Java and JCA

Kenneth Evans, Jr.
October 12, 2004

Part of the EPICS “Getting Started” Lecture Series

Argonne National Laboratory

A U.S. Department of Energy
Office of Science Laboratory
Operated by The University of Chicago
Outline

• Java
• Java and EPICS
• Overview of JCA
• Examples
  - SimpleJCAGet
  - SimpleJCAMonitor
  - JProbe
Java

- **Java is designed to be platform independent**
  - Write once, run everywhere
- **Java programs are interpreted by another program, possibly on another machine**
  - The Java Virtual Machine (Java VM)
- **Java technology includes**
  - J2SE Standard Edition
  - J2EE Enterprise Edition (Multi-tier business apps)
  - J2ME Micro Edition (Phones, PDAs, etc.)
- **Java is advertised to be all of these**
  - Simple
  - Architecture neutral
  - Object oriented
  - Portable
  - Distributed
  - High performance
  - Interpreted
  - Multithreaded
  - Robust
  - Dynamic
  - Secure
Java and EPICS

- EPICS Channel Access is native code
  - Not platform independent
  - Build cycle is edit – compile [to machine code] - link - run
- Pure Java
  - Build cycle is edit – compile [to byte code] – run [anywhere]
- Java Native Interface [JNI] is used to access native code
  - Not “Pure Java”
  - No longer platform independent
  - You generate shared object or DLL libraries that must be used with the Java program
- You can write your own JNI
  - Not that hard if you know Channel Access
- The EPICS build system handles Java projects and JNI
- Ant is an alternative
JCA

- Stands for Java Channel Access
- JCA is a JNI implementation of an EPICS Channel Access client library for Java
  - Provides equivalents to most of the Channel Access API
  - Developed by Eric Boucher while at the APS
  - Currently taken over by Cosylab
- Available for download at
  - http://www.cosylab.com/jca
- Latest version is 2.1.3 at Cosylab
- JCA Version 1 uses EPICS Base 3.13
- JCA Version 2 uses EPICS Base 3.14
  - Channel Access is threaded
  - Allows for preemptive callbacks
    - Works better with Java, which is inherently threaded
CAJ

- CAJ is a Java replacement for Channel Access
- Under development at Cosylab (Control Systems Laboratory)
  - [http://www.cosylab.com](http://www.cosylab.com)
  - Located in Ljubljana in Slovenia
  - Situated between Austria (North), Italy (West), Hungary (East) and Croatia (South)
  - Cosylab also develops VDCT
- Will allow your programs to be “Pure Java”
- Can be used with JCA
  - Replaces JNI implementation
  - Requires replacing only one line of code
    - `jca createContext(JCALibraryJNI_THREAD_SAFE);`
    - `jca createContext("com.cosylab.epics.caj.CAJContext");`
Requirements

- Java J2SE installed (Current [suggested] version is 1.4.2)
- JCA
  - Java libraries
    - Download source and/or JAR files from the web
  - Native JNI libraries
    - Download from the web or build them
    - Currently found with the 2.1.2 distribution only
      - jca.dll Windows
      - libjca.so Unix (Currently only Linux)
- Your project
  - JCA files need to be in your CLASSPATH
  - UNIX: Shared object library needs to be in your LD_LIBRARY_PATH
  - Windows: DLL needs to be in your PATH
Resources

• EPICS web pages
  - Look under Extensions, then JCA

• JCA 2.1.2 API

• JCA 2.1.3 API

• Java Tutorial

• J2SE Documentation

• J2SE 1.4.2 API (Javadoc)
  - http://java.sun.com/j2se/1.4.2/docs/api/overview-summary.html
JCA Packages

• Five Packages
  - gov.aps.jca Channel-Access-like routines
  - gov.aps.jca.configuration Configuration
  - gov.aps.jca.dbr DBR types
  - gov.aps.jca.event Event handling
  - gov.aps.jca.jni Native interface functions
gov.aps.jca

• This is the package you will use most directly
• Classes
  - CASeverity Enum
  - CAStatus JCALibrary
  - Channel Monitor
  - Channel.ConnectionState ValuedEnum
  - Context
• Exceptions
  - CAException TimeoutException
JCALibrary

- Initializes JCA
  
  ```java
  JCALibrary jca = JCALibrary.getInstance();
  ```

- There is only one instance
- Used to create contexts and manage JCA configuration info
- Properties
  - JNI_THREAD_SAFE preemptive
    - Suggested for Java, which is inherently threaded
  - JNI_SINGLE_THREADED non-preemptive
- Methods
  - createContext
  - getProperty
  - listProperties
  - getVersion, getRevision, getModification
Context

- Corresponds to a Channel Access context
- Created by JCALibrary createContext
  createContext(JCALibrary.JNI_SINGLE_THREADED)
  createContext(JCALibrary.JNI_THREAD_SAFE)
- Controls all IO
- You can have more than one context
- Methods
  - createChannel
  - flushIO, pendIO, pendEvent, poll
  - attachCurrentThread
  - addContextExceptionListener, removeContextExceptionListener
  - addContextMessageListener, removeContextMessageListener
  - destroy
Channel

• Represents a Channel Access channel
• Created by Context.createChannel
  
  createChannel(String name, connectionListener l)

• Properties
  - CLOSED CONNECTED
  - DISCONNECTED NEVER_CONNECTED

• Methods
  - get, many overloads
  - put, many overloads
  - getName, getConnectionState, getElementCount, etc.
  - addMonitor
  - addConnectionListener, removeConnectionListener
  - addAccessRightsListener, removeAccessRightsListener
  - destroy
Monitor

- Represents a Channel Access monitor
- Created by Channel.addMonitor
  
  ```java
  addMonitor(DBRType type, int count, int mask, MonitorListener l)
  ```

- Properties
  - ALARM LOG VALUE

- Methods
  - addMonitorListener, removeMonitorListener
  - getMonitorListener, getMonitorListeners
  - clear
  - getChannel, getContext
  - getCount, getMask, getType
  - isMonitoringAlarm, isMonitoringLog, isMonitoringValue
MonitorListener

- Part of gov.aps.jca.event
- One method
  - monitorChanged
- Example
  ```java
  private class MyMonitorListener implements MonitorListener {
    public void monitorChanged(MonitorEvent ev) {
      // Call my handler
      onValueChanged(ev);
    }
  }
  ```
- The value and status comes with the MonitorEvent
MonitorEvent

- Part of gov.aps.jca.event
- Methods
  - getDBR How you get the value
  - getStatus How you determine the status
- Example
  
  ```java
  if (ev.getStatus() == CAStatus.NORMAL) {
      DBR dbr = ev.getDBR();
      double [] value = ((DOUBLE)dbr).getDoubleValue();
  }
  ```
Event Types

- MonitorListener: MonitorEvent
- GetListener: GetEvent
- PutListener: PutEvent
- AccessRightsListener: AccessRightsEvent
- ConnectionListener: Connection Event
- ContextExceptionListener: ContextExceptionEvent
- ContextMessageListener: ContextMessageEvent

- Events all inherit from CAEvent
- They all work similarly to Monitor
  - Call the routine that fires the event when it occurs
  - Add a listener with the appropriate handler
  - Get the data from the event that is passed to your handler
gov.aps.jca.db

- Implements the EPICS DBR_XXX types
- Interfaces
  - DOUBLE, FLOAT, INT, STRING, TIME, CTRL, etc.
- Primary Class
  - DBR
- Subclasses of DBR
  - DBR_Double, DBR_Float, DBR_Int, DBR_STS_Double, etc.
- Example: DBR_STS_Double
  - Interfaces
    - STS, DOUBLE
  - Extends
    - DBR_Double
  - Subclasses
    - DBR_GR_Double, DBR_Time_Double
SimpleJCAGet

```java
package simplejca;

import gov.aps.jca.*;
import gov.aps.jca.dbr.*;
```
public class SimpleJCAGet
{
    public static void main(String[] args)
    {
        SimpleJCAGet simpleJCAGet = new SimpleJCAGet();
        JCALibrary jca=null;
        Context ctxt=null;
        Channel chan=null;

        // Parse the command line
        if(!simpleJCAGet.parseCommand(args)) System.exit(1);
        if(!simpleJCAGet.pvSpecified) {
            System.err.println("No PV specified\n");
            System.exit(1);
        }
    }
}
// Initialize and search

try {
    // Get the JCAldinary instance
    jca=JCAldinary.getInstance();
    // Create a non-preemptive context
    context=jca createContext(
        JCAldinary.JNI_SINGLE_THREAD);
    // Search
    chan=ctxt.createChannel(simpleJCAGet.name);
    // Wait for search
    ctxt.pendIO(simpleJCAGet.timeout);
} catch(Exception ex) {
    System.err.println("Search failed for " +
        simpleJCAGet.name + ":\n" + ex);
    System.exit(1);
}
```java
SimpleJCAGet

// Get the first value as a String
try {
    // Get the value
    String [] value;
    value=((STRING)chan.get(DBRType.STRING,1)).getStringValue();
    // Wait for the get
    ctxt.pendIO(simpleJCAGet.timeout);
    // Print the value
    System.out.println("The value of " + simpleJCAGet.name + " is " + value[0]);
} catch(Exception ex) {
    System.err.println("Get failed for " + simpleJCAGet.name + ":\n" + ex);
    System.exit(1);
}
```
SimpleJCAGet

// Clean up
try {
    // Clear the channel
    chan.destroy();
    // Destroy the context
    ctxt.destroy();
} catch(Exception ex) {
    System.err.println("Clean up failed for " +
        simpleJCAGet.name + "\n" + ex);
    System.exit(1);
}

// Successful exit
System.exit(0);
SimpleJCAGet output

java.exe -classpath
  <simplejca-path>\SimpleJCA.jar;
  <jca-path>\jca-2.1.3.jar
  simplejca.SimpleJCAGet evans:calc
The value of evans:calc is 3
SimpleJCAMonitor

- Similar to SimpleJCAGet
  - Imports, parsing the command line, etc. are the same
- We will have listeners
- We will use JNI_THREAD_SAFE (preemptive)
- We will use flushIO and not pendIO, etc.
SimpleJCAMonitor

/** Implementation of Connection Listener class */
private class SJCAConnectionListener implements ConnectionListener {
    public void connectionChanged(ConnectionEvent ev) {
        onConnectionChanged(ev);
    }
};

/** Implementation of MonitorListener class */
private class SJCAMonitorListener implements MonitorListener {
    public void monitorChanged(MonitorEvent ev) {
        onValueChanged(ev);
    }
};
SimpleJCAMonitor

    // Instance of SimpleJCAMonitor
    SimpleJCAMonitor sjcam = new SimpleJCAMonitor();

    // Initialize JCA
    try {
        // Get the JCALibrary instance
        jca = JCALibrary.getInstance();
        // Create a preemptive context, default configuration
        ctxt = jca createContext(JCALibrary.JNI_THREAD_SAFE);
    } catch (Exception ex) {
        System.err.println("Initialization failed for " +
                           sjcam.name + " :\n" + ex);
        System.exit(1);
    }
SimpleJCAMonitor

```java
// Search
try {
    // Search
    chan=ctxt.createChannel(sjcam.name,
    sjcam.new SJCAConnectionListener());
    ctxt.flushIO();
} catch(Exception ex) {
    System.err.println("Search failed for " +
    sjcam.name + ":\n" + ex);
    System.exit(1);
}
```
```
private void onConnectionChanged(ConnectionEvent ev) {
    Channel ch = (Channel) ev.getSource();
    Context ctxt = ch.getContext();
    // Start a monitor on the first connection
    if (connectionCounter == 0 &&
        ch.getConnectionState() == Channel.CONNECTED) {
        try {
            // Add a monitor listener and flush
            ch.addMonitor(DBRType.STRING, 1,
                          Monitor.VALUE | Monitor.LOG | Monitor.ALARM,
                          new SJCAMonitorListener());
            ctxt.flushIO();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```
SimpleJCAMonitor

// Print connection state
if (ch.getConnectionState() == Channel.CONNECTED) {
    System.out.println(ch.getName() + " is connected");
} else if (ch.getConnectionState() == Channel.CLOSED) {
    System.out.println(ch.getName() + " is closed");
} else if (ch.getConnectionState() == Channel.DISCONNECTED) {
    System.out.println(ch.getName() + " is disconnected");
} else if (ch.getConnectionState() == Channel.NEVER_CONNECTED) {
    System.out.println(ch.getName() + " is never connected");
}
private void onValueChanged(MonitorEvent ev) {
    Channel ch=(Channel)ev.getSource();
    Context ctxt=ch.getContext();
    // Check the status
    if (ev.getStatus() != CAStatus.NORMAL) {
        System.err.println("monitorChanged: Bad status");
    }
    // Get the value from the DBR
    try {
        DBR dbr=ev.getDBR();
        String [] value=((STRING)dbr).getStringValue();
        System.out.print(SJCAUtils.timeStamp() + " " +
                        getName() + ": " + value[0]);
    } catch(Exception ex) {
        ...
    }
}
**Simple JCA Monitor output**

Oct 11, 2004 10:36:43.661 Starting Simple JCA Monitor
Oct 11, 2004 10:36:44.083 Search successful for:
  evans:calc
CHANNEL    : evans:calc
TYPE       : gov.aps.jca.dbr.DBRTyp[DBR_DOUBLE=6]
COUNT      : 1
STATE      :
  gov.aps.jca.Channel$ConnectionState[CONNECTED=2]
HOSTNAME   : ctlapps41188.aps4.anl.gov:5064
READ       : true
WRITE      : true
Oct 11, 2004 10:36:44.208 evans:calc is connected
Oct 11, 2004 10:36:44.224 evans:calc: 2
Oct 11, 2004 10:36:44.224 evans:calc: 3
...          
Oct 11, 2004 10:36:54.036 All Done
JProbe is a simple example that demonstrates using JCA in a Swing GUI.
Write once, run everywhere?
Source files for Simple JCA routines

- All the source and JAR files should be available with the presentation
  - LICENSE
  - SimpleJCA.jar
  - simplejca
    - SimpleJCAMonitor.java
    - SimpleJCAGet.java
    - SJCAUtils.java
  - JProbe.jar
  - jprobe
    - JProbe.java
    - MainFrame.java
    - AboutBoxPanel.java
- Stored as SimpleJCA.zip
Acknowledgements

• JCA was developed by Eric Boucher while at the APS
• Matej Sekoranja [Cosylab] has taken over JCA and is developing CAJ

• Both of these people were very helpful in getting JCA working for me
Thank You

This has been an
APS Controls Presentation
Thank You

This has been an APS Controls Presentation