

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class AbsoluteTimeReply extends Reply {
    long absoluteTime;

    public AbsoluteTimeReply(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public AbsoluteTimeReply(Message m, long absoluteTime) {
        GetAbsoluteTimeCmd origCmd = (GetAbsoluteTimeCmd)m;
        this.seqNo = origCmd.seqNo;
        this.absoluteTime = absoluteTime;

        this.len = 8;
    }

    public void decerealize(DataInputStream dis) throws IOException {
        super.decerealize(dis);
        this.absoluteTime = dis.readLong();
    }

    public int getMessageType(){
        return ABSOLUTE_TIME_REPLY;
    }

    public long getAbsoluteTime() {
        return absoluteTime;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);

        s.writeLong (absoluteTime);
    }

    public String toString(){

        return "<%=msgType=AbsoluteTimeReply%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%absoluteTime="
            + this.absoluteTime
            + "%>";
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class AckReply extends Reply {

    public AckReply(Command c) {
        this.seqNo = c.seqNo;
        this.len = 0;
    }

    public AckReply(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public int getMessageType(){
        return ACK_REPLY;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
    }

    public String toString(){
        return "%<%msgType=AckReply%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%>%";
    }
}
```

06/13/02
20:06:58

common\Command.java

1

```
package nmc.common;

public abstract class Command extends Message {
    private static int nextSeqNo = 1;
    public static synchronized int getNextCmdSeqNo() {
        return nextSeqNo++;
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class ConfigurationCmd extends Command {
    public int channelNumber;
    public int destination;    // one of ConfigurationType
    public byte[] data;

    public ConfigurationCmd(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public ConfigurationCmd(int channelNumber, int destination, byte[] data) {
        this.seqNo = getNextCmdSeqNo();
        this.len = 4+4+data.length; // for the channelNumber
        this.channelNumber = channelNumber;
        this.destination = destination;
        this.data = data;
    }

    public int getMessageType(){
        return CONFIGURATION_CMD;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeInt(channelNumber);
        s.writeInt(destination);
        s.write(data, 0, data.length);
    }

    public void decerealize(DataInputStream s) throws IOException {
        System.out.println("Rockin and rollin");
        super.decerealize(s);
        channelNumber = s.readInt();
        destination = s.readInt();
        data = new byte[this.len-4-4];
        s.read(data, 0, data.length);
    }

    public String toString() {
        return "<%msgType=ConfigurationCmd%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%destination="
            + ConfigurationType.toString(this.destination)
            + "%data="
            + (new String(data))
            + "%>%";
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class ConfigurationReply extends Reply {
    int    channelNumber; // 4
    boolean endOfData;    // 1
    int    subseqNo;      // 4
    long   timestamp;     // 8
    byte[] data;          // varies

    public ConfigurationReply(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public ConfigurationReply(Message m, TimestampedData configResults) {
        ConfigurationCmd origCmd = (ConfigurationCmd)m;
        this.channelNumber = origCmd.channelNumber;
        this.len = 4+1+4+8+configResults.data.length;

        this.seqNo = origCmd.seqNo;
        this.endOfData = true;
        this.subseqNo = 0;
        this.timestamp = configResults.timestamp;
        this.data = configResults.data; // no reason to copy it
    }

    public void decerealize(DataInputStream dis) throws IOException {
        super.decerealize(dis);
        this.channelNumber = dis.readInt();
        this.endOfData = dis.readBoolean();
        this.subseqNo = dis.readInt();
        this.timestamp = dis.readLong();
        this.data = new byte[this.len-4-1-4-8];
        dis.read(data);
    }

    public int getMessageType(){
        return CONFIGURATION_REPLY;
    }

    public byte[] getData() {
        return data;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);

        s.writeInt    (channelNumber);
        s.writeBoolean (endOfData);
        s.writeInt    (subseqNo);
        s.writeLong   (timestamp);

        if (data != null) {
            System.out.println("Cerealizing ConfigurationReply, data.length="
                               + data.length);
        } else {
            System.out.println("Cerealizing ConfigurationReply, data is null");
        }

        s.write(data, 0, data.length);
    }

    public String toString(){

        return "<%msgType=ConfigurationReply%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%endOfData="
            + this.endOfData
            + "%subseqNo="
            + this.subseqNo
            + "%timestamp="
            + this.timestamp
            + "%data="
            + new String(this.data)
            + "%>";
    }
}
```

```
package nmc.common;

// package scope
public class ConfigurationType {
    static final public int DRIVER    = 0;
    static final public int DPA      = 1;
    static final public int INSTRUMENT = 2;
    static public String toString(int i) {
        switch (i) {
            case 0:
                return "DRIVER";
            case 1:
                return "DPA";
            case 2:
                return "INSTRUMENT";
            default:
                throw new IllegalArgumentException("bad value ("
                                                + i
                                                + " for ConfigurationType" );
        }
        // cannot reach here
    }
}
```

```
package nmc.common;

import java.io.IOException;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.lang.Exception;

public class DecerealizingFactory extends Object {

    public static Message parse(byte[] buf) throws IOException {
        Message retval = null;
        ByteArrayInputStream bis= new ByteArrayInputStream(buf);
        DataInputStream dis = new DataInputStream(bis);
        int msgType = dis.readInt();
        switch (msgType) {
            case Message.ACK_REPLY:
                retval = (Message)new AckReply(dis);
                break;
            case Message.POLL_CMD:
                retval = (Message)new PollCmd(dis);
                break;
            case Message.POLLED_DATA_REPLY:
                retval = (Message)new PolledDataReply(dis);
                break;
            case Message.INITIALIZE_INSTRUMENT_CMD:
                retval = (Message)new InitializeInstrumentCmd(dis);
                break;
            case Message.INITIALIZE_INSTRUMENT_REPLY:
                retval = (Message)new InitializeInstrumentReply(dis);
                break;
            case Message.CONFIGURATION_CMD:
                retval = (Message)new ConfigurationCmd(dis);
                break;
            case Message.CONFIGURATION_REPLY:
                retval = (Message)new ConfigurationReply(dis);
                break;
            case Message.WAKEUP_CMD:
                retval = (Message)new WakeupCmd(dis);
                break;
            case Message.GET_ABSOLUTE_TIME_CMD:
                retval = (Message)new GetAbsoluteTimeCmd(dis);
                break;
            case Message.ABSOLUTE_TIME_REPLY:
                retval = (Message)new AbsoluteTimeReply(dis);
                break;
            default:
                int seqno = dis.readInt();
                throw new IOException("DecerealizationFactory:  bad msgType "
                    + "("
                    + msgType
                    + ") "
                    + "from message seqNo="
                    + seqno);
        }
        return retval;
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class GetAbsoluteTimeCmd extends Command {

    public GetAbsoluteTimeCmd(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public GetAbsoluteTimeCmd() {
        this.seqNo = getNextCmdSeqNo();
        this.len = 0; // for the absoluteTime
    }

    public int getMessageType(){
        return GET_ABSOLUTE_TIME_CMD;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
    }

    public void decerealize(DataInputStream s) throws IOException {
        super.decerealize(s);
    }

    public String toString(){
        return "%<%msgType=GetAbsoluteTimeCmd%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%>%";
    }
}
```



```
package nmc.common;

import nmc.platform.*;
import java.io.IOException;

import nmc.platform.DatagramIoPort;

import nmc.common.*;

public class InboundMessageQ implements Runnable {
    private FifoQueue q = new FifoQueue(50);
    private DatagramIoPort dgmRx;

    Thread myThread;

    public InboundMessageQ(int myRxPortNumber) throws IOException {
        dgmRx = DatagramIoPort.GetRcvPort(myRxPortNumber);
        // we won't get here if an IOException is thrown
        myThread = new Thread(this);
    }

    public Message blockingGet() {
        return (Message)q.blockingDequeue();
    }

    public void getOne() throws IOException {
        System.out.println("Inbound blocking on socket read...");
        byte[] rcvdData = dgmRx.blockingRx();
        Message m = DecerealizingFactory.parse(rcvdData);
        q.enqueue(m);
        System.out.println("received: " + m.toString());
    }

    public synchronized void startInboundMessaging() {
        myThread = new Thread(this);
        myThread.start();
    }

    public void run() {
        System.out.println("OutboundMessage thread starting...");
        try {
            while (true) {
                getOne();
            }
        } catch (IOException e) {
            System.out.println("Inbound caught exception, stopping. e="
                               + e);
        }
    }

    public boolean isEmpty() {
        return q.isEmpty();
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class InitializeInstrumentCmd extends Command {

    public int channelNumber;

    public InitializeInstrumentCmd(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public InitializeInstrumentCmd(int channelNumber) {
        this.seqNo = getNextCmdSeqNo();
        this.len = 8;
        this.channelNumber = channelNumber;
    }

    public int getMessageType() {
        return INITIALIZE_INSTRUMENT_CMD;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeInt(channelNumber);
    }

    public void decerealize(DataInputStream s) throws IOException {
        super.decerealize(s);
        channelNumber = s.readInt();
    }

    public String toString(){
        return "<%msgType=InitializeInstrumentCmd%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%>%";
    }
}
```

common\InitializeInstrumentReply.java

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class InitializeInstrumentReply extends Reply {
    int channelNumber;
    boolean endOfData;
    int subseqNo;
    int seconds;

    public InitializeInstrumentReply(Message m, int seconds) {
        InitializeInstrumentCmd iic = (InitializeInstrumentCmd)m;
        this.seqNo = iic.seqNo;
        this.len = 4+1+4+4;
        this.channelNumber = iic.channelNumber;
        this.endOfData = true;
        this.subseqNo = 0;
        this.seconds = seconds;
    }

    public InitializeInstrumentReply(DataInputStream dis) throws IOException {
        this.decerealize(dis);
        decerealize(dis);
    }

    public void decerealize(DataInputStream dis) throws IOException {
        super.decerealize(dis);
        channelNumber = dis.readInt();
        endOfData = dis.readBoolean();
        subseqNo = dis.readInt();
        this.seconds = dis.readInt();
    }

    public int getMessageType(){
        return INITIALIZE_INSTRUMENT_REPLY;
    }

    public int getSeconds() {
        return seconds;
    }

    public byte[] cerealize() throws IOException {
        ByteArrayOutputStream bos= new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        this.cerealize(dos);
        return bos.toByteArray();
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeInt(channelNumber);
        s.writeBoolean(endOfData);
        s.writeInt(subseqNo);
        s.writeInt(seconds);
    }

    public String toString(){
```

```
        return "<%=msgType=InitializeInstrumentReply%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%endOfData="
            + this.endOfData
            + "%subseqNo="
            + this.subseqNo
            + "%seconds="
            + this.seconds
            + ">";
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public abstract class Message extends Object {
    public static final int SET_CURRENT_LIMIT_CMD      = 10; // obsolete
    public static final int ACK_REPLY                  = 11;
    public static final int START_STREAMING_CMD        = 12; // obsolete
    public static final int STOP_STREAMING_CMD         = 13; // obsolete
    public static final int STREAMING_DATA_REPLY       = 14; // obsolete
    public static final int POLL_CMD                   = 15;
    public static final int POLLED_DATA_REPLY          = 16;
    public static final int ERROR_REPLY                = 18; // obsolete
    public static final int ECHO_CMD                   = 19; // ?
    public static final int ECHO_REPLY                 = 20; // ?
    public static final int INITIALIZE_INSTRUMENT_CMD  = 21; // ??
    public static final int INITIALIZE_INSTRUMENT_REPLY = 22; // ??
    public static final int CONFIGURATION_CMD          = 23;
    public static final int CONFIGURATION_REPLY        = 24;
    public static final int WAKEUP_CMD                  = 25;
    public static final int GET_ABSOLUTE_TIME_CMD      = 26;
    public static final int ABSOLUTE_TIME_REPLY        = 27;

    protected int seqNo; // the sequence number
    protected int len;   // length (in bytes) beyond this

    public abstract int getMessageType(); // one of the above types

    // uses the cerealize(DataOutputStream s), which must be
    // implemented by a subclass.
    public byte[] cerealize() throws IOException {
        ByteArrayOutputStream bos= new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(bos);
        this.cerealize(dos);
        return bos.toByteArray();
    }

    protected void cerealize(DataOutputStream s) throws IOException {
        s.writeInt(getMessageType());
        s.writeInt(seqNo);
        s.writeInt(len);
    }

    public void decerealize(DataInputStream s) throws IOException {
        seqNo = s.readInt();
        len = s.readInt();
    }

    public abstract String toString();
}
```

```
package nmc.common;

import nmc.platform.DatagramIoPort;
import nmc.platform.*;
import java.io.IOException;

import java.io.InterruptedIOException;

import nmc.common.*;

public class OutboundMessageQ implements Runnable {
    private Object mutex; // because q can change (later)
    private FifoQueue q = new FifoQueue(50);
    private Thread myThread;
    private String destinationIpAddr;
    private DatagramIoPort dgmTx;
    private int destinationPortNumber;

    public OutboundMessageQ(String destinationIpAddr,
                           int destinationPortNumber)
        throws IOException
    {
        this.dgmTx = DatagramIoPort.GetTxPort(destinationIpAddr,
                                               destinationPortNumber);
        this.destinationIpAddr = destinationIpAddr;
        this.destinationPortNumber = destinationPortNumber;
    }

    public void put(Message msg) {
        q.enqueue(msg);
    }

    public void sendOne() throws IOException {
        Message m = (Message)q.blockingDequeue();
        byte[] msgStr = m.cerealize();
        dgmTx.send(msgStr);
    }

    public synchronized void startOutboundMessaging() {
        myThread = new Thread(this);
        myThread.start();
    }

    public void run() {
        System.out.println("OutboundMessage thread starting...");
        try {
            while (true) {
                sendOne();
            }
        } catch (IOException e) {
            System.out.println("Outbound caught exception, stopping. e="
                               + e);
        }
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class PollCmd extends Command {
    public int channelNumber;

    public PollCmd(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public PollCmd(int channelNumber) {
        this.seqNo = getNextCmdSeqNo();
        this.len = 4; // for the channelNumber
        this.channelNumber = channelNumber;
    }

    public int getMessageType(){
        return POLL_CMD;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeInt(channelNumber);
    }

    public void decerealize(DataInputStream s) throws IOException {
        super.decerealize(s);
        channelNumber = s.readInt();
    }

    public String toString(){
        return "<%msgType=PollCmd%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%>";
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class PolledDataReply extends Reply {
    int    channelNumber;
    boolean endOfData;
    int    subseqNo;
    long   timestamp;
    byte[] data;

    public PolledDataReply(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public PolledDataReply(Message m, TimestampedData dataSample) {
        PollCmd pc = (PollCmd)m;
        this.channelNumber = pc.channelNumber;
        this.len = 4+1+4+8+dataSample.data.length;

        this.seqNo = pc.seqNo;
        this.endOfData = true;
        this.subseqNo = 0;
        this.timestamp = dataSample.timestamp;
        this.data = dataSample.data; // no reason to copy it
    }

    public void decerealize(DataInputStream dis) throws IOException {
        super.decerealize(dis);
        this.channelNumber = dis.readInt();
        this.endOfData = dis.readBoolean();
        this.subseqNo = dis.readInt();
        this.timestamp = dis.readLong();
        this.data = new byte[this.len-4-1-4-8];
        dis.read(data);
    }

    public int getMessageType(){
        return POLLED_DATA_REPLY;
    }

    public byte[] getData() {
        return data;
    }

    public int getChannelNumber() {
        return this.channelNumber;
    }

    public long getTimestamp() {
        return this.timestamp;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);

        s.writeInt    (channelNumber);
        s.writeBoolean (endOfData);
        s.writeInt    (subseqNo);
        s.writeLong   (timestamp);

        if (data != null) {
            System.out.println("Cerealizing PolledDataReply, data.length="
                               + data.length);
        } else {
            System.out.println("Cerealizing PolledDataReply, data is null");
        }
        s.write(data, 0, data.length);
    }

    public String toString(){

        char[] better = new char[data.length];
        for (int i=0; i<data.length; i++) {
            better[i] = (char)data[i];
        }
        String theData = new String(better);

        return "<%msgType=PolledDataReply%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%channelNumber="
            + this.channelNumber
            + "%endOfData="
            + this.endOfData
            + "%subseqNo="
            + this.subseqNo
            + "%timestamp="
            + this.timestamp
            + "%data="
            + theData
            + "%>%";
    }
}
```

06/13/02
20:06:58

common\Reply.java

1

```
package nmc.common;
```

```
public abstract class Reply extends Message {  
}
```



```
package nmc.common;

public class SystemHardwiredInfo {

    public String    sidearmIpAddr;
    public int       sidearmRcvPort;

    public String    sidekickIpAddr;
    public int       sidekickRcvPort;

    public boolean debugging;    // use this to control timeouts, etc.

    private SystemHardwiredInfo(String sidearmIpAddr, int sidearmRcvPort,
                                String sidekickIpAddr, int sidekickRcvPort) {

        this.sidearmIpAddr = sidearmIpAddr;
        this.sidearmRcvPort = sidearmRcvPort;

        this.sidekickIpAddr = sidekickIpAddr;
        this.sidekickRcvPort = sidekickRcvPort;

        this.debugging = true;
    }

    //
    // ADD SETTINGS HERE -- but keep the static fields 'private'
    //
    static private SystemHardwiredInfo Wbaugh =
        new SystemHardwiredInfo("10.0.0.2", 8188, "10.0.0.3", 8190 );

    //
    // CHANGE ONLY THIS
    //
    static public SystemHardwiredInfo Current = Wbaugh;

    /**
     * Returns a String that represents the value of this object.
     * @return a string representation of the receiver
     */
    public String toString() {
        // Insert code to print the receiver here.
        // This implementation forwards the message to super.
        // You may replace or supplement this.
        return "sidearmIpAddress = "
            + sidearmIpAddr
            + ", sidekickRcvPort = "
            + sidekickRcvPort
            + ", sidekickIpAddress = "
            + sidekickIpAddr
            + ", sideKickRcvPort = "
            + sidekickRcvPort
            + ";";
    }
}
```

```
package nmc.common;

public class TimestampedData {
    public long timestamp;
    public byte[] data;
    public TimestampedData(byte[] data) {
        this.data = data;
        this.timestamp = System.currentTimeMillis();
    }
}
```

```
package nmc.common;

import java.io.DataOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class WakeupCmd extends Command {
    public long absoluteTime;

    public WakeupCmd(DataInputStream dis) throws IOException {
        this.decerealize(dis);
    }

    public WakeupCmd(long absoluteTime) {
        this.seqNo = getNextCmdSeqNo();
        this.len = 8; // for the absoluteTime
        this.absoluteTime = absoluteTime;
    }

    public int getMessageType(){
        return WAKEUP_CMD;
    }

    public void cerealize(DataOutputStream s) throws IOException {
        super.cerealize(s);
        s.writeLong(absoluteTime);
    }

    public void decerealize(DataInputStream s) throws IOException {
        super.decerealize(s);
        absoluteTime = s.readLong();
    }

    public String toString(){
        return "<%msgType=WakeupCmd%seqNo="
            + this.seqNo
            + "%len="
            + this.len
            + "%absoluteTime="
            + this.absoluteTime
            + ">%";
    }
}
```