

```
public  RDBVar(java.lang.String name)
```

RDBVar

```
public  RDBVar(java.lang.String name,
               byte[] value)
```

RDBVar

```
public  RDBVar(java.lang.String name,
               byte[] value,
               int flags,
               int perms)
```

Methods

flagsToString

```
public java.lang.String flagsToString()
```

---

getInt

```
public int getInt()
           throws java.lang.NumberFormatException
```

Convert value to an integer. Value must be a sequence of digits.

Returns:

result of conversion

Throws:

java.lang.NumberFormatException - if it's not an integer

---

getIntDef

```
public int getIntDef(int def)
```

Convert value to an integer. Value should be a sequence of digits.

Parameters:

def - default value to use if conversion's not possible

Returns:

result of conversion or def if it's not an integer

---

getString

```
public java.lang.String getString()
```

Convert value to a string. Value must be either ASCII or Java's modified UTF8.

Returns:

result of conversion

---

setString

```
public void setString(java.lang.String in)
```

Set value from string. Value will be Java's modified UTF8.

---

toString

```
public java.lang.String toString()
```

Overrides:

toString in class java.lang.Object

INDEX