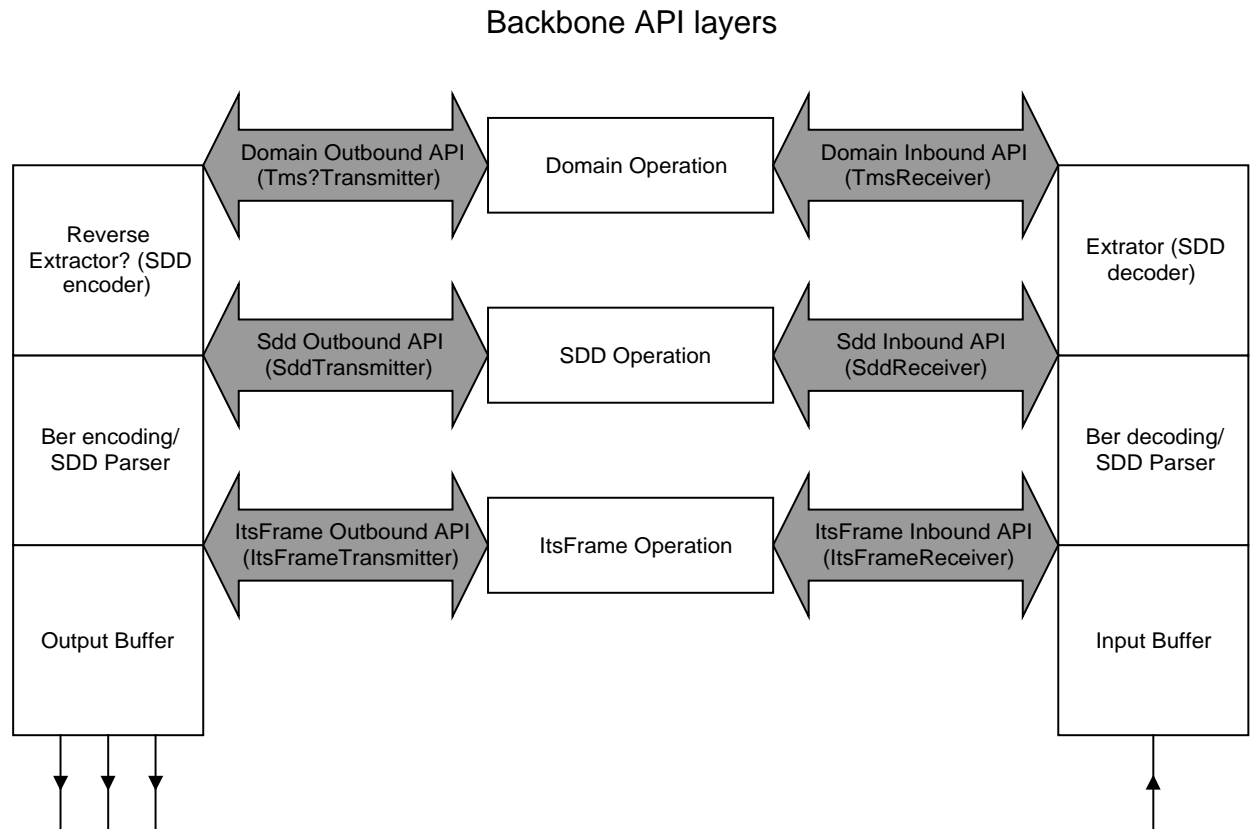


Smart Trek Proposed BackBone API Preliminary Draft v1.0

Table of Contents

- 1.0) API diagram / Introduction
- 2.0) Requirements
- 3.0) Class skeletons
- 4.0) Isp Examples
 - 4.1) Domain Inbound API example
 - 4.2) SDD Inbound API example
 - 4.3) SDD Outbound API example
 - 4.4) ItsFrame Inbound API example
 - 4.5) ItsFrame Outbound API example

1.0) API diagram



The API's that is described is indicated by shaded double arrows in the above diagram. The combined API will consist of a total of five Java classes that can be instantiated. Each layer of the API provides both an inbound and outbound class, except for the domain layer, where there is only a receiver. The inbound classes provide an Event callback scheme to handle incoming data (specific at each layer). The outbound classes provide send methods to transmit outgoing data (specific at each layer). Each class provides a status update callback.

The easiest way to describe the API is to write example code that uses an interface.

2.0) Requirements

Provide a three layer API for Smart Trek data users.

At each layer, provide both receive and transmit capability via Java classes (except for the domain layer).

Hide the layer implementation from the user.

3.0) Class skeletons

```
public class TmsReceiver extends Thread
    implements ContentListener,
               DataListener,
               ExtractorListener,
               SensorListener,
               TmsListener
{
    private SddReceiver sddReceiver;
    private Extractor extractor = new Extractor();
    private boolean stayAlive = true;
    private final static int TYPE_CONTENT = 1;
    private final static int TYPE_DATA = 2;
    private final static int TYPE_EXTRACTOR = 3;
    private Vector incomingVec = new Vector();
    private Vector sensorListenerVec = new Vector();
    private Vector tmsListenerVec = new Vector();

    public TmsReceiver( String host, int port )
        throws java.net.UnknownHostException
    public void contentReceived(ContentEvent event)
    public void dataReceived(DataEvent event)
    public void extractorReceived( ExtractorEvent event )
    private synchronized void mywait()
    private synchronized void mynotify( )
    public void run()
    private boolean checkVec()
    private synchronized void notifySensorListeners( Enumeration enum )
    private synchronized void notifyDataListeners( TmsData[] data )
    public void exit( )
    public Extractor getExtractor()
    public SddReceiver getSddReceiver()
    public synchronized void addSensorListener( SensorListener sensorListener )
```

```

    public synchronized void addTmsListener( TmsListener tmsListener )
    private synchronized void removeListener( Vector v, Object o )
    public void removeSensorListener( SensorListener sensorListener )
    public void removeTmsListener( TmsListener tmsListener )
    public void sensorReceived( SensorEvent event )
    public void tmsReceived( TmsEvent event )
    public void selectSensorAll( )
    public void selectSensorFile( String filename )
    public void selectSensor( String sensor )
    public void generateSQL( PrintStream ps, TmsData[] data )
    public void generateOutput( PrintStream ps, TmsData[] data )
}

```

```

public class SddReceiver extends Thread
    implements ItsFrameListener,
           SchemaListener,
           ContentListener,
           DataListener,
           SerialNumListener,
           ExtractorListener
{
    private ItsFrameReceiver  itsFrameReceiver;
    private Parser parser = new Parser();
    private SddReceiverStatus sddReceiverStatus = new SddReceiverStatus();
    private boolean  parsingEnabled = false;
    private boolean  parsingFailExitEnabled = true;
    private int      serverPort;
    private boolean  stayAlive = true;
    private Vector listenerVec = new Vector();
    private Vector schemaListenerVec = new Vector();
    private Vector contentListenerVec = new Vector();
    private Vector dataListenerVec = new Vector();
    private Vector serialNumListenerVec = new Vector();
    private Vector extractorListenerVec = new Vector();
    public SddReceiver( String serverName, int serverPort )
        throws java.net.UnknownHostException
    public void itsFrameReceived( ItsFrame frame )
    private synchronized void mywait()
    private synchronized void mynotify( )
    public void run()
    private boolean checkListenerVec()
    private synchronized void notifyListeners( ItsFrame frame )
    public void exit()
    public boolean isSqlEnabled( )
    public void enableSql( )
    public void disableSql( )
    public synchronized void addSchemaListener( SchemaListener schemaListener )
    public synchronized void addContentListener( ContentListener contentListener )
    public synchronized void addDataListener( DataListener dataListener )
    public synchronized void addSerialNumListener( SerialNumListener serialNumListener )
    public synchronized void addExtractorListener( ExtractorListener extractorListener )
    private synchronized void removeListener( Vector v, Object o )
    public void removeSchemaListener( SchemaListener schemaListener )
}

```

```

    public void removeContentListener( ContentListener contentListener )
    public void removeDataListener( DataListener dataListener )
    public void removeSerialNumListener( SerialNumListener serialNumListener )
    public void removeExtractorListener( ExtractorListener extractorListener )
    public void schemaReceived( SchemaEvent event )
    public void contentReceived( ContentEvent event )
    public void dataReceived( DataEvent event )
    public void serialNumReceived( SerialNumEvent event )
    public void extractorReceived( ExtractorEvent event )
    public SddReceiverStatus getStatus()
    public ItsFrameReceiver getItsFrameReceiver()
    public boolean isParsingEnabled()
    public void disableParsing()
    public void enableParsing()
    public boolean isParsingFailExitEnabled()
    public void disableParsingFailExit()
    public void enableParsingFailExit()
}

```

```

public class SddTransmitter extends ItsFrameTransmitter
{
    private boolean parsingEnabled = true;
    private boolean parsingExceptionsEnabled = true;
    private byte[] serial_encoded = null;
    public SddTransmitter( int portnum )
        throws java.io.IOException
    public SddTransmitter( String configFile )
        throws java.io.IOException
    removeStatusListener( ItsStatusListener itsStatusListener )
    public void sendDictionary( byte[] schema, byte[] contents )
        throws  BadColumnConstraintException,
                BadColumnNameException,
                BadColumnReferenceException,
                BadColumnValueException,
                BadColumnValueSizeException,
                BadNumberOfColumnsException,
                BadTableNameException,
                java.io.IOException,
                java.lang.Exception
    public void sendBerPacket( byte bertype, byte[] data )
        throws java.io.IOException, BadTypeException
    public void sendData( byte[] data )
        throws java.io.IOException
    private static byte[] byteCombine( byte[] a, byte[] b )
    public ItsStatus getStatus()
    public boolean isParsingEnabled()
    public void disableParsing()
    public void enableParsing()
    public boolean isParsingExceptionsEnabled()
    public void disableParsingExceptions()
    public void enableParsingExceptions()
}

```

```
class SddTransmitter
```

```
{  
    SddTransmitter( int portnum_to_accept_connections );  
    addStatusListener( StatusListener );  
    removeStatusListener( StatusListener );  
    sendDataDictionary( SddSchema, SddContents );  
    sendData( SddData );  
}
```

```
public class ItsFrameReceiver extends Thread  
    implements FrameTarget, ItsFrameListener
```

```
{  
  
    private Runnable    inputBuffer;  
    private int         serverPort;  
    private InetAddress serverAddr;  
    private int sleepTime = 5000;  
    private boolean stayAlive = true;  
    private ItsFrameReceiverStatus itsFrameReceiver = new ItsFrameReceiverStatus();  
    public ItsFrameReceiver( String serverName, int serverPort )  
        throws java.net.UnknownHostException  
    private Vector listenerVec = new Vector();  
    private Vector frameVec = new Vector();  
    public synchronized void add( ItsFrame newFrame )  
    private synchronized void mywait()  
    public void run()  
    private boolean checkFrameVec()  
    private void notifyListeners( ItsFrame frame )  
    public void exit()  
    public void addItsFrameListener( ItsFrameListener listener )  
    public void removeItsFrameListener( ItsFrameListener listener )  
    public void itsFrameReceived( ItsFrame frame )  
    public ItsStatus getStatus()  
}
```

```
public class ItsFrameTransmitter  
    implements ConsoleListener
```

```
{  
  
    private String configFile = null;  
    private DataProperties dataProperties = null;  
    private ItsFrameTransmitterStatus itsFrameTransmitterStatus =  
        new ItsFrameTransmitterStatus();  
    private boolean loggingEnabled = true;  
    private int loggingInterval = 30;  
    private int serverPort = 0;  
    private ServerSocket requestSocket = null;  
    private OutputQueue toSink = null;  
    private Runnable connectionDaemon = null;  
    private ConsoleLoggerSnmp consoleLogger = null;  
    private Thread cdThread = null;  
    private Thread clThread = null;
```

```

        private int totalFramesSent = 0;
        private ThreadGroup clients = null;
        public ItsFrameTransmitter( int port_num, String config_file )
            throws java.io.IOException
        public ItsFrameTransmitter( int port_num )
            throws java.io.IOException
        public ItsFrameTransmitter( String config_file )
        public void consoleUpdate()
        public ItsStatus getStatus()
        public void send( ItsFrame itsFrame )
            throws java.io.IOException
        public void disableLogging()
            throws NoLoggerException
        public void enableLogging()
            throws NoLoggerException
        public void setLoggingInterval( int new_interval )
            throws BadLogIntervalException, NoLoggerException
    }

```

4.0) Examples

4.1) Domain Inbound API example

```

package its.app.backbone;

import java.io.RandomAccessFile;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.io.File;
import java.util.Enumeration;

import its.protocol.frame.*;
import its.protocol.sdd.*;
import its.backbone.frame.*;
import its.backbone.sdd.*;
import its.backbone.domain.*;
import its.backbone.domain.tms.*;

/**
 * Example class to demonstrate the use of the TmsReceiver class.
 * The extracted data is written out to a file (see tmsReceived method).
 *
 * @version rcs id: $Id: TRtester.java,v 1.5 1997/11/26 05:39:13 lawton Exp $
 * @author George Lawton, Battelle
 * @since JDK1.1
 */
class TRtester
    implements SensorListener,
               TmsListener
{

```

```

/**
 * contains the TmsReceiver object
 */
private TmsReceiver tmsReceiver;

/**
 * default sensor output file, will contain a list of all available sensors within data stream
 */
private String SENSOR_FILENAME = "tms.sensors";

/**
 * default sensor selection file (see sensorReceived & tmsReceiver.selectSensorFile(...))
 */
private String SELECT_FILENAME = "tms.select";

/**
 * default data output file
 */
private String TMS_DATA_OUTPUT_FILENAME = "tms.data";

/**
 * contains latest serial number received
 */
private String serial = null;

TRtester( String host, int port )
{
    try {
        tmsReceiver = new TmsReceiver( host, port );
    } catch ( java.net.UnknownHostException e )
    {
        System.err.println( "unknown host: " + host );
        System.exit(1);
    }

    tmsReceiver.getSddReceiver().disableSql();
    tmsReceiver.getSddReceiver().disableParsing();

    tmsReceiver.addSensorListener( this );
    tmsReceiver.addTmsListener( this );
}

public void sensorReceived( SensorEvent event )
{
    System.out.println( "sensor Received" );

    try {
        FileOutputStream file = new FileOutputStream( SENSOR_FILENAME );

        Enumeration sensors = event.getSensorList();

```

```

        while ( sensors.hasMoreElements() )
        {
            file.write( ( ((TmsSensor)sensors.nextElement()).toString() + '\n' )
                .getBytes() );
        }

        file.close();
    } catch ( java.io.IOException e )
    {
        System.err.println( "Data file output, IOException" );
    }

// tmsReceiver.selectSensorAll();

    try {

        tmsReceiver.selectSensorFile( SELECT_FILENAME );

    } catch ( java.io.IOException e )
    {
        System.err.println( "Data file output, IOException" );
    }

}

public void tmsReceived( TmsEvent event )
{

    TmsData[] data = event.getTms();

    System.out.println( "tmsReceived, writing data..." + data.length );

    try {

        FileOutputStream file = new
            FileOutputStream( TMS_DATA_OUTPUT_FILENAME );
        PrintStream ps = new PrintStream( file );

        tmsReceiver.generateOutput( ps, data );

        for (int i = 0 ; i < data.length ; i++)
        {

            if ( data[i] instanceof TmsLoop )
            {
                TmsLoop loop = (TmsLoop) data[i];
                file.write( ( loop.getSensor().getId() + " " +
                    loop.getVolume() + " " +
                    loop.getOccupancy() + '\n' )
                    .getBytes() );

                System.out.println(
                    loop.getSensor().getId() + " " +
                    loop.getVolume() + " " +

```

```

        loop.getOccupancy()
        );
    }
    else if ( data[i] instanceof TmsStation )
    {
        TmsStation station = (TmsStation) data[i];
        file.write( (      station.getSensor().getId() + " " +
                        station. getNLoops() + " " +
                        station.getVolume() + " " +
                        station.getOccupancy() + '\n'
                        ).getBytes() );
    }
    else if ( data[i] instanceof TmsTrap )
    {
        TmsTrap trap = (TmsTrap) data[i];
        file.write( ( trap.getSensor().getId() + " " +
                    trap.getAvgSpeed() + " " +
                    trap.getAvgLength() + '\n'
                    ).getBytes() );
    }
} // end for

file.close();

} catch ( java.io.IOException e )
{
    System.err.println( "Data file output, IOException" );
}

}

/**
 */
public static void main(String[] args)
    throws java.io.IOException, BadTypeException
{
    TRtester trTester = null;

    //
    // accept a port number as a command line param, or use "test.prop" for
    // properties file
    //
    if ( args.length == 2 )
        trTester = new TRtester( args[0], Integer.parseInt(args[1]) );
    else
        trTester = new TRtester( "sdd.its.washington.edu", 9033 );
}

```

```
}
```

4.2) SDD Inbound API example

```
package its.app.backbone;
```

```
import java.io.RandomAccessFile;  
import java.io.FileOutputStream;  
import java.io.File;
```

```
import its.protocol.frame.*;  
import its.protocol.sdd.*;  
import its.backbone.frame.*;  
import its.backbone.sdd.*;
```

```
/**
```

```
 * Example class to demonstrate the use of the SddReceiver class. This is a command  
 * line example program. The program will connect to the SDD server and write out  
 * the schema, content, and data frames out to files. The filename will be generated  
 * based on the received serial number. The serial number is used to tie the schema,  
 * content, and data together.
```

```
 *
```

```
 * @version rcs id: $Id: SRtester.java,v 1.10 1997/11/26 05:39:12 lawton Exp $
```

```
 * @author George Lawton, Battelle
```

```
 * @since JDK1.1
```

```
 */
```

```
class SRtester
```

```
    implements SchemaListener,  
                ContentListener,  
                DataListener,  
                SerialNumListener,  
                ExtractorListener
```

```
{
```

```
    /**
```

```
     * contains SddReceiver object
```

```
     */
```

```
    private SddReceiver sddReceiver;
```

```
    /**
```

```
     * default filename for schema output
```

```
     */
```

```
    private String SCHEMA_FILENAME = "sdd.schema";
```

```

/**
 * default filename for contents output
 */
private String CONTENTS_FILENAME = "sdd.contents";

/**
 * default filename for extractor output
 */
private String EXTRACTOR_FILENAME = "sdd.extractor";

/**
 * default filename for data output
 */
private String DATA_FILENAME = "sdd.data";

/**
 * contains latest serial number received
 */
private String serial = null;

/**
 * constructor
 *
 * @param
 * @returns
 * @throws java.io.IOException
 * @since JDK1.1
 */
SRtester( String host, int port )
{
    try {
        // construct a receiver object with host and port number
        sddReceiver = new SddReceiver( host, port );
    } catch ( java.net.UnknownHostException e )
    {
        System.err.println( "unknown host: " + host );
        System.exit(1);
    }

    // disable parsing and SQL generation
    sddReceiver.disableSql();
    sddReceiver.disableParsing();

    // register callbacks
    sddReceiver.addSchemaListener( this );
    sddReceiver.addContentListener( this );

```

```

        sddReceiver.addDataListener( this );
        sddReceiver.addSerialNumListener( this );
        sddReceiver.addExtractorListener( this );
    }

    /**
     * schema callback
     *
     * @param
     * @returns
     * @throws java.io.IOException
     * @since JDK1.1
     */
    public void schemaReceived( SchemaEvent event )
    {
        System.out.println( "schemaReceived" );

        try {

            // write the schema to a file

            FileOutputStream file = new
                FileOutputStream( ( SCHEMA_FILENAME + "_" + serial ) );
            file.write( event.getSchema() );
            file.close();

        } catch ( java.io.IOException e )
        {
            System.err.println( "Data file output, IOException" );
        }
    }

    /**
     * content callback
     *
     * @param
     * @returns
     * @throws java.io.IOException
     * @since JDK1.1
     */
    public void contentReceived( ContentEvent event )
    {
        System.out.println( "contentReceived" );

        try {

            // write the contents to a file

            FileOutputStream file = new
                FileOutputStream( CONTENTS_FILENAME + "_" + serial );
            file.write( event.getContent() );
            file.close();
        }
    }

```

```

        } catch ( java.io.IOException e )
        {
            System.err.println( "Data file output, IOException" );
        }
    }

/**
 * data callback
 *
 * @param
 * @returns
 * @throws java.io.IOException
 * @since JDK1.1
 */
public void dataReceived( DataEvent event )
{
    System.out.println( "dataReceived" );

    try {

        // write the data to a file

        FileOutputStream file = new
            FileOutputStream( DATA_FILENAME + "_" + serial );
        file.write( event.getData() );
        file.close();

    } catch ( java.io.IOException e )
    {
        System.err.println( "Data file output, IOException" );
    }
}

/**
 * serial number callback
 *
 * @param
 * @returns
 * @throws java.io.IOException
 * @since JDK1.1
 */
public void serialNumReceived( SerialNumEvent event )
{
    System.out.println( "serialNumReceived" );

    // save the serial number

    serial = new String( event.getSerialNum() );
}

```

```

/**
 * extractor callback
 *
 * @param
 * @returns
 * @throws java.io.IOException
 * @since JDK1.1
 */
public void extractorReceived( ExtractorEvent event )
{
    System.out.println( "ExtractorReceived" );

    // just ignore extractor for now...

}

/**
 *
 * @param args command line parameters
 * @throws java.io.IOException error connecting to server
 * @throws BadTypeException error
 * @since JDK1.1
 */
public static void main(String[] args)
    throws java.io.IOException, BadTypeException
{

    SRtester srTester = null;

    //
    // accept a port number as a command line param, or use "test.prop" for
    // properties file
    //
    if ( args.length == 2 )

        // use command line host and port number

        srTester = new SRtester( args[0], Integer.parseInt(args[1]) );

    else

        // use default tms port

        srTester = new SRtester( "sdd.its.washington.edu", 9033 );

}

}

```

4.3) SDD Outbound API example

```
package its.app.backbone;
```

```
import java.io.RandomAccessFile;  
import java.io.File;
```

```
import its.protocol.frame.*;  
import its.protocol.sdd.*;  
import its.backbone.frame.*;  
import its.backbone.sdd.*;
```

```
/**
```

```
 * SddTransmitter tester class. class will transmit four files:
```

```
 * <pre>
```

```
 *   sdd_example_schema
```

```
 *   sdd_example_contents
```

```
 *   sdd_example_data
```

```
 *   sdd_example_extractor
```

```
 * </pre>
```

```
 * the files sdd_example_schema and sdd_example_content contain the data
```

```
 * dictionary for the SDD. The sdd_example_data file contains the raw
```

```
 * data. And, the sdd_example_extractor file contains the extractor java
```

```
 * code (format TBD).
```

```
 *
```

```
 * The class also check for a date changes with the files and re-send them when the
```

```
 * access date changes. Thus, this could be used as a server, with another program
```

```
 * updating the data file. Of course, you would probably want to have a programmatic
```

```
 * interface between the data generator program and this server, for those with timing and
```

```
 * performance in their heads. ;)
```

```
 *
```

```
 * @version rcs id: $Id: STtester.java,v 1.10 1997/11/26 05:37:35 lawton Exp $
```

```
 * @author George Lawton, Battelle
```

```
 * @since JDK1.1
```

```
 */
```

```
class STtester
```

```
{
```

```
    /**
```

```
     * main method that creates an SddTransmitter object, reads in the
```

```
     * files and sends them.
```

```
     *
```

```
     * @param args  command line parameters
```

```
     * @throws java.io.IOException
```

```
     * @throws its.core.itsframe.BadTypeException
```

```
     * @since JDK1.1
```

```
     */
```

```
    public static void main(String[] args)
```

```
        throws java.io.IOException, BadTypeException
```

```
    {
```

```
        /**
```

```
         * contains the SddTransmitter object
```

```
        */
```

```
SddTransmitter sddTransmitter = null;

/**
 * used to read in the files
 */
RandomAccessFile raf;

/**
 * default config file to be used if command line port # is not used.
 * config file contains the port number to use.
 */
String CONFIG_FILE = "ST.prop";

/**
 * schema file
 */
String SCHEMA_FILE = "sdd_example_schema";

/**
 * contents file
 */
String CONTENTS_FILE = "sdd_example_contents";

/**
 * extractor file
 */
String EXTRACTOR_FILE = "sdd_example_extractor";

/**
 * data file
 */
String DATA_FILE = "sdd_example_data";

/**
 * schema File
 */
File schema_file = new File( SCHEMA_FILE );

/**
 * contents File
 */
File contents_file = new File( CONTENTS_FILE );

/**
 * extractor File
 */
File extractor_file = new File( EXTRACTOR_FILE );

/**
 * data File
 */
File data_file = new File( DATA_FILE );

/**
```

```

* last iteration schema file modification time
*/
long last_schema_mod = 0;

/**
* last iteration contents file modification time
*/
long last_contents_mod = 0;

/**
* last iteration extractor file modification time
*/
long last_extractor_mod = 0;

/**
* last iteration data file modification time
*/
long last_data_mod = 0;

/**
* delay between transmissions in millisecs
*/
int delay = 5000;

/**
* data storage location before transmitting
*/
byte[] data = null;

/**
* true if new dictionary detected
*/
boolean new_dictionary = true;

//
// accept a port number as a command line param, or use CONFIG_FILE for
// properties file
//
if ( args.length > 0 )
    sddTransmitter = new SddTransmitter( Integer.parseInt(args[0]) );
else
    sddTransmitter = new SddTransmitter( CONFIG_FILE );

// disable the SQL parsing, this *should* be enabled in production servers!
sddTransmitter.disableParsing();

// loop forever and ever...

while ( true )
{

```

```

//
// Check the last modified dates on the schema,contents, and extractor
// files, if any one has been modified, then resend the entire data
// dictionary ( via setting new_dictionary to true ).
//

long schema_mod      = schema_file.lastModified();
long contents_mod    = contents_file.lastModified();
long extractor_mod   = extractor_file.lastModified();
long data_mod        = data_file.lastModified();

if ( ( last_schema_mod  != schema_mod  ) ||
      ( last_contents_mod != contents_mod ) ||
      ( last_extractor_mod != extractor_mod ) )
{
    last_schema_mod = schema_mod;
    last_contents_mod = contents_mod;
    last_extractor_mod = extractor_mod;
    new_dictionary = true;
}

// if new dictionary detected...

if ( new_dictionary )
{
    new_dictionary = false;

    raf = new RandomAccessFile( SCHEMA_FILE, "r" );
    byte[] schema = new byte[ (int)raf.length() ];
    raf.readFully( schema );

    raf = new RandomAccessFile( CONTENTS_FILE, "r" );
    byte[] contents = new byte[ (int)raf.length() ];
    raf.readFully( contents );

    raf = new RandomAccessFile( EXTRACTOR_FILE, "r" );
    byte[] extractor = new byte[ (int)raf.length() ];
    raf.readFully( extractor );

    try
    {
        {
            sddTransmitter.sendDictionary( schema, contents );
        }
    }
    catch( Exception e )
    {
        {
            System.err.println( "Error sending dictionaries: " + e );
            System.exit( 1 );
        }
    }
    sddTransmitter.sendBerPacket( Ber.TYPE_EXTRACTOR, extractor );
}

//
// if data file has been modified, reread the data file

```

```

//
if ( last_data_mod != data_mod )
{
    last_data_mod = data_mod;

    raf = new RandomAccessFile( DATA_FILE, "r" );
    data = new byte[ (int)raf.length() ];
    raf.readFully( data );

    System.out.println( "re-reading data..." );

}

sddTransmitter.sendData( data );

//
// delay for delay millisecs
//

try {
    Thread.sleep( delay );
} catch ( Exception e )
{}

}

}

}

```

4.4) ItsFrame Inbound API example

```
package its.app.backbone;
```

```
import its.protocol.frame.*;
import its.protocol.sdd.*;
import its.backbone.frame.*;
import its.backbone.sdd.*;
```

```
/**
```

```

* Example class to demonstrate the use of the ItsFrameReceiver class.
* Simply connects to a ItsFrame server and prints out the frames to System.out
*
* @version rcs id: $Id: IFRtester.java,v 1.7 1997/11/26 03:52:22 lawton Exp $
* @author George Lawton, Battelle
* @see its.backbone.frame.ItsFrameReceiver
* @since JDK1.1

```

```

*/
class IFRtester
    implements ItsFrameListener
{

    /**
     * ItsFrame received callback, is called when a ItsFrame is received
     *
     * @param    frame    contains the ItsFrame received from the server
     * @since    JDK1.1
     */
    public void itsFrameReceived( ItsFrame frame )
    {

        // print data received

        System.out.println( "inside of itsFrameReceived: " + new String( frame.getData() ) );
    }

    /**
     * constructor
     *
     * @param    host    contains the ItsFrame server hostname
     * @param    port    contains the ItsFrame server port number
     * @since    JDK1.1
     */
    IFRtester( String host, String port )
        throws java.net.UnknownHostException, Exception
    {

        // construct an ItsFrameReceiver object, sending in host and port

        ItsFrameReceiver itsFrameReceiver =
            new ItsFrameReceiver( host, Integer.parseInt( port ) );

        // register callback

        itsFrameReceiver.addItsFrameListener( this );
    }

    /**
     * main method, will just instantiate an IFRtester object
     *
     * @param    args    contains the command line args
     * @since    JDK1.1
     */
    public static void main(String[] args)
        throws java.io.IOException, BadTypeException, Exception
    {

        // check for proper usage
    }
}

```

```

        if ( args.length == 2 )
        {

                // create instance of IFRtester with two cmd line args

                IFRtester ifr = new IFRtester( args[0], args[1] );

        }
        else
        {

                // print usage and exit

                System.out.println( "usage: host port" );
                System.exit(1);

        }

}
}

```

4.5) ItsFrame Outbound API example

```
package its.app.backbone;
```

```
import its.protocol.frame.*;
import its.protocol.sdd.*;
import its.backbone.frame.*;
import its.backbone.sdd.*;
```

```

/**
 * Example class to demonstrate the use of the ItsFrameTransmitter class.
 * Simply sends the String "This is a test\n\n" to the ItsFrame port. Can be used
 * as a test ItsFrame server, or you could read something interesting from a file
 * and send it down the pipe.
 *
 * @version rcs id: $Id: IFTtester.java,v 1.8 1997/11/26 03:52:21 lawton Exp $
 * @author George Lawton, Battelle
 * @since JDK1.1
 */
class IFTtester
{

        /**
         * main method that creates an ItsFrameTransmitter object and sends
         * a dummy ascii string.
         *
         * @param args command line parameters

```

```

* @throws java.io.IOException
* @throws its.core.itsframe.BadTypeException
* @since JDK1.1
*/
public static void main(String[] args)
    throws java.io.IOException, BadTypeException
{

    ItsFrameTransmitter itsFrameTransmitter = null;

    if ( args.length == 1 )
    {
        //
        // create a new ItsFrameTransmitter with the port number given
        // in the command line
        //
        itsFrameTransmitter =
            new ItsFrameTransmitter( Integer.parseInt(args[0]),
                                    "IFT.prop" );
    }
    else
    {
        //
        // create a new ItsFrameTransmitter with the port number from
        // the config/properties file "IFT.prop"
        //
        itsFrameTransmitter = new ItsFrameTransmitter( "IFT.prop" );
    }

    while ( true )
    {

        //
        // create a new ItsFrame with some test ascii data
        //
        ItsFrame myframe = new ItsFrame( ItsFrameType.DICTIONARY,
                                        "This is a test\n\n".getBytes() );

        //
        // send the ItsFrame to the transmitter
        //
        itsFrameTransmitter.send( myframe );

        //
        // goto sleep for a lil' while
        //
        try
        {
            Thread.sleep( 5000 );
        }
        catch( Exception e )
        {
        }
    }
}

```

}
}
}