Message Issues

J. Patrick

6/23/99

Java Objects in Messages

- For Java<->Java communication, messages may include Java objects:
  - `msg.appendObject(Object o)`
  - `(MyClass) msg.nextObject()`
- Such a message must be of type OBJECT_DATA and can contain only Java objects
  - Presumably can include multiple ones
  - Recall a TipcMsg can be encapsulated in another
- Note that you still must know (based on subject or something) what types to expect in order to properly cast the result

User Property Field

- Each messsage does contain a 32 bit field the use of which is open to the end user
  - `setUserProp(int)`
  - `int getUserProp()`
- Use for low 32 bits of 64 bit time?

DaqMsg Interface Class

- Simplify packing/unpacking of SmartSockets message to/from Java classes
- Keep Java/C code in synch
- Prototype implementation:
  - Message formats described as Java classes
    - » must implement Serializable
    - » must subclass common base class (“DaqMsg”)
  - Packing/Unpacking done using reflection
    - » **No code generation on Java side**
    - » Requires class name to be first element in messages
      - I can’t think of another way to handle nested messages
  - Generation of analogous C structure file
  - Generation of analogous C packing/unpacking code (not yet implemented)

DaqMsg interface class

```
public abstract class DaqMsg {

    public DaqMsg();

// pack by Java objects
    public TipcMsg packJava();
    public void packJava(TipcMsg);
    public static TipcMsg packJava(DaqMsg);
    public static void packJava(DaqMsg, TipcMsg);

// pack by message fields
    public TipcMsg pack();
    public void TipcMsg pack(TipcMsg);
    public static TipcMsg pack(DaqMsg);
    public static void pack(DaqMsg, TipcMsg);

    public static DaqMsg unpack(TipcMsg);

    public static DaqMsg unpackJava(TipcMsg);

// determine message grammar and register type
    public void register( );
}
```

Example Subclass

```
public class GeneralStatus extends DaqMsg implements Serializable {

// If a field is an object (including array or string) it
// must be constructed in the constructor
// Otherwise an attempt will be made to serialize a null field
// Also, fields must be public for reflection to work across packages

    public int severity;
    public String message = “Hi There”;
    public int[] severitys = new int[2];
    public float[] floats = new float[5];
    public long[] longs = new long[4];
    public short[] shorts = new short[3];

    public SubStatus sub = new SubStatus();

    public static int MESSAGE_TYPE = 1001;

    public GeneralStatus() {
        super();
        messageType = MESSAGE_TYPE; // superclass member variable
    }
}
```

Example Usage

```
public class Test {
    public static void main(String[] argv) {

// Construct a GeneralStatus object, pack it as an object, then unpack
            GeneralStatus gs = new GeneralStatus();
            TipcMsg msg = gs.packJava( );
            GeneralStatus gs2 = (GeneralStatus) DaqMsg.unpackJava(msg);
            gs2.print();
            System.out.println("--------------------");

// Change some fields, reset the TipcMsg,
// pack the modified object by fields, then unpack and print
            gs.message = "Hello There XXXX";
            gs.severity = 453;
            try {
                msg.setNumFields(0);
            } catch (TipcException e) {
                System.out.println("Error " + e.getMessage());
            }
            gs.pack(msg);
            GeneralStatus gs3 = (GeneralStatus) DaqMsg.unpack(msg);
            gs3.print();
    }
```

More details

```
// Callback to unpack either format

public synchronized void process(TipcMsg msg, Object arg) {

        int mt = msg.getType().getNum();
        GeneralStatus gs = null;
        if (mt == TipcMt.OBJECT_DATA) {
            gs = (GeneralStatus) DaqMsg.unpackJava(msg);
            System.out.println("Received object message: ");
        } else {
            gs = (GeneralStatus) DaqMsg.unpack(msg);
            System.out.println("Received field message: " );
        }
        gs.print();
}
```

Current Status

- product daqmsg v4_0 - more creative names welcome
- Currently supports
  - packing/unpacking both via message fields and via Java objects
  - primitive fields
  - arrays of primitive fields
  - Other objects (including strings, arrays of strings/objects)
    - » Note this is handled transparently in Java mode
    - » Handled by nested TipcMsg's in field mode
  - Generation of C header file (to terminal at present)
- Currently lacks
  - Repeated substructures
    - » Recall the last "group" (which can be the entire message) is assumed to be repeated in a TipcMsg
    - » A common scenario might be to get back a variable length array of some structure
  - Generation of C packing/unpacking code
  - time not included in UserProp field
  - comments in the code or any documentation
  - cvs import
  - Undoubtedly everything doesn't work perfectly