



Self Describing and Self Extracting Data Flows (SDD/SED)

Smart Trek

*Intelligent Transportation for
the Puget Sound Region*

**Client User Information
Bulletin**

**A User Introduction to the Smart Trek Data Transfer
Methodology**

for the

Metropolitan Model Deployment Initiative Project

of the

**Washington State Department of Transportation
Regional and Local Agencies
Commercial and Academic Partners**

and the

U S Department of Transportation

Version 1.0
August 29, 1997

Table of Contents

1.	Introduction	3
1.1	Data Dictionary	3
1.1.1	Schema	3
1.1.2	Contents	4
1.2	The Data	4
1.2.1	Continuous Data Flow.....	4
1.2.2	Change to Number of Data Structures	5
1.2.3	Changes to Structure or Elements Contained.....	5
1.2.4	SDD Change Management.....	5
2.	Client Use of the SDD Interface	6
2.1	Step 1 - Connection to SDD Source	6
2.2	Step 2 – Receipt of Data Dictionary	6
2.3	Step 3 – Receipt of Data	7
2.3.1	Client Developer Working in Other Than Java (The Hard Way)	7
2.3.2	Client Developer Working in Java (The Easy Way).....	7
2.3.2.1	Schema/Contents SQL to File.....	8
2.3.2.2	Data Extraction to ASCII File	8
2.3.2.3	Raw Binary Data.....	8
2.3.2.4	Data Extraction as SQL-92.....	8

Figures and Illustrations

Figure 1	SDD/SED Data Stream Overview - Components	3
Figure 2	SDD/SED Data Component	4
Figure 3	SDD Receiver and Extractor Overview.....	7

Tables

Table 1	Identification of SDD Data Flows and Port Locations	6
---------	---	---

1. Introduction

The Self-Describing Data (SDD) format conceived for use in the Smart Trek Metropolitan Model Deployment Initiative (MMDI) is a data transfer mechanism with the means to both describe and package a dynamic data stream. The SDD stream contains, in order of receipt, both a data dictionary and the raw data as shown in Figure 1.

1.1 Data Dictionary

The SDD data dictionary portion is further broken into two parts: (1) the Schema and (2) the Contents. Both the schema and the contents components are sent as ITS Frames containing two BER encoded ASCII packets and use a subset of entry-level SQL-92 compliant language. The SDD/SED process acquires this knowledge of the schema and contents at the source of the dynamic data stream. Depending on the implementation for the specific interface, the SDD/SED process point-of-presence (POP) might acquire this metadata via a “trusted interface” to the source application’s database engine, or the POP may acquire this metadata from static files prepared by the source data provider. Other POP arrangements are possible.

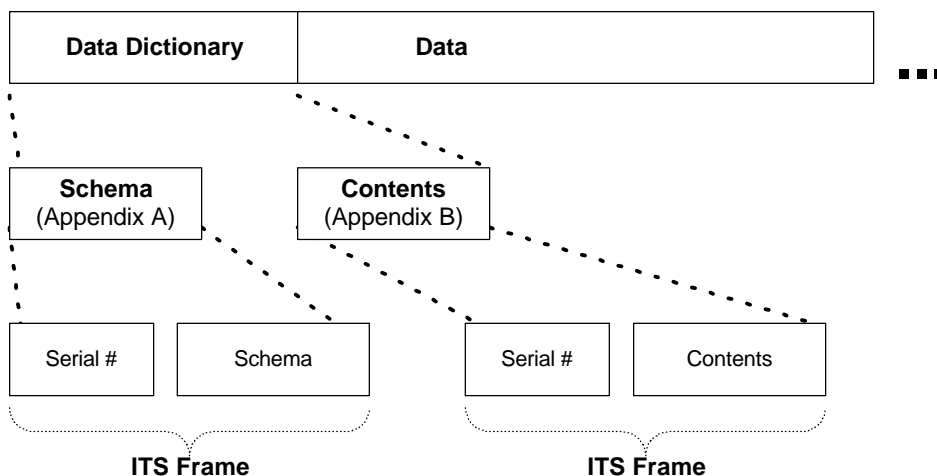


Figure 1 SDD/SED Data Stream Overview - Components

1.1.1 Schema

The first component part of the SDD data definition is the Schema Definition which is comprised of two BER packets. The first packet contains a serial number (YYYYMMDDHHMMSSSS) which indicates the version of the schema to follow. The second packet contains the ASCII command lines that define the table structure of the schema contents. A schema definition will contain a “CREATE SCHEMA” statement and at least one “CREATE TABLE name” statement followed by the table name(s) and the table structure(s). The table structure(s) will consist of one or more variable declarations and any variable constraints (e.g., NOT NULL, PRIMARY KEY, etc.). The content of a typical schema component for freeway loops is illustrated by the example in Appendix A.

1.1.2 Contents

The second component part of the SDD data definition is the Schema Contents which is also comprised of two BER packets. As with the schema, the first contents packet contains a serial number (YYYYMMDDHHMMSSSS) which indicates the version of the contents to follow. The second packet contains the data that can be used to populate the table(s) defined in the schema definition. This table data is ordered by specifying the table name, then the variable names, or columns, as declared within the schema definition. Following this is an array of row entries within the table. The content of a typical freeway loops “contents” component is illustrated by the example in Appendix B.

1.2 The Data

The actual data content of the SDD stream follows the data dictionary. The dynamic data stream will contain a timestamp (YYYYMMDDHHMMSSSS) followed by raw binary data consisting of an array of structures in network byte order as shown in Figure 2. One of the primary benefits of SDD is that the process can accommodate a continuous data flow, as well as the dynamic aspects of the data stream related to: (1) changes in the number or type of data structures present in the array, or (2) changes in the actual structure of tables, variables or relationships defined in the schema.

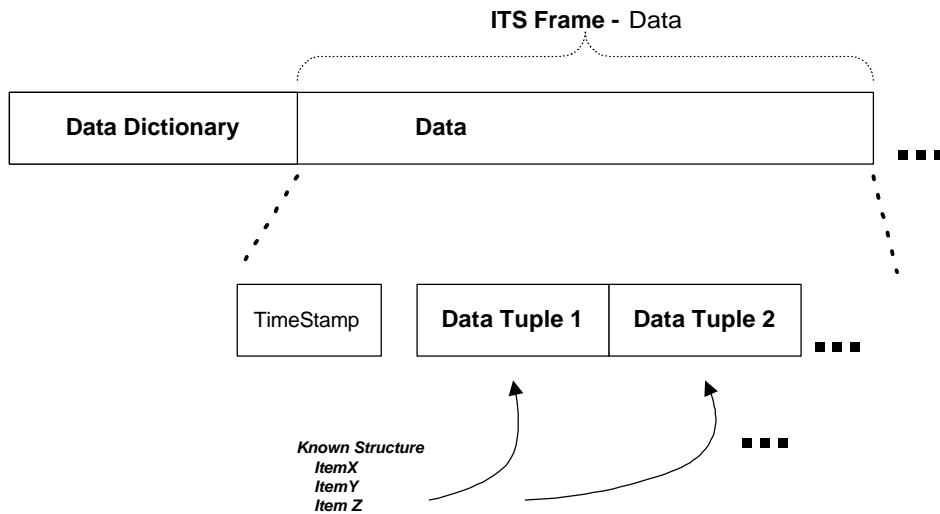


Figure 2 SDD/SED Data Component

1.2.1 Continuous Data Flow

The preferred alternative for data receipt across an interface is that once a client-server connection is made, the data stream will flow continuously without change in schema or structure, and with minimal concern for synchronization and connection management. Once the client application connects to the SDD/SED server and receives one iteration of the data dictionary, that is exactly what happens using SDD/SED. The “data” flow of 1’s and 0’s continues “forever” until

the client or the server disconnects. However, if the SDD/SED server detects a change in either “contents” or “schema”, then the following discussion applies.

1.2.2 Change to Number of Data Structures

The first or lowest level of change in the structure of a dynamic data stream would be a change in the number of data structures contained in the data array. This change can be detected by receipt of a new data dictionary contents component. As illustrated by the example in Appendix B, if the number of freeway loops changed (i.e., increased or decreased in number), the “contents” component for the “TABLE LOOPS” would reflect that same change. The SDD/SED process provides SDD Receiver tools that enable this change to be adapted with or without the client application’s knowledge, and with minimum or no effect on the application’s processing of the dynamic data stream.

1.2.3 Changes to Structure or Elements Contained

The highest level of change in the structure of a dynamic data stream would be a change in the schema. That is, a change in one or more of the tables that are defined, or a change in one or more of the variables, attributes or relationships contained in those tables. This change can be detected by receipt of new data dictionary schema and contents components. The new contents is required to provide the initial data to populate the tables and variables contained in the new definition of the schema. As above, the SDD/SED process provides SDD Receiver tools that enable this change to be adapted with or without the client application’s knowledge, and with minimum or no effect on the application’s processing of the dynamic data stream.

1.2.4 SDD Change Management

The goal of using SDD/SED is to minimize or eliminate the effect of change on the deployed operational applications. There will be rules that govern the mechanism of change to achieve this goal in all cases where possible. These rules are (TBD).

[This section will explain how a variety of different changes will be implemented to avoid or minimize the impact on deployed applications. For example, if it were necessary to change the units of a variable from MPH to KPH, the change process should probably be: (1) retain the existing variable “speed” as MPH, (2) add a new variable “speed_kph” as the container for KPH, (3) alter the schema for the affected table(s). Deployed applications expecting a variable named “speed” in MPH will continue to function as designed.]

As mentioned above, the SDD Receiver tools can protect the application from change and allow it to continue processing the data stream. But the application must be “well behaved” to benefit from these protective features. The rules for a “well behaved” application are (TBD).

[In general, an application is “well behaved” if it uses the SDD/SED service options to receive the data stream rather than processing the raw data directly. It is obvious that if the structure of the raw data changes, applications that process it directly will fail, or at the very least, give erroneous results.]

2. Client Use of the SDD Interface

This section presents the basic procedure for use of the SDD/SED interface. This use is described as the normal sequence of steps to be followed by a typical client application.

2.1 Step 1 - Connection to SDD Source

The initiation of an SDD/SED data transfer consists of making a TCP/IP connection to the SDD Server port for the specific data stream. These data streams and the associated ports are enumerated in Table 1. This table includes remarks which describe the characteristics of the particular data stream (e.g., its reporting cycle, etc.).

Table 1 Identification of SDD Data Flows and Port Locations

Data Flow Identifier	Domain Name or IP Address	Port	Remarks
freeway loops data	sdd.its.washington.edu	9033	Loops data reporting cycle is every 20 seconds
<i>metro transit avl</i>			
<i>regional incident information</i>			
<i>regional arterial status</i>			

Depending on the specific operational need, the client application may select any of several procedural options for connection management. The three basic SDD/SED connection options include:

- Connect once and stay connected (e.g., connect at 0530, disconnect at 0100)
- Connect when needed to refresh databases then disconnect (e.g., every hour, every 15 minutes, etc.)
- Connect on event to refresh databases then disconnect (e.g., upon operator query, etc.)

2.2 Step 2 – Receipt of Data Dictionary

An SDD Receiver application will be provided to developers of client applications for use within the Smart Trek partnership. The Java source/executable file(s) for the SDD Receiver are available by FTP from the Smart Trek SDD/SED Distribution site at “ftp://ftp.its.washington.edu/pub/mdi/SDD”. The version number and status of the SDD Receiver file(s) is clearly indicated by the file naming convention. The SDD Receiver was written in Java and contains both a Schema/Contents parser and a tailorable extractor as shown in Figure 3.

The schema and contents parser will verify that the schema and contents data types match and that they are SQL compliant. The parser will also verify the variable types of the schema content versus the variable type declarations within the schema definition.

The parser will create an SQL command file of the same name each time a data dictionary is received; if it exists from a prior data dictionary, it is overwritten. This schema command file can be used with an SQL compliant commercial database system to build a client database and populate the resulting tables (see Figure 3, Appendices A and B). If the client application wanted to detect a change in schema or contents, it could use the existence of this SQL command file as an indication. For example, the client application could delete the command file after it is used to create and populate the database schema; if it exists subsequently, then a new data dictionary has been received.

2.3 Step 3 – Receipt of Data

As mentioned above, an SDD Receiver application will be provided to developers of client applications. This SDD Receiver contains a tailorable SDD extractor as shown in Figure 3. The extractor will provide the client application the mechanisms to access the data stream as discussed in the following sections.

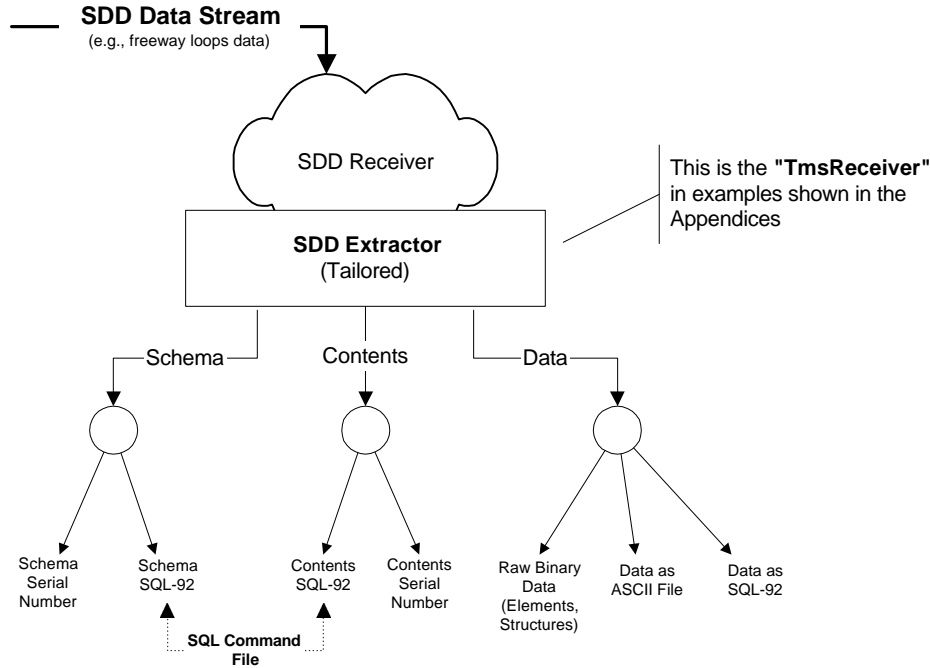


Figure 3 SDD Receiver and Extractor Overview

2.3.1 Client Developer Working in Other Than Java (The Hard Way)

The client application developer has the option of working with the data stream using C/C++ structures or any equivalent technique. Processing in C/C++ or other non-Java language requires an overall knowledge of how the raw data is structured—this is illustrated by the embedded source code and comments found in the table named “ALG_DESCRIPT” in the data dictionary “contents” component (see Appendix B). This text table also contains sample extraction algorithms written in Java which could be used as models for development of equivalent functionality in another language. A client application developer working in C/C++ will also need to embed the University of Washington header files to successfully compile their application--these are available upon request. Also see Appendix B for an example of how the necessary data structure information can be obtained from the “contents”.

While there may be reasons that a client developer would elect this approach, the resulting application would of course be insensitive to the dynamic aspects of the schema and contents and is likely to fail or produce erroneous results if and when change occurs. The better approach is to develop using Java and take full advantage of the SDD Receiver/Extractor tools available in the distribution modules at “ftp://ftp.its.washington.edu/pub/mdi/SDD”, and the source code and algorithms provided in the “ALG_DESCRIPT” table.

2.3.2 Client Developer Working in Java (The Easy Way)

If the client developer is working in Java, then the full features and functionality of the SDD Receiver and tailorable Extractor are available for use in the construction of a very robust application. Without getting into too many technical details, the client application can take advantage of the following features:

- Receipt and parsing of the data dictionary schema and contents; placement of the resulting SQL statements in a file
- Receipt and extraction of the data stream in one of three ways:
 1. Extraction of all data and placement in an ASCII file
 2. Extraction of selected data by name and returned as raw binary data
 3. Extraction of all data as SQL-92

More complete technical information is contained in a developer's "Read Me" file in Appendix D. This file explains the process in more technical detail and provides details on the current release and process for reporting and resolving problems. In cases where there may be disagreement between the contents, explanations or examples of this document and the current version of the "Read Me" file, the Read Me file shall be used as the most current definitive source.

2.3.2.1 Schema/Contents SQL to File

The SDD Receiver parses and presents the schema and contents in an ASCII file named "sdd.sql". The client application can use this file as mentioned above to create a database, and populate the tables in that database. If it exists, this file is overwritten each time a new data dictionary is received.

2.3.2.2 Data Extraction to ASCII File

The SDD Receiver will extract the complete data set from the data stream and present in an ASCII file named "sdd.textdata". A prototype example of this option is shown as Method 1 in Appendix C. The Java source application is named "TmsReceiver.java". An example of what this file would contain for the freeway loops data stream follows the sample source code file in Appendix C. If it exists, this file is overwritten each time a new data component stream is received.

2.3.2.3 Raw Binary Data

This option refers to the receipt and presentation of the data stream as raw data located in memory. In general, the SDD Extractor returns a data pointer which can be used by the client application to locate and process the data. A prototype example of this option is shown as Method 2 in Appendix C.

In this example, the SDD Receiver uses an ASCII file for input as the selection criteria for the specific data items to be extracted from the data stream. The Java source application is named "TmsSampleFile.java" and the data specification for this example is contained in a file named "sensor.txt". An example of what the selection file might contain for the freeway loops data stream follows the sample source code file in Appendix C. The data returned from the use of this approach is as shown in the example code and can be tailored to the specific application need.

2.3.2.4 Data Extraction as SQL-92

The SDD Receiver will extract the complete data set from the data stream and present it in an SQL command file named "sdd.sqldata". This command file can then be used to populate the defined data tables. For freeway loops data, this SQL command file would contain the data to be used to populate the tables "LOOP_DATA", "STATION_DATA" and "SPEED_TRAP_DATA" as shown in the schema example, Appendix A. If it exists, this file is overwritten each time a new data component stream is received.

Appendix A

Example Schema for Freeway Loops Data Stream

This is an example of what would be received as the “Schema” portion of the SDD Data Dictionary for the freeway loops data stream. As illustrated, it contains SQL-92 statements which can (optionally) be used by the client to construct a database suitable for error checking, validation and processing of the loops data stream. The file used for this example is “sdd.schema_19970826020526818” and was obtained by FTP from the Smart Trek SDD/SED Distribution site at “[\(ftp://\(TBD\)\)](ftp://(TBD))”. The date-time group filename suffix establishes the version of this “schema” data.

```
-- File TMS_LOOPS.ddl
-- Author: David Meyers
-- Last Modified: 22 July 1997
-- Purpose: Modified from TDAD.ddl to better deal with TMS loops data for
--          Self Describing Data.
```

```
-----
CREATE SCHEMA -- recipient will need to supply AUTHORIZATION information
```

```
CREATE TABLE CABINETS
(CABINET_ID CHAR(7) NOT NULL PRIMARY KEY,
 FREEWAY CHAR(10) NOT NULL,
 TEXT CHAR(255),
 RAMP SMALLINT NOT NULL)
```

```
CREATE TABLE LOOPS
(LOOP_ID CHAR(16) NOT NULL PRIMARY KEY,
 CABINET_ID CHAR(7) NOT NULL,
 METERED CHAR(4) NOT NULL,
 RD_TYPE CHAR(30) NOT NULL,
 DIRECTION CHAR(12) NOT NULL,
 LANE_TYPE CHAR(20) NOT NULL,
 LANE_NUM SMALLINT,
 SENSOR_TYPE CHAR(12) NOT NULL,
 DATA_OFFSET INTEGER NOT NULL)
```

```
CREATE TABLE COORDINATES
(COORD_TYPE CHAR(40) NOT NULL PRIMARY KEY,
 NAME1 CHAR(40) NOT NULL,
 NAME2 CHAR(40) NOT NULL,
 NAME3 CHAR(40) NOT NULL,
 UNIT1 CHAR(40) NOT NULL,
 UNIT2 CHAR(40) NOT NULL,
 UNIT3 CHAR(40) NOT NULL)
```

```
CREATE TABLE CABINET_LOCATION
(CABINET_ID CHAR(7) NOT NULL,
 COORD_TYPE CHAR(40) NOT NULL,
 AUTHORITY CHAR(30) NOT NULL,
 VALUE1 DEC(11,8),
 VALUE2 DEC(11,8),
 VALUE3 DEC(11,8),
 PRIMARY KEY (CABINET_ID, COORD_TYPE, AUTHORITY),
 FOREIGN KEY (COORD_TYPE, AUTHORITY) REFERENCES MEASURES)
```

```
CREATE TABLE MEASURES
(COORD_TYPE CHAR(40) NOT NULL,
 AUTHORITY CHAR(30) NOT NULL,
 REF_SYSTEM CHAR(128) NOT NULL,
 REF_PT1 DEC(11,8),
 REF_PT2 DEC(11,8),
 REF_PT3 DEC(11,8),
 ACCURACY1 DEC(11,8),
```

```

ACCURACY2 DEC(11,8),
ACCURACY3 DEC(11,8),
PRIMARY KEY (COORD_TYPE, AUTHORITY),
FOREIGN KEY (COORD_TYPE) REFERENCES COORDINATES)

CREATE TABLE LOOP_DATA
(SENSOR_ID CHAR(15) NOT NULL,
DATA_TIME CHAR(20) NOT NULL,
VOLUME SMALLINT NOT NULL,
SCAN_COUNT SMALLINT NOT NULL,
FLAG SMALLINT NOT NULL,
LANE_COUNT SMALLINT NOT NULL,
INCIDENT_DETECT SMALLINT NOT NULL,
PRIMARY KEY (SENSOR_ID, DATA_TIME))

CREATE TABLE STATION_DATA
(SENSOR_ID CHAR(15) NOT NULL,
DATA_TIME CHAR(20) NOT NULL,
VOLUME SMALLINT NOT NULL,
SCAN_COUNT SMALLINT NOT NULL,
FLAG SMALLINT NOT NULL,
LANE_COUNT SMALLINT NOT NULL,
INCIDENT_DETECT SMALLINT NOT NULL,
PRIMARY KEY (SENSOR_ID, DATA_TIME))

CREATE TABLE SPEED_TRAP_DATA
(SENSOR_ID CHAR(15) NOT NULL,
DATA_TIME CHAR(20) NOT NULL,
SPEED SMALLINT NOT NULL, -- or SPEED FLOAT NOT_NULL
LENGTH SMALLINT NOT NULL, -- or LENGTH FLOAT NOT_NULL
FLAG SMALLINT NOT NULL,
BIN1 SMALLINT NOT NULL,
BIN2 SMALLINT NOT NULL,
BIN3 SMALLINT NOT NULL,
BIN4 SMALLINT NOT NULL,
PRIMARY KEY (SENSOR_ID, DATA_TIME))

CREATE TABLE LOOP_FLAGS
(FLAG_VAL SMALLINT NOT NULL,
EXPLANATION CHAR(24) NOT NULL,
PRIMARY KEY (FLAG_VAL))

CREATE TABLE STATION_FLAGS
(FLAG_VAL SMALLINT NOT NULL,
EXPLANATION CHAR(24) NOT NULL,
PRIMARY KEY (FLAG_VAL))

CREATE TABLE INCIDENT_DETECT
(FLAG_VAL SMALLINT NOT NULL,
EXPLANATION CHAR(24) NOT NULL,
PRIMARY KEY (FLAG_VAL))

-- To deal with lack of a "text" data type in SQL, we use the construct
-- that follows.
-- The lines are stored in a table, keyed by line number and
-- measure name. A bit of a pain, but it should be pretty flexible.
-- To get the text for a given algorithm, select all lines with the
-- desired FILE_NAME, in LINE_NUMBER order.

CREATE TABLE ALG_DESCRIPT -- Algorithms Used To Extract Data From
-- The Real Data Stream.
(FILE_NAME CHAR(16) NOT NULL,
LINE_NUMBER INTEGER NOT NULL,
LINE_TEXT CHAR(120),
PRIMARY KEY (FILE_NAME, LINE_NUMBER))

```

```
;
```

Appendix B

Example Contents for Freeway Loops Data Stream

This is an example of what would be received as the “Contents” portion of the SDD Data Dictionary for the freeway loops data stream. The Java source code table (TABLE ALG_DESCRIPT) is provided in its entirety, repetitive entries in other tables have been removed for compactness. The entire file used for this example is “sdd.schema_19970826020526818” and was obtained by FTP from the Smart Trek SDD/SED Distribution site at “[\(TBD\)](ftp://(TBD))”. The date-time group filename suffix establishes the version of this “contents” data.

Discussion for a Client Developer Working in Other Than Java. The information on data structure and location is contained in the “contents” component of the data dictionary. Examine the contents at the following locations in table ALG_DESCRIPT for examples of how this is specified for the freeway loops data stream:

- lines “TmsLoop.java”, 335-369 illustrate the structure of a volume/occupancy loop data packet
- lines “TmsStation.java”, 286-316 illustrate the structure of a loop station data packet

Then examine the contents for table LOOPS including the variable “DATA_OFFSET” which is the byte offset pointer to the start of each type of loops data packet in the data stream. Using this information, a developer could process the raw data stream in other than Java.

```
TABLE ALG_DESCRIPT
COLUMN (FILE_NAME, LINE_NUMBER, LINE_TEXT)
'TmsLoop.java', 0, '' ;
'TmsLoop.java', 1, '' ;
'TmsLoop.java', 2, '/*' ;
'TmsLoop.java', 3, ' * Encapsulates a TMS loop sensor in packed format, plus other fields' ;
'TmsLoop.java', 4, ' * necessary to support SDD. This is an immutable class which may' ;
'TmsLoop.java', 5, ' * only be initialized from a TMS data block.' ;
'TmsLoop.java', 6, ' *' ;
'TmsLoop.java', 7, ' * <P>Should this be extended to include (a) a ref to the TmsSensor, (b)' ;
'TmsLoop.java', 8, ' * the sensor ID, (c) the timestamp, (d) a ref to the TmsData? Or should' ;
'TmsLoop.java', 9, ' * there maybe be another type which includes all this stuff through' ;
'TmsLoop.java', 10, ' * aggregation?' ;
'TmsLoop.java', 11, ' */' ;
'TmsLoop.java', 12, '' ;
'TmsLoop.java', 13, 'public class TmsLoop extends TmsDataItem' ;
'TmsLoop.java', 14, '{' ;
'TmsLoop.java', 15, ' /*' ;
'TmsLoop.java', 16, ' * Constructs a TmsLoop object using the 3 bytes of data at' ;
'TmsLoop.java', 17, ' * <CODE>offset</CODE> in <CODE>dataBlock</CODE>.' ;
'TmsLoop.java', 18, ' *' ;
'TmsLoop.java', 19, ' * @param dataBlock byte array containing data' ;
'TmsLoop.java', 20, ' * @param offset offset within <CODE>dataBlock</CODE>' ;
'TmsLoop.java', 21, ' * @param sensor reference to sensor for this data' ;
'TmsLoop.java', 22, ' *' ;
'TmsLoop.java', 23, ' * @exception BadOffsetException if <CODE>offset</CODE> is less than' ;
'TmsLoop.java', 24, ' * the header size or greater than the' ;
'TmsLoop.java', 25, ' * data block size' ;
'TmsLoop.java', 26, ' */' ;
'TmsLoop.java', 27, ' public TmsLoop(TmsDataBlock dataBlock, int offset, TmsSensor sensor)' ;
'TmsLoop.java', 28, ' throws BadOffsetException' ;
'TmsLoop.java', 29, '{' ;
'TmsLoop.java', 30, ' stringRep = null;' ;
'TmsLoop.java', 31, ' this.sensor = sensor;' ;
'TmsLoop.java', 32, ' this.timestamp = dataBlock.getTimestamp();' ;
'TmsLoop.java', 33, ' read(dataBlock, offset);' ;
'TmsLoop.java', 34, ' }' ;
'TmsLoop.java', 35, '' ;
'TmsLoop.java', 36, ' /*' ;
'TmsLoop.java', 37, ' * reads values for a TmsLoop object from the 3 bytes of data at' ;
'TmsLoop.java', 38, ' * <CODE>offset</CODE> in <CODE>dataBlock</CODE>.' ;
'TmsLoop.java', 39, ' *' ;
'TmsLoop.java', 40, ' * @param dataBlock byte array containing data' ;
'TmsLoop.java', 41, ' * @param offset offset within <CODE>dataBlock</CODE>' ;
```

```

'TmsLoop.java', 42, ' * ' ;
'TmsLoop.java', 43, ' * @exception BadOffsetException if <CODE>offset</CODE> is less than' ;
'TmsLoop.java', 44, ' * the header size or greater than the' ;
'TmsLoop.java', 45, ' * data block size' ;
'TmsLoop.java', 46, ' */ ' ;
'TmsLoop.java', 47, ' protected void read(TmsDataBlock dataBlock, int offset)' ;
'TmsLoop.java', 48, '     throws BadOffsetException' ;
'TmsLoop.java', 49, ' {' ;
'TmsLoop.java', 50, '     // check offset against bounds, this may throw BadOffsetException' ;
'TmsLoop.java', 51, '     checkOffsetBounds(dataBlock, offset);' ;
'TmsLoop.java', 52, ' ' ;
'TmsLoop.java', 53, '     // copy raw data' ;
'TmsLoop.java', 54, '     initializeRawData(dataBlock, offset, PACKED_SIZE);' ;
'TmsLoop.java', 55, ' ' ;
'TmsLoop.java', 56, '     // decode raw data' ;
'TmsLoop.java', 57, ' ' ;
'TmsLoop.java', 58, '     // incident detect is the top 2 bits (6-7) of byte 0' ;
'TmsLoop.java', 59, '     incidentDetect = (rawData[0] >> 6) & MASK_TWO_BIT;' ;
'TmsLoop.java', 60, ' ' ;
'TmsLoop.java', 61, '     // n_loops is bits 3-5 of byte 0' ;
'TmsLoop.java', 62, '     nLoops = (rawData[0] >> 3) & MASK_THREE_BIT;' ;
'TmsLoop.java', 63, ' ' ;
'TmsLoop.java', 64, '     // scan count is the bottom three bits (0-2) byte 0 and all of byte 1' ;
'TmsLoop.java', 65, '     scanCount = ((rawData[0] & MASK_THREE_BIT) << 8) | ' ;
'TmsLoop.java', 66, '         (rawData[1] & MASK_EIGHT_BIT) ;' ;
'TmsLoop.java', 67, ' ' ;
'TmsLoop.java', 68, '     // flag is the top 3 bits (5-7) of byte 2' ;
'TmsLoop.java', 69, '     flag = (rawData[2] >> 5) & MASK_THREE_BIT;' ;
'TmsLoop.java', 70, ' ' ;
'TmsLoop.java', 71, '     // volume is the low 5 bits (0-4) of byte 2' ;
'TmsLoop.java', 72, '     volume = rawData[2] & MASK_FIVE_BIT;' ;
'TmsLoop.java', 73, ' }' ;
'TmsLoop.java', 74, ' ' ;
'TmsLoop.java', 75, ' /** ' ;
'TmsLoop.java', 76, ' * @return printable representation of data in format' ;
'TmsLoop.java', 77, ' * "<CODE>loop volume scanCount nLoops flag incidentDetect</CODE>"' ;
'TmsLoop.java', 78, ' */ ' ;
'TmsLoop.java', 79, ' public String toString()' ;
'TmsLoop.java', 80, ' {' ;
'TmsLoop.java', 81, '     if (stringRep == null)' ;
'TmsLoop.java', 82, '         stringRep = new String(sensor.getId() + " " + NAME + " " + volume + " " + ' ;
'TmsLoop.java', 83, '             scanCount + " " + nLoops + " " + flag + " " + ' ;
'TmsLoop.java', 84, '             incidentDetect);' ;
'TmsLoop.java', 85, '     return stringRep;' ;
'TmsLoop.java', 86, ' }' ;
'TmsLoop.java', 87, ' ' ;
'TmsLoop.java', 88, ' /** ' ;
'TmsLoop.java', 89, ' * Generates commands to insert object into database per SDD schema.' ;
'TmsLoop.java', 90, ' * <P>' ;
'TmsLoop.java', 91, ' * Note: this might better go into a separate class, it makes TmsLoop kind' ;
'TmsLoop.java', 92, ' * of heavyweight and adds dependencies on TmsSensor and TmsDataBlock.' ;
'TmsLoop.java', 93, ' * A separate TmsSql would also isolate the SDD schema dependencies.' ;
'TmsLoop.java', 94, ' * ' ;
'TmsLoop.java', 95, ' * @return array of strings containing SQL commands' ;
'TmsLoop.java', 96, ' */ ' ;
'TmsLoop.java', 97, ' public String[] toSql() {' ;
'TmsLoop.java', 98, '     String[] sqlCmds = new String[5];' ;
'TmsLoop.java', 99, ' ' ;
'TmsLoop.java', 100, '     sqlCmds[0] = firstSQLLine;' ;
'TmsLoop.java', 101, '     sqlCmds[1] = secondSQLLine;' ;
'TmsLoop.java', 102, '     sqlCmds[2] = thirdSQLLine;' ;
'TmsLoop.java', 103, '     sqlCmds[4] = fifthSQLLine;' ;
'TmsLoop.java', 104, ' ' ;
'TmsLoop.java', 105, '     sqlCmds[3] = new String("\'" + sensor.getId() + "\'" + "," + \'\' + timestamp' ;
'TmsLoop.java', 106, ' '+' ;
'TmsLoop.java', 107, '         "\'," + volume + "," + scanCount + "," + flag + ' ;
'TmsLoop.java', 108, '         "," + nLoops + "," + incidentDetect);' ;
'TmsLoop.java', 109, '     return sqlCmds;' ;
'TmsLoop.java', 110, ' }' ;
'TmsLoop.java', 111, ' ' ;
'TmsLoop.java', 112, ' // static strings for SQL commands which don't change per instance' ;
'TmsLoop.java', 113, ' private static final String firstSQLLine = "INSERT INTO LOOP_DATA (" ;
'TmsLoop.java', 114, ' private static final String secondSQLLine = "SENSOR_ID DATA_TIME VOLUME SCAN_CO' ;
'TmsLoop.java', 115, ' NT FLAG LANE_COUNT INCIDENT_DETECT)";' ;
'TmsLoop.java', 116, ' private static final String thirdSQLLine = "VALUES (" ;
'TmsLoop.java', 117, ' private static final String fifthSQLLine = ")\n";' ;
'TmsLoop.java', 118, ' ' ;

```

```

'TmsLoop.java', 119, ' /**' ;
'TmsLoop.java', 120, ' * @return reference to sensor' ;
'TmsLoop.java', 121, ' */' ;
'TmsLoop.java', 122, ' public TmsSensor getSensor() {' ;
'TmsLoop.java', 123, '     return sensor;' ;
'TmsLoop.java', 124, ' }' ;
'TmsLoop.java', 125, ' ;
'TmsLoop.java', 126, ' /**' ;
'TmsLoop.java', 127, ' * @return timestamp' ;
'TmsLoop.java', 128, ' */' ;
'TmsLoop.java', 129, ' public TmsTimestamp getTimestamp() {' ;
'TmsLoop.java', 130, '     return timestamp;' ;
'TmsLoop.java', 131, ' }' ;
'TmsLoop.java', 132, ' ;
'TmsLoop.java', 133, ' /**' ;
'TmsLoop.java', 134, ' * @return incidentDetect value (values are enumerated in TmsDataItem).' ;
'TmsLoop.java', 135, ' */' ;
'TmsLoop.java', 136, ' public int getIncidentDetect()' ;
'TmsLoop.java', 137, ' {' ;
'TmsLoop.java', 138, '     return incidentDetect;' ;
'TmsLoop.java', 139, ' }' ;
'TmsLoop.java', 140, ' ;
'TmsLoop.java', 141, ' /**' ;
'TmsLoop.java', 142, ' * @return number of loops reporting, this should be 1' ;
'TmsLoop.java', 143, ' */' ;
'TmsLoop.java', 144, ' public int getNLoops()' ;
'TmsLoop.java', 145, ' {' ;
'TmsLoop.java', 146, '     return nLoops;' ;
'TmsLoop.java', 147, ' }' ;
'TmsLoop.java', 148, ' ;
'TmsLoop.java', 149, ' /**' ;
'TmsLoop.java', 150, ' * @return data validity flag sent by TMS' ;
'TmsLoop.java', 151, ' */' ;
'TmsLoop.java', 152, ' public int getFlag()' ;
'TmsLoop.java', 153, ' {' ;
'TmsLoop.java', 154, '     return flag;' ;
'TmsLoop.java', 155, ' }' ;
'TmsLoop.java', 156, ' ;
'TmsLoop.java', 157, ' /**' ;
'TmsLoop.java', 158, ' * @return number of scans (60 per second for 20 seconds) in which' ;
'TmsLoop.java', 159, ' * was occupied.' ;
'TmsLoop.java', 160, ' */' ;
'TmsLoop.java', 161, ' public final int getScanCount()' ;
'TmsLoop.java', 162, ' {' ;
'TmsLoop.java', 163, '     return scanCount;' ;
'TmsLoop.java', 164, ' }' ;
'TmsLoop.java', 165, ' ;
'TmsLoop.java', 166, ' /**' ;
'TmsLoop.java', 167, ' * @return raw volume, which is a 20-second vehicle count. This' ;
'TmsLoop.java', 168, ' * should be between 0 and 17.' ;
'TmsLoop.java', 169, ' */' ;
'TmsLoop.java', 170, ' public int getRawVolume()' ;
'TmsLoop.java', 171, ' {' ;
'TmsLoop.java', 172, '     return volume;' ;
'TmsLoop.java', 173, ' }' ;
'TmsLoop.java', 174, ' ;
'TmsLoop.java', 175, ' /**' ;
'TmsLoop.java', 176, ' * @return occupancy as percent, calculated from scanCount' ;
'TmsLoop.java', 177, ' */' ;
'TmsLoop.java', 178, ' public float getOccupancy()' ;
'TmsLoop.java', 179, ' {' ;
'TmsLoop.java', 180, '     return ((float) scanCount) / 12.0F;' ;
'TmsLoop.java', 181, ' }' ;
'TmsLoop.java', 182, ' ;
'TmsLoop.java', 183, ' /**' ;
'TmsLoop.java', 184, ' * @return volume rescaled to vehicles per hour' ;
'TmsLoop.java', 185, ' */' ;
'TmsLoop.java', 186, ' public int getVolume()' ;
'TmsLoop.java', 187, ' {' ;
'TmsLoop.java', 188, '     return volume * 180;' ;
'TmsLoop.java', 189, ' }' ;
'TmsLoop.java', 190, ' ;
'TmsLoop.java', 191, ' /**' ;
'TmsLoop.java', 192, ' * @return the size of the packed data (3 bytes)' ;
'TmsLoop.java', 193, ' */' ;
'TmsLoop.java', 194, ' public int getPackedSize()' ;
'TmsLoop.java', 195, ' {' ;

```

```

'TmsLoop.java', 196, '    return PACKED_SIZE;' ;
'TmsLoop.java', 197, '}' ;
'TmsLoop.java', 198, ';' ;
'TmsLoop.java', 199, '/*' ;
'TmsLoop.java', 200, ' * @return printable name "Loop".' ;
'TmsLoop.java', 201, ' */' ;
'TmsLoop.java', 202, ' public String getName() ' ;
'TmsLoop.java', 203, '{' ;
'TmsLoop.java', 204, '     return NAME;' ;
'TmsLoop.java', 205, '}' ;
'TmsLoop.java', 206, ';' ;
'TmsLoop.java', 207, '/*' ;
'TmsLoop.java', 208, ' * Checks to see if data meets validity constraints.' ;
'TmsLoop.java', 209, ' * This is an external' ;
'TmsLoop.java', 210, ' * test performed by TmsLoop, it is not sent by the TMS.' ;
'TmsLoop.java', 211, ' *' ;
'TmsLoop.java', 212, ' * @return true if data is valid, as determined by tests:' ;
'TmsLoop.java', 213, ' * <UL>' ;
'TmsLoop.java', 214, ' *     <LI> flag == DATA_GOOD' ;
'TmsLoop.java', 215, ' *     <LI> scanCount <= MAX_SCAN_COUNT' ;
'TmsLoop.java', 216, ' *     <LI> volume <= MAX_VOLUME' ;
'TmsLoop.java', 217, ' *     <LI> nLoops <= MAX_LOOPS' ;
'TmsLoop.java', 218, ' * </UL>' ;
'TmsLoop.java', 219, ' */' ;
'TmsLoop.java', 220, ' public boolean isValid()' ;
'TmsLoop.java', 221, '{' ;
'TmsLoop.java', 222, '     if ((flag == DATA_GOOD) && (scanCount <= MAX_SCAN_COUNT) &&' ;
'TmsLoop.java', 223, '         (volume <= MAX_VOLUME) && (nLoops == 1))' ;
'TmsLoop.java', 224, '         return true;' ;
'TmsLoop.java', 225, '}' ;
'TmsLoop.java', 226, '     return false;' ;
'TmsLoop.java', 227, '}' ;
'TmsLoop.java', 228, ';' ;
'TmsLoop.java', 229, ' private TmsSensor sensor;' ;
'TmsLoop.java', 230, ' private TmsTimestamp timestamp;' ;
'TmsLoop.java', 231, ';' ;
'TmsLoop.java', 232, ' // data fields sent by TMS' ;
'TmsLoop.java', 233, ';' ;
'TmsLoop.java', 234, '/*' ;
'TmsLoop.java', 235, ' * was an incident detected?' ;
'TmsLoop.java', 236, ' */' ;
'TmsLoop.java', 237, ' private int incidentDetect = 0;' ;
'TmsLoop.java', 238, ';' ;
'TmsLoop.java', 239, '/*' ;
'TmsLoop.java', 240, ' * number of loops reporting (should be 1 or 0)' ;
'TmsLoop.java', 241, ' */' ;
'TmsLoop.java', 242, ' private int nLoops = 0;' ;
'TmsLoop.java', 243, ';' ;
'TmsLoop.java', 244, '/*' ;
'TmsLoop.java', 245, ' * number of scans (60 per second) where sensor was occupied' ;
'TmsLoop.java', 246, ' */' ;
'TmsLoop.java', 247, ' private int scanCount = 0;' ;
'TmsLoop.java', 248, ';' ;
'TmsLoop.java', 249, '/*' ;
'TmsLoop.java', 250, ' * validity flag, 0 = good' ;
'TmsLoop.java', 251, ' */' ;
'TmsLoop.java', 252, ' private int flag = 0;' ;
'TmsLoop.java', 253, ';' ;
'TmsLoop.java', 254, '/*' ;
'TmsLoop.java', 255, ' * number of vehicles recorded in 20 seconds.' ;
'TmsLoop.java', 256, ' * Multiply by 180 to get vehicles/hour.' ;
'TmsLoop.java', 257, ' */' ;
'TmsLoop.java', 258, ' private int volume = 0;' ;
'TmsLoop.java', 259, ';' ;
'TmsLoop.java', 260, ' // local data fields' ;
'TmsLoop.java', 261, ';' ;
'TmsLoop.java', 262, '/*' ;
'TmsLoop.java', 263, ' * String representation, saved as optimization' ;
'TmsLoop.java', 264, ' */' ;
'TmsLoop.java', 265, ' private String stringRep; // String representation, saved as optimization' ;
'TmsLoop.java', 266, ';' ;
'TmsLoop.java', 267, '/*' ;
'TmsLoop.java', 268, ' * PACKED_SIZE uses package access so that TmsSensorType can access it' ;
'TmsLoop.java', 269, ' * without an instance.' ;
'TmsLoop.java', 270, ' */' ;
'TmsLoop.java', 271, ' * The alternative was making getPackedSize() static, but then it can't' ;
'TmsLoop.java', 272, ' * be in a superclass or an interface.' ;

```

```

'TmsLoop.java', 273, '    */ ;
'TmsLoop.java', 274, '    static final int PACKED_SIZE      = 3; ;
'TmsLoop.java', 275, '    ;
'TmsLoop.java', 276, '    /** ;
'TmsLoop.java', 277, '     * NAME uses package access so that TmsSensorType can access it' ;
'TmsLoop.java', 278, '     */ ;
'TmsLoop.java', 279, '    static final String NAME          = "Loop"; ;
'TmsLoop.java', 280, '    ;
'TmsLoop.java', 281, '    ;
'TmsLoop.java', 282, '    /** ;
'TmsLoop.java', 283, '     * maximum possible scan count, computed as 60 scans/second * 20 seconds' ;
'TmsLoop.java', 284, '     */ ;
'TmsLoop.java', 285, '    private static final int MAX_SCAN_COUNT  = 1200; ;
'TmsLoop.java', 286, '    ;
'TmsLoop.java', 287, '    /** ;
'TmsLoop.java', 288, '     * maximum volume, from TMS docs' ;
'TmsLoop.java', 289, '     */ ;
'TmsLoop.java', 290, '    private static final int MAX_VOLUME     = 17; ;
'TmsLoop.java', 291, '    ;
'TmsLoop.java', 292, '    /* ;
'TmsLoop.java', 293, '     * FLAG VALUES' ;
'TmsLoop.java', 294, '     */ ;
'TmsLoop.java', 295, '    ;
'TmsLoop.java', 296, '    /** ;
'TmsLoop.java', 297, '     * flag value indicating data is good. Any other value may mean' ;
'TmsLoop.java', 298, '     * the data is bad.' ;
'TmsLoop.java', 299, '     */ ;
'TmsLoop.java', 300, '    public static final int DATA_GOOD      = 0; ;
'TmsLoop.java', 301, '    ;
'TmsLoop.java', 302, '    /** ;
'TmsLoop.java', 303, '     * flag value indicating short pulse (< 1/15 sec)' ;
'TmsLoop.java', 304, '     */ ;
'TmsLoop.java', 305, '    public static final int SHORT_PULSE    = 1; ;
'TmsLoop.java', 306, '    ;
'TmsLoop.java', 307, '    /** ;
'TmsLoop.java', 308, '     * flag value indicating chatter (> 3 counts / second)' ;
'TmsLoop.java', 309, '     */ ;
'TmsLoop.java', 310, '    public static final int CHATTER        = 2; ;
'TmsLoop.java', 311, '    ;
'TmsLoop.java', 312, '    /** ;
'TmsLoop.java', 313, '     * flag value indicating that the values from the sensor are outside' ;
'TmsLoop.java', 314, '     * an externally defined volume/occupancy envelope. The algorithm' ;
'TmsLoop.java', 315, '     * used to determine this envelope is experimental.' ;
'TmsLoop.java', 316, '     */ ;
'TmsLoop.java', 317, '    public static final int OUTSIDE_ENVELOPE = 3; ;
'TmsLoop.java', 318, '    ;
'TmsLoop.java', 319, '    // flag values 4 and 5 are reserved' ;
'TmsLoop.java', 320, '    ;
'TmsLoop.java', 321, '    /** ;
'TmsLoop.java', 322, '     * flag value indicating the the loop was disable by the operator.' ;
'TmsLoop.java', 323, '     */ ;
'TmsLoop.java', 324, '    public static final int DISABLED       = 6; ;
'TmsLoop.java', 325, '    ;
'TmsLoop.java', 326, '    /** ;
'TmsLoop.java', 327, '     * flag value indicating the loop is bad.' ;
'TmsLoop.java', 328, '     */ ;
'TmsLoop.java', 329, '    public static final int BAD_LOOP       = 7; ;
'TmsLoop.java', 330, '}' ;
'TmsLoop.java', 331, ' ;
'TmsLoop.java', 332, ' ;
'TmsLoop.java', 333, ' /***** ;
'TmsLoop.java', 334, '  * ;
'TmsLoop.java', 335, '  * DOCUMENTATION FROM TMS:' ;
'TmsLoop.java', 336, '  * ;
'TmsLoop.java', 337, '  * RTDB_LOOP format:' ;
'TmsLoop.java', 338, '  * ;
'TmsLoop.java', 339, '  * 3 bytes, packed as follows:' ;
'TmsLoop.java', 340, '  * ;
'TmsLoop.java', 341, '  *      +-----+-----+-----+-----+ ;
'TmsLoop.java', 342, '  *      |Inc|  n |         |         | ;
'TmsLoop.java', 343, '  *      |Det|Loops|   Scan Cnt   |Flag|Volume| ;
'TmsLoop.java', 344, '  *      +-----+-----+-----+-----+ ;
'TmsLoop.java', 345, '  *      ^      [0]      ^      [1]      ^      [2]      &' ;
'TmsLoop.java', 346, '  * ;
'TmsLoop.java', 347, '  * +-----+-----+-----+-----+ ;
'TmsLoop.java', 348, '  * | Field |Width|Byte|Bits| Explanation | ;
'TmsLoop.java', 349, '  * +-----+-----+-----+-----+ ;

```



```

'TmsStation.java', 57, '    scanCount = ((rawData[0] & MASK_THREE_BIT) << 8) | ' ;
'TmsStation.java', 58, '        (rawData[1] & MASK_EIGHT_BIT) ;' ;
'TmsStation.java', 59, '    ' ;
'TmsStation.java', 60, '    // flag is the top bit (7) of byte 2' ;
'TmsStation.java', 61, '    flag = (rawData[2] >> 7) & MASK_ONE_BIT;' ;
'TmsStation.java', 62, '    ' ;
'TmsStation.java', 63, '    // volume is the low 7 bits (0-6) of byte 2' ;
'TmsStation.java', 64, '    volume = rawData[2] & MASK_SEVEN_BIT;' ;
'TmsStation.java', 65, '    }' ;
'TmsStation.java', 66, '    ' ;
'TmsStation.java', 67, '    /**' ;
'TmsStation.java', 68, '     * @return printable representation of data in format' ;
'TmsStation.java', 69, '     * "<CODE>"station volume scanCount nLoops flag incidentDetect</CODE>"' ;
'TmsStation.java', 70, '     */' ;
'TmsStation.java', 71, '    public String toString()' ;
'TmsStation.java', 72, '    {' ;
'TmsStation.java', 73, '        if (stringRep == null)' ;
'TmsStation.java', 74, '            stringRep = new String(sensor.getId() + " " + NAME + " " + volume + " " +' ;
'TmsStation.java', 75, '                scanCount + " " + nLoops + " " + flag + " " +' ;
'TmsStation.java', 76, '                incidentDetect);' ;
'TmsStation.java', 77, '        return stringRep;' ;
'TmsStation.java', 78, '    }' ;
'TmsStation.java', 79, '    ' ;
'TmsStation.java', 80, '    /**' ;
'TmsStation.java', 81, '     * Generates commands to insert object into database per SDD schema.' ;
'TmsStation.java', 82, '     *' ;
'TmsStation.java', 83, '     * @return array of strings containing SQL commands' ;
'TmsStation.java', 84, '     */' ;
'TmsStation.java', 85, '    public String[] toSql() {' ;
'TmsStation.java', 86, '        String[] sqlCmds = new String[5];' ;
'TmsStation.java', 87, '        ' ;
'TmsStation.java', 88, '        sqlCmds[0] = firstSQLLine;' ;
'TmsStation.java', 89, '        sqlCmds[1] = secondSQLLine;' ;
'TmsStation.java', 90, '        sqlCmds[2] = thirdSQLLine;' ;
'TmsStation.java', 91, '        sqlCmds[4] = fifthSQLLine;' ;
'TmsStation.java', 92, '        ' ;
'TmsStation.java', 93, '        sqlCmds[3] = new String("\'" + sensor.getId() + "\'" + ",\'" + timestamp' ;
'TmsStation.java', 94, '        '+' ;
'TmsStation.java', 95, '                "\'," + volume + "," + scanCount + "," + flag +' ;
'TmsStation.java', 96, '                "," + nLoops + "," + incidentDetect);' ;
'TmsStation.java', 97, '        return sqlCmds;' ;
'TmsStation.java', 98, '    }' ;
'TmsStation.java', 99, '    ' ;
'TmsStation.java', 100, '    // static strings for SQL commands which don't change per instance' ;
'TmsStation.java', 101, '    private static final String firstSQLLine = "INSERT INTO STATION_DATA (";' ;
'TmsStation.java', 102, '    private static final String secondSQLLine = "SENSOR_ID DATA_TIME VOLUME SCAN_CO' ;
'TmsStation.java', 103, '    'NT FLAG LANE_COUNT INCIDENT_DETECT)";' ;
'TmsStation.java', 104, '    private static final String thirdSQLLine = "VALUES (";' ;
'TmsStation.java', 105, '    private static final String fifthSQLLine = ")\\n";' ;
'TmsStation.java', 106, '    ' ;
'TmsStation.java', 107, '    /**' ;
'TmsStation.java', 108, '     * @return reference to sensor' ;
'TmsStation.java', 109, '     */' ;
'TmsStation.java', 110, '    public TmsSensor getSensor() {' ;
'TmsStation.java', 111, '        return sensor;' ;
'TmsStation.java', 112, '    }' ;
'TmsStation.java', 113, '    ' ;
'TmsStation.java', 114, '    /**' ;
'TmsStation.java', 115, '     * @return timestamp' ;
'TmsStation.java', 116, '     */' ;
'TmsStation.java', 117, '    public TmsTimestamp getTimestamp() {' ;
'TmsStation.java', 118, '        return timestamp;' ;
'TmsStation.java', 119, '    }' ;
'TmsStation.java', 120, '    ' ;
'TmsStation.java', 121, '    /**' ;
'TmsStation.java', 122, '     * @return incidentDetect value (values are enumerated in TmsDataItem).' ;
'TmsStation.java', 123, '     */' ;
'TmsStation.java', 124, '    public int getIncidentDetect()' ;
'TmsStation.java', 125, '    {' ;
'TmsStation.java', 126, '        return incidentDetect;' ;
'TmsStation.java', 127, '    }' ;
'TmsStation.java', 128, '    ' ;
'TmsStation.java', 129, '    /**' ;
'TmsStation.java', 130, '     * @return number of loops reporting, this should be 2 to 7' ;
'TmsStation.java', 131, '     */' ;
'TmsStation.java', 132, '    ' ;
'TmsStation.java', 133, '    public int getNLoops()' ;

```

```

'TmsStation.java', 134, '    {' ;
'TmsStation.java', 135, '        return nLoops;' ;
'TmsStation.java', 136, '    }' ;
'TmsStation.java', 137, '' ;
'TmsStation.java', 138, '    /** ;
'TmsStation.java', 139, '     * @return data validity flag, 0 means data not usable, 1 means data usable.' ;
'TmsStation.java', 140, '     */ ;
'TmsStation.java', 141, '' ;
'TmsStation.java', 142, '    public int getFlag()' ;
'TmsStation.java', 143, '    {' ;
'TmsStation.java', 144, '        return flag;' ;
'TmsStation.java', 145, '    }' ;
'TmsStation.java', 146, '' ;
'TmsStation.java', 147, '    /** ;
'TmsStation.java', 148, '     * @return number of scans (60 per second for 20 seconds) in which' ;
'TmsStation.java', 149, '     * was occupied.' ;
'TmsStation.java', 150, '     */ ;
'TmsStation.java', 151, '    public int getScanCount()' ;
'TmsStation.java', 152, '    {' ;
'TmsStation.java', 153, '        return scanCount;' ;
'TmsStation.java', 154, '    }' ;
'TmsStation.java', 155, '' ;
'TmsStation.java', 156, '    /** ;
'TmsStation.java', 157, '     * @return raw volume, which is a 20-second vehicle count. This' ;
'TmsStation.java', 158, '     * should be between 0 and 119.' ;
'TmsStation.java', 159, '     */ ;
'TmsStation.java', 160, '' ;
'TmsStation.java', 161, '    public int getRawVolume()' ;
'TmsStation.java', 162, '    {' ;
'TmsStation.java', 163, '        return volume;' ;
'TmsStation.java', 164, '    }' ;
'TmsStation.java', 165, '' ;
'TmsStation.java', 166, '    /** ;
'TmsStation.java', 167, '     * @return occupancy as percent, calculated from scanCount' ;
'TmsStation.java', 168, '     */ ;
'TmsStation.java', 169, '    public float getOccupancy()' ;
'TmsStation.java', 170, '    {' ;
'TmsStation.java', 171, '        return ((float) scanCount) / 12.0F;' ;
'TmsStation.java', 172, '    }' ;
'TmsStation.java', 173, '' ;
'TmsStation.java', 174, '    /** ;
'TmsStation.java', 175, '     * return volume in vehicles/hour, average of loops' ;
'TmsStation.java', 176, '     */ ;
'TmsStation.java', 177, '    public int getVolume()' ;
'TmsStation.java', 178, '    {' ;
'TmsStation.java', 179, '        // XXX: what if nLoops is 0? ;
'TmsStation.java', 180, '    } ;
'TmsStation.java', 181, '        return (volume * 180) / nLoops;' ;
'TmsStation.java', 182, '    }' ;
'TmsStation.java', 183, '' ;
'TmsStation.java', 184, '    /** ;
'TmsStation.java', 185, '     * @return the size of the packed data (3 bytes)' ;
'TmsStation.java', 186, '     */ ;
'TmsStation.java', 187, '' ;
'TmsStation.java', 188, '    public int getPackedSize()' ;
'TmsStation.java', 189, '    {' ;
'TmsStation.java', 190, '        return PACKED_SIZE;' ;
'TmsStation.java', 191, '    }' ;
'TmsStation.java', 192, '' ;
'TmsStation.java', 193, '    /** ;
'TmsStation.java', 194, '     * @return the printable string "Station"' ;
'TmsStation.java', 195, '     */ ;
'TmsStation.java', 196, '' ;
'TmsStation.java', 197, '    public String getName()' ;
'TmsStation.java', 198, '    {' ;
'TmsStation.java', 199, '        return NAME;' ;
'TmsStation.java', 200, '    }' ;
'TmsStation.java', 201, '' ;
'TmsStation.java', 202, '    /** ;
'TmsStation.java', 203, '     * Checks to see if data passes constraints. This is an external' ;
'TmsStation.java', 204, '     * test performed by TmsStation, it is not sent by the TMS.' ;
'TmsStation.java', 205, '     * ;
'TmsStation.java', 206, '     * @return true if data is valid, as determined by tests:' ;
'TmsStation.java', 207, '     * <UL>' ;
'TmsStation.java', 208, '     * <LI>flag == 1' ;
'TmsStation.java', 209, '     * <LI>scanCount <= MAX_SCAN_COUNT' ;
'TmsStation.java', 210, '     * <LI>volume <= MAX_VOLUME' ;

```

```

'TmsStation.java', 211, ' * <LI>nLoops <= MAX_LOOPS' ;
'TmsStation.java', 212, ' * </UL>' ;
'TmsStation.java', 213, ' */' ;
'TmsStation.java', 214, '' ;
'TmsStation.java', 215, ' public boolean isValid()' ;
'TmsStation.java', 216, ' {' ;
'TmsStation.java', 217, '     if ((flag == DATA_USABLE) && (scanCount <= MAX_SCAN_COUNT) &&' ;
'TmsStation.java', 218, ' (volume <= MAX_VOLUME) && (nLoops <= MAX_LOOPS))' ;
'TmsStation.java', 219, '     return true;' ;
'TmsStation.java', 220, '' ;
'TmsStation.java', 221, '     return false;' ;
'TmsStation.java', 222, ' }' ;
'TmsStation.java', 223, '' ;
'TmsStation.java', 224, ' private TmsSensor sensor;' ;
'TmsStation.java', 225, ' private TmsTimestamp timestamp;' ;
'TmsStation.java', 226, '' ;
'TmsStation.java', 227, ' // data fields from TMS' ;
'TmsStation.java', 228, '' ;
'TmsStation.java', 229, ' /**' ;
'TmsStation.java', 230, ' * was an incident reported?' ;
'TmsStation.java', 231, ' */' ;
'TmsStation.java', 232, '' ;
'TmsStation.java', 233, ' public int incidentDetect = 0;' ;
'TmsStation.java', 234, '' ;
'TmsStation.java', 235, ' /**' ;
'TmsStation.java', 236, ' * number of loops reporting' ;
'TmsStation.java', 237, ' */' ;
'TmsStation.java', 238, '' ;
'TmsStation.java', 239, ' public int nLoops = 0;' ;
'TmsStation.java', 240, '' ;
'TmsStation.java', 241, ' /**' ;
'TmsStation.java', 242, ' * occupancy' ;
'TmsStation.java', 243, ' */' ;
'TmsStation.java', 244, ' public int scanCount = 0;' ;
'TmsStation.java', 245, '' ;
'TmsStation.java', 246, ' /**' ;
'TmsStation.java', 247, ' * validity flag (1 = good)' ;
'TmsStation.java', 248, ' */' ;
'TmsStation.java', 249, ' public int flag = 0;' ;
'TmsStation.java', 250, '' ;
'TmsStation.java', 251, ' /**' ;
'TmsStation.java', 252, ' * number of vehicles counted for all loops in 20 seconds' ;
'TmsStation.java', 253, ' */' ;
'TmsStation.java', 254, ' public int volume = 0;' ;
'TmsStation.java', 255, '' ;
'TmsStation.java', 256, ' /**' ;
'TmsStation.java', 257, ' * value for flag indicating data is usable. 0 means it is not usable.' ;
'TmsStation.java', 258, ' */' ;
'TmsStation.java', 259, ' public static final int DATA_USABLE = 1;' ;
'TmsStation.java', 260, '' ;
'TmsStation.java', 261, ' // package access' ;
'TmsStation.java', 262, ' final static int PACKED_SIZE = 3; // size in bytes' ;
'TmsStation.java', 263, ' final static String NAME = "Station";' ;
'TmsStation.java', 264, '' ;
'TmsStation.java', 265, ' private String stringRep;' ;
'TmsStation.java', 266, '' ;
'TmsStation.java', 267, ' /**' ;
'TmsStation.java', 268, ' * Maximum number of loops which can report' ;
'TmsStation.java', 269, ' */' ;
'TmsStation.java', 270, ' public static final int MAX_LOOPS = 7;' ;
'TmsStation.java', 271, '' ;
'TmsStation.java', 272, ' /**' ;
'TmsStation.java', 273, ' * Max scan count' ;
'TmsStation.java', 274, ' */' ;
'TmsStation.java', 275, ' public static final int MAX_SCAN_COUNT = 1200;' ;
'TmsStation.java', 276, '' ;
'TmsStation.java', 277, ' /**' ;
'TmsStation.java', 278, ' * Max 20 second volume' ;
'TmsStation.java', 279, ' */' ;
'TmsStation.java', 280, ' public static final int MAX_VOLUME = 119;' ;
'TmsStation.java', 281, '' ;
'TmsStation.java', 282, ' }' ;
'TmsStation.java', 283, '' ;
'TmsStation.java', 284, ' /*****' ;
'TmsStation.java', 285, ' *' ;
'TmsStation.java', 286, ' * DOCUMENTATION FROM TMS:' ;
'TmsStation.java', 287, ' *' ;

```



```

'TmsTrap.java', 48, '    int tempSpeed, tempLength;' ;
'TmsTrap.java', 49, '    ' ;
'TmsTrap.java', 50, '    /* flags1 is all of byte 0 */' ;
'TmsTrap.java', 51, '    flags1 = rawData[0] & MASK_EIGHT_BIT;' ;
'TmsTrap.java', 52, '    ' ;
'TmsTrap.java', 53, '    /* flags2 is the high 4 bits (4-7) of byte 1 */' ;
'TmsTrap.java', 54, '    flags2 = (rawData[1] >> 4) & MASK_FOUR_BIT;' ;
'TmsTrap.java', 55, '    ' ;
'TmsTrap.java', 56, '    /* avg speed is the low 2 bits (0-1) of byte 1 and all of byte 2 */' ;
'TmsTrap.java', 57, '    tempSpeed = ((rawData[1] & MASK_TWO_BIT) << 8)' ;
'TmsTrap.java', 58, '    | (rawData[2] & MASK_EIGHT_BIT);' ;
'TmsTrap.java', 59, '    avgSpeed = tempSpeed / 10.0F;' ;
'TmsTrap.java', 60, '    ' ;
'TmsTrap.java', 61, '    /* avg length is bits 2 and 3 of byte 1 and all of byte 3 */' ;
'TmsTrap.java', 62, '    tempLength = ((rawData[1] & MASK_BITS_23) << 6) |' ;
'TmsTrap.java', 63, '    (rawData[3] & MASK_EIGHT_BIT);' ;
'TmsTrap.java', 64, '    avgLength = tempLength / 10.0F;' ;
'TmsTrap.java', 65, '    ' ;
'TmsTrap.java', 66, '    /* bin1 is the low 5 bits (0-4) of byte 4 */' ;
'TmsTrap.java', 67, '    bin1 = rawData[4] & MASK_FIVE_BIT;' ;
'TmsTrap.java', 68, '    ' ;
'TmsTrap.java', 69, '    /* bin2 is the high 4 bits (4-7) of byte 5 */' ;
'TmsTrap.java', 70, '    bin2 = (rawData[5] >> 4) & MASK_FOUR_BIT;' ;
'TmsTrap.java', 71, '    ' ;
'TmsTrap.java', 72, '    /* bin3 is the low 4 bits (0-3) of byte 5 */' ;
'TmsTrap.java', 73, '    bin3 = rawData[5] & MASK_FOUR_BIT;' ;
'TmsTrap.java', 74, '    ' ;
'TmsTrap.java', 75, '    /* bin4 is the high 3 bits (5-7) of byte 4 */' ;
'TmsTrap.java', 76, '    bin4 = (rawData[4] >> 5) & MASK_THREE_BIT;' ;
'TmsTrap.java', 77, '    }' ;
'TmsTrap.java', 78, '    ' ;
'TmsTrap.java', 79, '    /**' ;
'TmsTrap.java', 80, '     * @return reference to sensor' ;
'TmsTrap.java', 81, '     */' ;
'TmsTrap.java', 82, '    public TmsSensor getSensor() {' ;
'TmsTrap.java', 83, '        return sensor;' ;
'TmsTrap.java', 84, '    }' ;
'TmsTrap.java', 85, '    ' ;
'TmsTrap.java', 86, '    /**' ;
'TmsTrap.java', 87, '     * @return timestamp' ;
'TmsTrap.java', 88, '     */' ;
'TmsTrap.java', 89, '    public TmsTimestamp getTimestamp() {' ;
'TmsTrap.java', 90, '        return timestamp;' ;
'TmsTrap.java', 91, '    }' ;
'TmsTrap.java', 92, '    ' ;
'TmsTrap.java', 93, '    /**' ;
'TmsTrap.java', 94, '     * returns flags1' ;
'TmsTrap.java', 95, '     */' ;
'TmsTrap.java', 96, '    public int getFlags1()' ;
'TmsTrap.java', 97, '    {' ;
'TmsTrap.java', 98, '        return flags1;' ;
'TmsTrap.java', 99, '    }' ;
'TmsTrap.java', 100, '    ' ;
'TmsTrap.java', 101, '    /**' ;
'TmsTrap.java', 102, '     * returns flags2' ;
'TmsTrap.java', 103, '     */' ;
'TmsTrap.java', 104, '    public int getFlags2()' ;
'TmsTrap.java', 105, '    {' ;
'TmsTrap.java', 106, '        return flags2;' ;
'TmsTrap.java', 107, '    }' ;
'TmsTrap.java', 108, '    ' ;
'TmsTrap.java', 109, '    /**' ;
'TmsTrap.java', 110, '     * returns bin1' ;
'TmsTrap.java', 111, '     */' ;
'TmsTrap.java', 112, '    public int getBin1()' ;
'TmsTrap.java', 113, '    {' ;
'TmsTrap.java', 114, '        return bin1;' ;
'TmsTrap.java', 115, '    }' ;
'TmsTrap.java', 116, '    ' ;
'TmsTrap.java', 117, '    /**' ;
'TmsTrap.java', 118, '     * returns bin2' ;
'TmsTrap.java', 119, '     */' ;
'TmsTrap.java', 120, '    public int getBin2()' ;
'TmsTrap.java', 121, '    {' ;
'TmsTrap.java', 122, '        return bin2;' ;
'TmsTrap.java', 123, '    }' ;
'TmsTrap.java', 124, '    ' ;

```

```

'TmsTrap.java', 125, ' /**' ;
'TmsTrap.java', 126, ' * returns bin3' ;
'TmsTrap.java', 127, ' */' ;
'TmsTrap.java', 128, ' public int getBin3()' ;
'TmsTrap.java', 129, ' {' ;
'TmsTrap.java', 130, '     return bin3;' ;
'TmsTrap.java', 131, ' }' ;
'TmsTrap.java', 132, ' ;
'TmsTrap.java', 133, ' /**' ;
'TmsTrap.java', 134, ' * returns bin4' ;
'TmsTrap.java', 135, ' */' ;
'TmsTrap.java', 136, ' public int getBin4()' ;
'TmsTrap.java', 137, ' {' ;
'TmsTrap.java', 138, '     return bin4;' ;
'TmsTrap.java', 139, ' }' ;
'TmsTrap.java', 140, ' ;
'TmsTrap.java', 141, ' /**' ;
'TmsTrap.java', 142, ' * Returns average speed' ;
'TmsTrap.java', 143, ' */' ;
'TmsTrap.java', 144, ' public float getAvgSpeed()' ;
'TmsTrap.java', 145, ' {' ;
'TmsTrap.java', 146, '     return avgSpeed;' ;
'TmsTrap.java', 147, ' }' ;
'TmsTrap.java', 148, ' ;
'TmsTrap.java', 149, ' /**' ;
'TmsTrap.java', 150, ' * Returns length' ;
'TmsTrap.java', 151, ' */' ;
'TmsTrap.java', 152, ' public float getAvgLength()' ;
'TmsTrap.java', 153, ' {' ;
'TmsTrap.java', 154, '     return avgLength;' ;
'TmsTrap.java', 155, ' }' ;
'TmsTrap.java', 156, ' ;
'TmsTrap.java', 157, ' /**' ;
'TmsTrap.java', 158, ' * Returns a string in format "TRAP avgSpeed avgLength flags1 flags2 bin1 bin' ;
'TmsTrap.java', 159, ' bin3 bin4"' ;
'TmsTrap.java', 160, ' */' ;
'TmsTrap.java', 161, ' public String toString()' ;
'TmsTrap.java', 162, ' {' ;
'TmsTrap.java', 163, '     if (stringRep == null)' ;
'TmsTrap.java', 164, '         stringRep = new String(sensor.getId() + " " + NAME + " " + avgSpeed + ' ;
'TmsTrap.java', 165, '             " " + avgLength + " " + flags1 + " " + flags2 + ' ;
'TmsTrap.java', 166, '             " " + bin1 + " " + bin2 + " " + bin3 + " " + bin4);' ;
'TmsTrap.java', 167, '     return stringRep;' ;
'TmsTrap.java', 168, ' }' ;
'TmsTrap.java', 169, ' ;
'TmsTrap.java', 170, ' /**' ;
'TmsTrap.java', 171, ' * Generates commands to insert object into database per SDD schema.' ;
'TmsTrap.java', 172, ' *' ;
'TmsTrap.java', 173, ' * @return array of strings containing SQL commands' ;
'TmsTrap.java', 174, ' */' ;
'TmsTrap.java', 175, ' public String[] toSql() {' ;
'TmsTrap.java', 176, '     String[] sqlCmds = new String[5];' ;
'TmsTrap.java', 177, ' ;
'TmsTrap.java', 178, '     sqlCmds[0] = firstSQLLine;' ;
'TmsTrap.java', 179, '     sqlCmds[1] = secondSQLLine;' ;
'TmsTrap.java', 180, '     sqlCmds[2] = thirdSQLLine;' ;
'TmsTrap.java', 181, '     sqlCmds[4] = fifthSQLLine;' ;
'TmsTrap.java', 182, ' ;
'TmsTrap.java', 183, '     sqlCmds[3] = new String("\'" + sensor.getId() + "\'" + "," + \'" + timestamp' ;
'TmsTrap.java', 184, '         +' ;
'TmsTrap.java', 185, '             "\'" + avgSpeed + "," + avgLength + "," + flags1 +' ;
'TmsTrap.java', 186, '             "," + flags2 + "," + bin1 + "," + bin2 + "," +' ;
'TmsTrap.java', 187, '             bin3 + "," + bin4);' ;
'TmsTrap.java', 188, '     return sqlCmds;' ;
'TmsTrap.java', 189, ' }' ;
'TmsTrap.java', 190, ' ;
'TmsTrap.java', 191, ' // static strings for SQL commands which don't change per instance' ;
'TmsTrap.java', 192, ' private static final String firstSQLLine = "INSERT INTO SPEED_TRAP_DATA (" ;
'TmsTrap.java', 193, ' private static final String secondSQLLine = "SENSOR_ID DATA_TIME SPEED LENGTH F' ;
'TmsTrap.java', 194, ' AGS1 FLAGS2 BIN1 BIN2 BIN3 BIN4 )";' ;
'TmsTrap.java', 195, ' private static final String thirdSQLLine = "VALUES (" ;
'TmsTrap.java', 196, ' private static final String fifthSQLLine = ")\n";' ;
'TmsTrap.java', 197, ' ;
'TmsTrap.java', 198, ' ;
'TmsTrap.java', 199, ' /**' ;
'TmsTrap.java', 200, ' * @return the size of the packed data (6 bytes)' ;
'TmsTrap.java', 201, ' */' ;

```

```

'TmsTrap.java', 202, ' public int getPackedSize() {' ;
'TmsTrap.java', 203, '     return PACKED_SIZE;' ;
'TmsTrap.java', 204, ' }' ;
'TmsTrap.java', 205, ' ;
'TmsTrap.java', 206, ' /**' ;
'TmsTrap.java', 207, '  * @return the string "Speed Trap" ;
'TmsTrap.java', 208, '  */' ;
'TmsTrap.java', 209, ' public String getName() {' ;
'TmsTrap.java', 210, '     return NAME;' ;
'TmsTrap.java', 211, ' }' ;
'TmsTrap.java', 212, ' ;
'TmsTrap.java', 213, ' /**' ;
'TmsTrap.java', 214, '  * returns true if data is valid, as determined by tests:' ;
'TmsTrap.java', 215, '  * flags1 == 0 (no bit flags set)' ;
'TmsTrap.java', 216, '  * flags2 bit 3 == 1' ;
'TmsTrap.java', 217, '  */' ;
'TmsTrap.java', 218, ' public boolean isValid()' ;
'TmsTrap.java', 219, ' {' ;
'TmsTrap.java', 220, '     if ((flags1 == 0) && ((flags2 & FLAG2_DATA_RECEIVED) == 1))' ;
'TmsTrap.java', 221, '         return true;' ;
'TmsTrap.java', 222, ' ;
'TmsTrap.java', 223, '     return false;' ;
'TmsTrap.java', 224, ' }' ;
'TmsTrap.java', 225, ' ;
'TmsTrap.java', 226, ' private TmsSensor sensor;' ;
'TmsTrap.java', 227, ' private TmsTimestamp timestamp;' ;
'TmsTrap.java', 228, ' ;
'TmsTrap.java', 229, ' private int flags1     = 0;' ;
'TmsTrap.java', 230, ' private int flags2     = 0;' ;
'TmsTrap.java', 231, ' private int bin1       = 0;' ;
'TmsTrap.java', 232, ' private int bin2       = 0;' ;
'TmsTrap.java', 233, ' private int bin3       = 0;' ;
'TmsTrap.java', 234, ' private int bin4       = 0;' ;
'TmsTrap.java', 235, ' private float avgSpeed = 0.0F;' ;
'TmsTrap.java', 236, ' private float avgLength = 0.0F;' ;
'TmsTrap.java', 237, ' ;
'TmsTrap.java', 238, ' // package access' ;
'TmsTrap.java', 239, ' static final int PACKED_SIZE     = 6;' ;
'TmsTrap.java', 240, ' static final String NAME         = "Trap";' ;
'TmsTrap.java', 241, ' ;
'TmsTrap.java', 242, ' private String stringRep;' ;
'TmsTrap.java', 243, ' ;
'TmsTrap.java', 244, ' public static final int FLAG1_DISABLED     = 0x080;' ;
'TmsTrap.java', 245, ' public static final int FLAG1_TWO_VEHICLES = 0x040;' ;
'TmsTrap.java', 246, ' public static final int FLAG1_LOST_VEHICLE = 0x020;' ;
'TmsTrap.java', 247, ' public static final int FLAG1_BAD_OCC_RATIO = 0x010;' ;
'TmsTrap.java', 248, ' public static final int FLAG1_MIN_SPEED    = 0x008;' ;
'TmsTrap.java', 249, ' public static final int FLAG1_MAX_SPEED    = 0x004;' ;
'TmsTrap.java', 250, ' public static final int FLAG1_MIN_LENGTH   = 0x002;' ;
'TmsTrap.java', 251, ' public static final int FLAG1_MAX_LENGTH   = 0x001;' ;
'TmsTrap.java', 252, ' ;
'TmsTrap.java', 253, ' public static final int FLAG2_DATA_RECEIVED = 0x008; /* 1=data received */' ;
'TmsTrap.java', 254, ' ;
'TmsTrap.java', 255, ' private final static int MASK_BITS_23 = 0x00c; /* used to extract bits 2-3' ;
'TmsTrap.java', 256, '     for the AL98 field in' ;
'TmsTrap.java', 257, '     RTDB_SPEED_TRAP format */' ;
'TmsTrap.java', 258, ' }' ;

```

TABLE COORDINATES

```

COLUMN (COORD_TYPE, NAME1, NAME2, NAME3, UNIT1, UNIT2, UNIT3)
'spherical', 'x', 'y', 'z', 'kilometers', 'kilometers', 'kilometers' ;
'geodetic', 'latitude', 'longitude', 'altitude', 'degrees', 'degrees', 'feet' ;
'state plane', 'x', 'y', 'not used', 'miles', 'miles', 'not used' ;
'linear referencing system', 'x', 'not used', 'not used', 'miles', 'not used', 'not used' ;

```

TABLE MEASURES

```

COLUMN (COORD_TYPE, AUTHORITY, REF_SYSTEM,
        REF_PT1, REF_PT2, REF_PT3,
        ACCURACY1, ACCURACY2, ACCURACY3)
'geodetic', 'WSDOT', 'NAD23', 0, 0, 0, 0.001, 0.001, 0.001 ;
'geodetic', 'Univ of Wash.', 'NAD89', 0, 0, 0, 0.001, 0.001, 0.001 ;

```

TABLE LOOP_FLAGS

```

COLUMN ( FLAG_VAL, EXPLANATION )
0, 'Good Data' ;
1, 'Short Pulse' ;

```

```

2, 'Chatter' ;
3, 'Outside Vol/Occ Envelope' ;
4, 'Reserved' ;
5, 'Reserved' ;
6, 'Operator Disabled' ;
7, 'Bad Loop' ;

```

```

TABLE STATION_FLAGS
COLUMN ( FLAG_VAL, EXPLANATION )
0, 'Data not Usable' ;
1, 'Data Usable' ;

```

```

TABLE INCIDENT_DETECT
COLUMN ( FLAG_VAL, EXPLANATION )
0, 'No Incident' ;
1, 'Tentative' ;
2, 'Occurred' ;
3, 'Continuing' ;

```

```

TABLE CABINETS
COLUMN ( CABINET_ID, FREEWAY, TEXT, RAMP )
'ES-059D', 'I-5', 'S170thSt', 0 ;
'ES-068D', 'I-5', 'S154thSt', 0 ;
'ES-069D', 'UNKNOWN', 'UNKNOWN', 0 ;
...
'ES-502D', 'SR-520', '10thAveE', 0 ;
'ES-504R', 'SR-520', 'MontlakeBlvd-EB', 1 ;
'ES-506R', 'SR-520', 'LkWashBlvd-EB', 1 ;
...
'ES-610D', 'I-405', 'I-5', 0 ;
'ES-612D', 'I-405', 'AndoverParkW', 0 ;
'ES-614D', 'I-405', 'AndoverParkE', 0 ;
...
'ES-810D', 'I-90', 'NBI-5-EB', 0 ;
'ES-812D', 'I-90', '4thAveS-EB', 0 ;
'ES-814D', 'I-90', 'SBI-5-EB', 0 ;
...
'ES-TR9D', 'UNKNOWN', 'UNKNOWN', 0 ;
'ES-WS1D', 'UNKNOWN', 'UNKNOWN', 0 ;
'ES-WS2D', 'UNKNOWN', 'UNKNOWN', 0 ;

```

```

TABLE CABINET_LOCATION
COLUMN ( CABINET_ID, COORD_TYPE, AUTHORITY, VALUE1, VALUE2, VALUE3 )
'ES-059D', 'linear', 'TMS', 153.51, NULL, NULL ;
'ES-059D', 'linear', 'WSDot', 153.510000, NULL, NULL ;
'ES-059D', 'geodetic', 'WSDot', 47.449, -122.267, NULL ;
'ES-059D', 'geodetic', 'UW ITS group', 47451574, -122264654, NULL ;
'ES-068D', 'linear', 'TMS', 154.71, NULL, NULL ;
'ES-068D', 'linear', 'WSDot', 154.710000, NULL, NULL ;
'ES-068D', 'geodetic', 'WSDot', 47.466, -122.267, NULL ;
'ES-068D', 'geodetic', 'UW ITS group', 47465477, -122265192, NULL ;
'ES-074D', 'linear', 'TMS', 156.50, NULL, NULL ;
'ES-074D', 'linear', 'WSDot', 156.500000, NULL, NULL ;
...
'ES-940D', 'linear', 'TMS', 16.33, NULL, NULL ;
'ES-940D', 'linear', 'WSDot', 16.330000, NULL, NULL ;
'ES-940D', 'geodetic', 'WSDot', 47.543, -122.042, NULL ;
'ES-940D', 'geodetic', 'UW ITS group', 47.536885, -122.006189, NULL ;
'ES-945R', 'linear', 'TMS', 17.04, NULL, NULL ;
'ES-945R', 'linear', 'WSDot', 17.040000, NULL, NULL ;
'ES-945R', 'geodetic', 'WSDot', 47.539, -122.035, NULL ;
'ES-945R', 'geodetic', 'UW ITS group', 47.532242, -121.992777, NULL ;
'ES-992D', 'linear', 'TMS', 40.97, NULL, NULL ;
'ES-994D', 'linear', 'TMS', 43.24, NULL, NULL ;

```

```

TABLE LOOPS
COLUMN ( LOOP_ID, CABINET_ID, METERED, RD_TYPE, DIRECTION, LANE_TYPE, LANE_NUM, SENSOR_TYPE, DATA_OFFSET )
'ES-059D: MNH__5', 'ES-059D', 'NO', 'mainline', 'northbound', 'HOV mainline', 5, 'loop', 16 ;
'ES-059D: MN_Stn', 'ES-059D', 'NO', 'mainline', 'northbound', 'mainline', 0, 'station', 19 ;
'ES-059D: MN__1', 'ES-059D', 'NO', 'mainline', 'northbound', 'mainline', 1, 'loop', 22 ;
'ES-059D: MN__2', 'ES-059D', 'NO', 'mainline', 'northbound', 'mainline', 2, 'loop', 25 ;
'ES-059D: MN__3', 'ES-059D', 'NO', 'mainline', 'northbound', 'mainline', 3, 'loop', 28 ;
'ES-059D: MN__4', 'ES-059D', 'NO', 'mainline', 'northbound', 'mainline', 4, 'loop', 31 ;
'ES-059D: MSH__5', 'ES-059D', 'NO', 'mainline', 'southbound', 'HOV mainline', 5, 'loop', 34 ;
'ES-059D: MS_Stn', 'ES-059D', 'NO', 'mainline', 'southbound', 'mainline', 0, 'station', 37 ;
'ES-059D: MS__1', 'ES-059D', 'NO', 'mainline', 'southbound', 'mainline', 1, 'loop', 40 ;
'ES-059D: MS__2', 'ES-059D', 'NO', 'mainline', 'southbound', 'mainline', 2, 'loop', 43 ;

```

```

'ES-059D:MS__3', 'ES-059D', 'NO', 'mainline', 'southbound', 'mainline', 3, 'loop', 46 ;
'ES-059D:MS__4', 'ES-059D', 'NO', 'mainline', 'southbound', 'mainline', 4, 'loop', 49 ;
...
'ES-079D:AMN__1', 'ES-079D', 'NO', 'aux mainline', 'northbound', 'mainline', 1, 'loop', 178 ;
'ES-079D:AMS__1', 'ES-079D', 'NO', 'aux mainline', 'southbound', 'mainline', 1, 'loop', 181 ;
'ES-079D:MNH_S5', 'ES-079D', 'NO', 'mainline', 'northbound', 'HOV mainline', 5, 'loop', 184 ;
'ES-079D:MNH_T5', 'ES-079D', 'NO', 'mainline', 'northbound', 'HOV mainline', 5, 'speed trap', 187 ;
'ES-079D:MNH__5', 'ES-079D', 'NO', 'mainline', 'northbound', 'HOV mainline', 5, 'loop', 193 ;
'ES-079D:MN_Stn', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 0, 'station', 196 ;
'ES-079D:MN__S2', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 2, 'loop', 199 ;
'ES-079D:MN__S3', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 3, 'loop', 202 ;
'ES-079D:MN__S4', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 4, 'loop', 205 ;
'ES-079D:MN__T2', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 2, 'speed trap', 208 ;
'ES-079D:MN__T3', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 3, 'speed trap', 214 ;
'ES-079D:MN__T4', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 4, 'speed trap', 220 ;
'ES-079D:MN__1', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 1, 'loop', 226 ;
'ES-079D:MN__2', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 2, 'loop', 229 ;
'ES-079D:MN__3', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 3, 'loop', 232 ;
'ES-079D:MN__4', 'ES-079D', 'NO', 'mainline', 'northbound', 'mainline', 4, 'loop', 235 ;
'ES-079D:MSH_S5', 'ES-079D', 'NO', 'mainline', 'southbound', 'HOV mainline', 5, 'loop', 238 ;
'ES-079D:MSH_T5', 'ES-079D', 'NO', 'mainline', 'southbound', 'HOV mainline', 5, 'speed trap', 241 ;
'ES-079D:MSH__5', 'ES-079D', 'NO', 'mainline', 'southbound', 'HOV mainline', 5, 'loop', 247 ;
'ES-079D:MS_Stn', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 0, 'station', 250 ;
'ES-079D:MS__S2', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 2, 'loop', 253 ;
'ES-079D:MS__S3', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 3, 'loop', 256 ;
'ES-079D:MS__S4', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 4, 'loop', 259 ;
'ES-079D:MS__T2', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 2, 'speed trap', 262 ;
'ES-079D:MS__T3', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 3, 'speed trap', 268 ;
'ES-079D:MS__T4', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 4, 'speed trap', 274 ;
'ES-079D:MS__1', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 1, 'loop', 280 ;
'ES-079D:MS__2', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 2, 'loop', 283 ;
'ES-079D:MS__3', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 3, 'loop', 286 ;
'ES-079D:MS__4', 'ES-079D', 'NO', 'mainline', 'southbound', 'mainline', 4, 'loop', 289 ;
...
'ES-085R:MMS_Stn', 'ES-085R', 'YES', 'metered mainline', 'southbound', 'mainline', 0, 'station', 751 ;
'ES-085R:MMS__1', 'ES-085R', 'YES', 'metered mainline', 'southbound', 'mainline', 1, 'loop', 754 ;
'ES-085R:MMS__2', 'ES-085R', 'YES', 'metered mainline', 'southbound', 'mainline', 2, 'loop', 757 ;
'ES-085R:MMS__3', 'ES-085R', 'YES', 'metered mainline', 'southbound', 'mainline', 3, 'loop', 760 ;
'ES-085R:MMS__4', 'ES-085R', 'YES', 'metered mainline', 'southbound', 'mainline', 4, 'loop', 763 ;
'ES-085R:MSHDS2', 'ES-085R', 'NO', 'mainline', 'southbound', 'HOV demand', 2, 'loop', 766 ;
'ES-085R:MSHDT2', 'ES-085R', 'NO', 'mainline', 'southbound', 'HOV demand', 2, 'speed trap', 769 ;
'ES-085R:MSHD__2', 'ES-085R', 'NO', 'mainline', 'southbound', 'HOV demand', 2, 'loop', 775 ;
'ES-085R:MSHP__2', 'ES-085R', 'NO', 'mainline', 'southbound', 'HOV passage', 2, 'loop', 778 ;
'ES-085R:MSH__5', 'ES-085R', 'NO', 'mainline', 'southbound', 'HOV mainline', 5, 'loop', 781 ;
'ES-085R:MSRA__1', 'ES-085R', 'NO', 'mainline', 'southbound', 'Rt Adv Q', 1, 'loop', 784 ;
'ES-085R:MSRA__2', 'ES-085R', 'NO', 'mainline', 'southbound', 'Rt Adv Q', 2, 'loop', 787 ;
'ES-085R:MS_D__1', 'ES-085R', 'NO', 'mainline', 'southbound', 'demand', 1, 'loop', 790 ;
'ES-085R:MS_D__2', 'ES-085R', 'NO', 'mainline', 'southbound', 'demand', 2, 'loop', 793 ;
'ES-085R:MS_I__1', 'ES-085R', 'NO', 'mainline', 'southbound', 'inter Q', 1, 'loop', 796 ;
'ES-085R:MS_I__2', 'ES-085R', 'NO', 'mainline', 'southbound', 'inter Q', 2, 'loop', 799 ;
'ES-085R:MS_P__1', 'ES-085R', 'NO', 'mainline', 'southbound', 'passage', 1, 'loop', 802 ;
'ES-085R:MS_P__2', 'ES-085R', 'NO', 'mainline', 'southbound', 'passage', 2, 'loop', 805 ;
'ES-085R:MS_Q__1', 'ES-085R', 'NO', 'mainline', 'southbound', 'queue', 1, 'loop', 808 ;
'ES-085R:MS_Q__2', 'ES-085R', 'NO', 'mainline', 'southbound', 'queue', 2, 'loop', 811 ;
...
'ES-102R:MMN_Stn', 'ES-102R', 'YES', 'metered mainline', 'northbound', 'mainline', 0, 'station', 1579 ;
'ES-102R:MMN__1', 'ES-102R', 'YES', 'metered mainline', 'northbound', 'mainline', 1, 'loop', 1582 ;
'ES-102R:MMN__2', 'ES-102R', 'YES', 'metered mainline', 'northbound', 'mainline', 2, 'loop', 1585 ;
'ES-102R:CNLA__2', 'ES-102R', 'NO', 'collector-distributor', 'northbound', 'Lt Adv Q', 2, 'loop', 1588 ;
'ES-102R:CNRA__1', 'ES-102R', 'NO', 'collector-distributor', 'northbound', 'Rt Adv Q', 1, 'loop', 1591 ;
'ES-102R:CNRA__2', 'ES-102R', 'NO', 'collector-distributor', 'northbound', 'Rt Adv Q', 2, 'loop', 1594 ;
...
'ES-108D:RN_O__1', 'ES-108D', 'NO', 'reversible', 'northbound', 'on ramp', 1, 'loop', 1903 ;
'ES-108D:RN__1', 'ES-108D', 'NO', 'reversible', 'northbound', 'mainline', 1, 'loop', 1906 ;
...
'ES-WS2D:MS_Stn', 'ES-WS2D', 'NO', 'mainline', 'southbound', 'mainline', 0, 'station', 14053 ;
'ES-WS2D:MS_XS1', 'ES-WS2D', 'NO', 'mainline', 'southbound', 'exit', 1, 'loop', 14056 ;
'ES-WS2D:MS_XT1', 'ES-WS2D', 'NO', 'mainline', 'southbound', 'exit', 1, 'speed trap', 14059 ;
'ES-WS2D:MS_X__1', 'ES-WS2D', 'NO', 'mainline', 'southbound', 'exit', 1, 'loop', 14065 ;
'ES-WS2D:MS_X__2', 'ES-WS2D', 'NO', 'mainline', 'southbound', 'exit', 2, 'loop', 14068 ;

```

Appendix C

SDD/SED Client Receiver Example

This section contains two examples of how an SDD/SED client application could connect, receive and extract useful information from the freeway loops data stream. As with the previous example files, these are available by FTP from the Smart Trek SDD/SED Distribution site at “ftp:// (TBD)”.

Method 1: Receipt and Extraction of Loops Data to ASCII File

Java source file: TmsReceiver.java

```
/**
 * Bridge class which adds traffic-specific callbacks to SddReceiver.
 * This class may be extended and the methods selectSensors() and
 * processData() overridden for different functionality, some sample
 * methods are illustrated.
 *
 * As implemented, TmsReceiver selects all files and generates the
 * six files mentioned in the README file. Output to standard output
 * is purely diagnostic and will be cleaned up for the next release.
 *
 * @author RWK
 * @version 0.90
 */

class TmsReceiver extends SddReceiver
    implements ContentListener, DataListener, StatusUpdateListener
{
    /**
     * Constructor sets SddReceiver attributes
     */
    TmsReceiver()
    {
        // construct SddReceiver with the GUI pointer set to null, the
        // compress flag cleared, and the fileOut and sqlOut flags set.
        super(null, false, true, true);
    }

    /**
     * Callback for the ContentListener interface, calls selectSensors().
     * This binds the generic contentReceived method to a traffic-
     * specific method.
     */
    public void contentReceived(ContentEvent event) {
        TmsSensor[] sensors = (TmsSensor[]) event.getContent();
        selectSensors(sensors);
    }

    /**
     * Callback for the DataListener interface, calls processData().
     */
    public void dataReceived(DataEvent event) {
        processData((TmsData[]) event.getData());
    }

    /**
     * selectSensors is where sensors are selected. This should be
     * overridden in a subclass to provide the desired behavior.
     * The default behavior provided is to select all sensors.
     * Some sample implementations are provided below.
     */
    public void selectSensors(TmsSensor[] sensor) {
        selectAll();
    }
}
```

```

/**
 * sample implementation for selectSensors() illustrating selection
 * of sensors named in a file
 */
public void selectSensorsFromFile() {
    setSensorFile("sensors.txt");
}

/**
 * sample implementation for selectSensors() illustrating selection
 * of specified sensors
 */
public void selectSpecifiedSensors() {
    select("ES-080D:_MN_Stn");
    select("ES-080D:_MN__1");
    select("ES-080D:_MN__T1");
}

/**
 * processes incoming TmsData. This method should be overridden,
 * the default action provided is to do nothing (so that TmsReceiver
 * may be used to generate output files)
 */
public void processData(TmsData[] data)
{
    System.out.println("got " + data.length + " data records");
}

/**
 * sample implementation for processData which writes SQL to standard
 * output
 */
public void generateSQL(TmsData[] data) {
    for (int i = 0 ; i < data.length ; i++) {
        String[] sql = data[i].toSql();
        for (int j = 0; j < sql.length; j++)
            System.out.println(sql[j]);
    }
}

/**
 * sample implementation for processData which writes generated data
 * fields to standard output:
 * for loops, volume (vehicles/hr) and occupancy
 * for stations, volume, occupancy, and number of lanes
 * for speed trap, speed and length
 */
public void generateOutput(TmsData[] data) {
    for (int i = 0 ; i < data.length ; i++) {
        if (data[i] instanceof TmsLoop) {
            TmsLoop loop = (TmsLoop) data[i];
            System.out.println(loop.getSensor().getId() + " " + loop.getVolume() + " " +
                loop.getOccupancy());
        }
        else if (data[i] instanceof TmsStation) {
            TmsStation station = (TmsStation) data[i];
            System.out.println(station.getSensor().getId() + " " + station.getNLoops() + "
" +
                station.getVolume() + " " + station.getOccupancy());
        }
        else if (data[i] instanceof TmsTrap) {
            TmsTrap trap = (TmsTrap) data[i];
            System.out.println(trap.getSensor().getId() + " " + trap.getAvgSpeed() + " " +
                trap.getAvgLength());
        }
    } // end for
}

```

```

/**
 * callback method for StatusUpdateListener.
 */
public void statusUpdateReceived(StatusUpdateEvent event)
{
    boolean printit = false;

    StatusUpdateData data = event.getUpdateData();

    if ( printit )
    {
        System.out.println( "IspExample: update" );
        System.out.println( "IspExample: schema: " +
            data.load_time_schema + " / " +
            data.parse_time_schema );
        System.out.println( "IspExample: contents: " +
            data.load_time_contents + " / " +
            data.parse_time_contents );
        System.out.println( "IspExample: data: " +
            data.load_time_data + " / " +
            data.parse_time_data );
        System.out.println( "IspExample: extractor: " +
            data.load_time_extractor + " / " +
            data.parse_time_extractor );
    }
}

/**
 * instantiates TmsReceiver from command line, connects to source
 */
public static void main(String[] args)
{
    TmsReceiver app = new TmsReceiver();
    app.connect("sdd.its.washington.edu", "9033");
}
}

```

TMS Loops ASCII File Result: sdd.textdata

```

ES-059D: _MNH__5 Loop 0 0 0 0 0
ES-059D: _MN_Stn Station 0 0 0 0 0
ES-059D: _MN___1 Loop 0 0 0 0 0
...
ES-068D: _MN_O_1 Loop 0 0 0 0 0
ES-068D: _MN_O_2 Loop 0 0 0 0 0
ES-068D: _MN_Stn Station 0 0 0 0 0
ES-068D: _MN___1 Loop 0 0 0 0 0
...
ES-079D: AMN___1 Loop 4 183 1 0 0
ES-079D: AMS___1 Loop 10 102 1 0 0
ES-079D: _MNH_S5 Loop 4 37 1 0 0
ES-079D: _MNH_T5 Speed Trap 0.0 0.0 0 0 0 0 0 0
ES-079D: _MNH__5 Loop 1 2 1 0 0
ES-079D: _MN_Stn Station 39 203 4 1 0
ES-079D: _MN__S2 Loop 6 159 1 0 0
ES-079D: _MN__S3 Loop 12 191 1 0 0
ES-079D: _MN__S4 Loop 14 210 1 0 0
ES-079D: _MN__T2 Speed Trap 57.9 25.7 0 8 7 1 1 0
ES-079D: _MN__T3 Speed Trap 64.9 24.4 16 8 6 1 0 1
ES-079D: _MN__T4 Speed Trap 71.0 21.7 16 8 3 0 0 0
ES-079D: _MN___1 Loop 6 188 1 0 0
...

```

Method 2: Receipt and Extraction of Selected Loops Data Using Selection Criteria in an ASCII File

Java source file: TmsSampleFile.java

```
/**
 * Sample Receiver based on TmsReceiver. This version selects sensors
 * listed in a file.
 */

class TmsSampleFile extends TmsReceiver
{
    /**
     * Constructor, connects to SDD Transmitter
     */
    TmsSampleFile()
    {
        super();
    }

    /**
     * selectSensors is overridden to select sensors from file
     */
    //XXX: need feedback as to what sensors were/weren't selected
    // due to availibility
    public void selectSensors(TmsSensor[] sensor) {
        setSensorFile("sensor.txt");
    }

    /**
     * processData is overridden to print the volume and occupancy for loops;
     * volume, occupancy, and number of lanes for stations; and speed and
     * length for speed traps.
     */
    public void processData(TmsData[] data)
    {
        System.out.println("got " + data.length + " data records");

        for (int i = 0 ; i < data.length ; i++) {
            if (data[i] instanceof TmsLoop) {
                TmsLoop loop = (TmsLoop) data[i];
                System.out.println(loop.getSensor().getId() + " " + loop.getVolume()
                    + " " + loop.getOccupancy());
            }
            else if (data[i] instanceof TmsStation) {
                TmsStation station = (TmsStation) data[i];
                System.out.println(station.getSensor().getId() + " " +
                    station.getNLoops() + " " + station.getVolume()
                    + " " + station.getOccupancy());
            }
            else if (data[i] instanceof TmsTrap) {
                TmsTrap trap = (TmsTrap) data[i];
                System.out.println(trap.getSensor().getId() + " " +
                    trap.getAvgSpeed() + " " + trap.getAvgLength());
            }
        } // end for
    }

    public void statusUpdateListener(StatusUpdateEvent event)
    {
        boolean printit = false;

        StatusUpdateData data = event.getUpdateData();

        if ( printit )
    }
}
```

```

    {
        System.out.println( "IspExample: update" );
        System.out.println( "IspExample: schema: " +
            data.load_time_schema + " / " +
            data.parse_time_schema );
        System.out.println( "IspExample: contents: " +
            data.load_time_contents + " / " +
            data.parse_time_contents );
        System.out.println( "IspExample: data: " +
            data.load_time_data + " / " +
            data.parse_time_data );
        System.out.println( "IspExample: extractor: " +
            data.load_time_extractor + " / " +
            data.parse_time_extractor );
    }
}

public static void main(String[] args)
{
    TmsSampleFile app = new TmsSampleFile();
    app.connect("sdd.its.washington.edu", "9033");
}
}

```

TMS Loops Selection File: **“sensor.txt”**

```

ES-059D: _MNH__5
ES-105D: _CS__2
ES-161D: _MS__4
ES-264D: _ME__1
ES-634R: _MN_P_1
ES-716R: _MN__2
ES-860D: _MW__3
ES-TD1D: AMS__T1

```

Appendix D

SDD/SED Developer's Technical Note Read Me File

Seattle MDI SDD Java port Version 0.0beta release readme 8/27/97

- 1) Intro
 - 1.1) Release Contents
 - 1.2) Quick Start
 - 1.3) Build Information
 - 1.4) Java links
- 2) Getting Started
 - 2.1) Extracting the archive
 - 2.2) Downloading Java
 - 2.3) Setting Java environment
 - 2.4) Execution
 - 2.5) Data output files descriptions
 - 2.6) Looking at the ISP examples
 - 2.7) Whats next
- 3) Trouble shooting
- 4) Known problems
- 5) Todo
- 6) Contact

-
- 1) Intro
 - 1.1) Release Contents

The main distribution of the SDD Receiver is a single file called: `sdd_release.zip` or `sdd_release.tar.gz`. Your target platform will determine the file you should use. Microsoft Dos/Windows platforms should use the zip file and Unix platforms should use the gzipped tar file. If you're reading this, then you probably have extracted the archive file.

The contents of the main archive file are all of the Java class, source code for the ISP example classes, source code archive, and a documentation archive. You will not need to extract the source code and documentation archives unless you are making code changes to Sdd Receiver base classes.

The Java class files (`?.class`) are Java bytecode binary files. The ISP example source code files (`?.java`) are Java source code files.

1.2) Quick start

If you have Java loaded with the current directory as part of your classpath environment variable (see section 2 if this sounds foreign to you), you can simply type:

```
java SddReceiverGui
```

This will start the GUI receiver. Simply click the "connect" button to start downloading data.

1.3) Build information

This version of the Java Port of the SDD receiver was compiled on Linux version 2.1.42. The Java compiler used is version 1.1.1 version 2 from the Java Development Kit. The JDK was ported to Linux by Steve Byrne (sbb@gnu.ai.mit.edu). Gnus' cpp version 2.7.2.1 (i386 Linux/ELF) was used as a generic preprocessor, so that C-line #defines could be used. No need to worry about implementing this, since you can just use the provided .java files. The pre-pre-processor (nice word?) source files are named with a .jpp extension. The development platform was a P-Pro (686) based linux box.

1.4) Related links

Suns JDKs:
<http://java.sun.com/products/index.html>

Cup:
http://www.cc.gatech.edu/gvu/people/Faculty/hudson/java_cup/home.html

Microsofts Java SDK 2.0beta:
<http://www.microsoft.com/java/sdk/>

Winzip link:
<http://www.winzip.com>

2) Usage

2.1) Extracting the archive

The SDD receiver is distributed in a zip file format for Windows users and a gzip compressed tar file for the unix platform users. You will need an archiving tool in order to extract the SDD release file. WinZip is the recommended archive/extraction program (see section 1.4: Related Links) for the Windows platform users. Tar and gzip are standard unix commands.

Windows:
Download the Winzip shareware program and extract the archive. See winzip help for help on extracting the zip file. If you are using the command line version of pkunzip, make sure you specify the '-d' option so that pkunzip will restore the correct directory structure of the archived files. Also make sure that you have a version that can handle extracting long filenames, otherwise the Java class filenames will get hosed.

Windows command prompt:

```
pkunzip -d sdd_release.zip
```

Unix:

```
gzip -cd sdd_release.tar.gz | tar -xvf -
```

2.2) Downloading java

You should choose the Java environment you wish to use. It is recommended that you use Sun's JDK. You will need a Java environment that supports Java 1.1 (for example, Microsofts Java SDK 1.5.1 only supports JDK 1.0.2; you will need to upgrade to 2.0, which is in beta as of this writing). See section 1 for the www links for the Java environment of your choice.

2.3) Setting Java environment

See the chosen distribution for help setting the environment classpath variable. Usually this variable only contains the directory of the Java classes or the class.zip file. If this is the case, you will need to add the current directory to the classpath variable (usually means simply adding a ":@" to the end of the current classpath). Below are some examples of how you might setup the classpath environment variable:

Windows 95:

```
add the following to the \autoexec.bat file (assuming that the
jdk was installed in the c:\progra~1\jdk directory):
```

```
set CLASSPATH=.;c:\progra~1\jdk\classes
```

Windows NT:

```
you will have to goto the user admin screen and add the CLASSPATH
variable
```

Unix:

```
add the following to the users .cshrc (provided of course they
are using csh/tcsh):
```

```
setenv CLASSPATH .:$CLASSPATH
```

Linux:

```
add the following to the users .cshrc (provided of course they
are using csh/tcsh, this also assumes that the jdk was installed
in the /usr/jdk1.1.1 directory):
```

```
setenv CLASSPATH "/usr/jdk1.1.1/lib/classes.zip:."
setenv LD_LIBRARY_PATH "/usr/jdk1.1.1/lib/i586/green_threads"
```

2.4) Execution

There are two ways that the user can get data from the UW server. The first is by way of a GUI. The gui is built in a Java class called SddReceiverGui. Type the following to invoke the gui (do this in the directory containing the receiver classes):

```
java SddReceiverGui
```

The gui should pop-up and have the following values within the text fields:

```
host: sdd.its.washington.edu (128.95.116.72 - subject to change)
port: 9033
```

These fields specify the host and port of the Traffic Loops data. You may have to type in the actual IP address of `sdd`, but this is only if your DNS is not working. Connecting to the server is accomplished by clicking on the "connect" button. This will connect you to the server and receive the loops data. Currently, the gui only supports sensor selection by means of a file, defaulted to 'sensor.txt'. During the download and parsing of the data, you should see some statistics displayed within the gui. These are used to measure performance and can be ignored.

The second method of invoking the receiver is by means of the command line ISP example programs. These are delineated by the sensor selection process that they use. Sensors can be selected in three different ways: All, select from a sensor list file (the GUI uses this method), and single selections (multiple sensors can be selected). These examples are invoked by typing:

```
java TmsReceiver
java TmsSampleFile
java TmsSampleSelect
```

The Sdd Receiver will store the data within files. These files are described in the following section.

2.5) Data output files descriptions

The Sdd receiver applications will generate the following files:

`sdd.schema_<data def serial number>` Contains the SDD schema definition as received from the server.

`sdd.contents_<data def serial number>` Contains the SDD contents as received from the server.

`sdd.data` Contains all of the raw binary data (all sensors) as received from the server. This file is overwritten with every receipt of new data.

* `sdd.textdata` Contains the extracted loop sensor data (only sensors selected). This file is overwritten with every receipt of new data.

```
Contains:  loop_name
           volume    ( # cars per 20 seconds not veh/hr )
           scanCount
           nLoops
           flag
           incidentDetect
```

% `sdd.sql` Contains the SQL table definition and table contents for the data dictionary.

##`sdd.sqldata` Contains the SQL commands for inserting loop sensor data into tables. Contains only the sensors selected. This file is overwritten with every receipt of new data. Contains the same information as `sdd.textdata`.

* --> this is the file that has *the data

```
% --> these files are currently not generated by the SddReceiverGui
      program.
# --> this file is currently not generated by the Tms* programs
```

2.6) Looking at the ISP examples

The Java source code is extracted with the classes for your convenience.

The data items in the TmsData objects are the data items from the TMS:
volume is in cars per 20 seconds (not veh/hr)
there is a scan count field
there is no occupancy field
there is no generated speed for single-loop sensors

The textual data files (sdd.textdata, sdd.sqldata) refer to the TMS data items, not to any generated values.

There are methods in the data classes which generate other values:

```
getVolume() returns volume scaled to vehicles/hour
(getRawVolume() returns the 20 second volume)
getOccupancy() calculates the occupancy from the scan count
```

There is no generated speed in the TmsLoop or TmsStation classes since we are not promoting use of G-Factor speeds.

3) Trouble shooting: TBD

4) Known problems

No major problems known at this time.

5) Todo

```
Phase II of the receiver will allow dynamic download of the extractor
Selectable file output
add sql data output file to gui and Tms* applications
```

6) Contact information

problems/questions/etc can be address via email to:

SDD@its.washington.edu