Transportation Data Research Laboratory Software Development Kit

User's Manual

7/1/2006

Table of Contents

Section 1.0- Introduction:

The Transportation Data Research Laboratory Software Development Kit, henceforth called TDRL SDK, is a software library package aimed at easing the process of writing traffic analysis software using *.traffic data file. It has been written to be used in any .NET application. The TDRL SDK has the following features:

- hierarchical structure including objects to represent roads, stations, and single detectors.
- Fast search capabilities for single detectors.
- Volume and Occupancy data retrieval at the road, station, or detector level.
- Automatic multi-day data retrieval, specified by start date and end date.
- Automatic generation of stations and road objects through road and station specification files.
- Custom exceptions thrown with detailed debugging information to assist in locating bugs.
- Full Ndoc documentation to provide help within Visual Studio, as well as provide an easy to use web interface.

Section 1.1- Setting up your application to include TDRL SDK

Setup for TMC SDK is very simple. Install the provided TDRL_SDK.msi file (just double click on the file). This will install example codes, manual and dll at the directory "Program Files\TDRL_SDK". After that all that is required is to make a reference to the TMC_SDK.dll file. This can be done in 3 easy steps:

- Navigate to the solution explorer in your project.
- Right click on the References icon, select Add Reference.
- Select the TDRL_SDK.dll file from the directory "Program Files\TDRL_SDK" to be added as a reference. Hit the Add button.

You may now use all classes included in TDRL_SDK! It will be useful to put the following line at the top of your code:

```
Imports TDRL_SDK
```

The Imports statement will prevent you from requiring a package specifier before declaring a TDRL_SDK object.

Section 2.0- TDRL_SDK Design

As noted in Section 1, TDRL_SDK uses a hierarchical structure to represent all of it's elements. Stations are built from many detectors, and Roads are built from many stations. This structure does not include the custom exception objects nor the datafile (i.e., *.traffic file) objects that must be used to extract data. Figure 1 depicts the structure of the library.

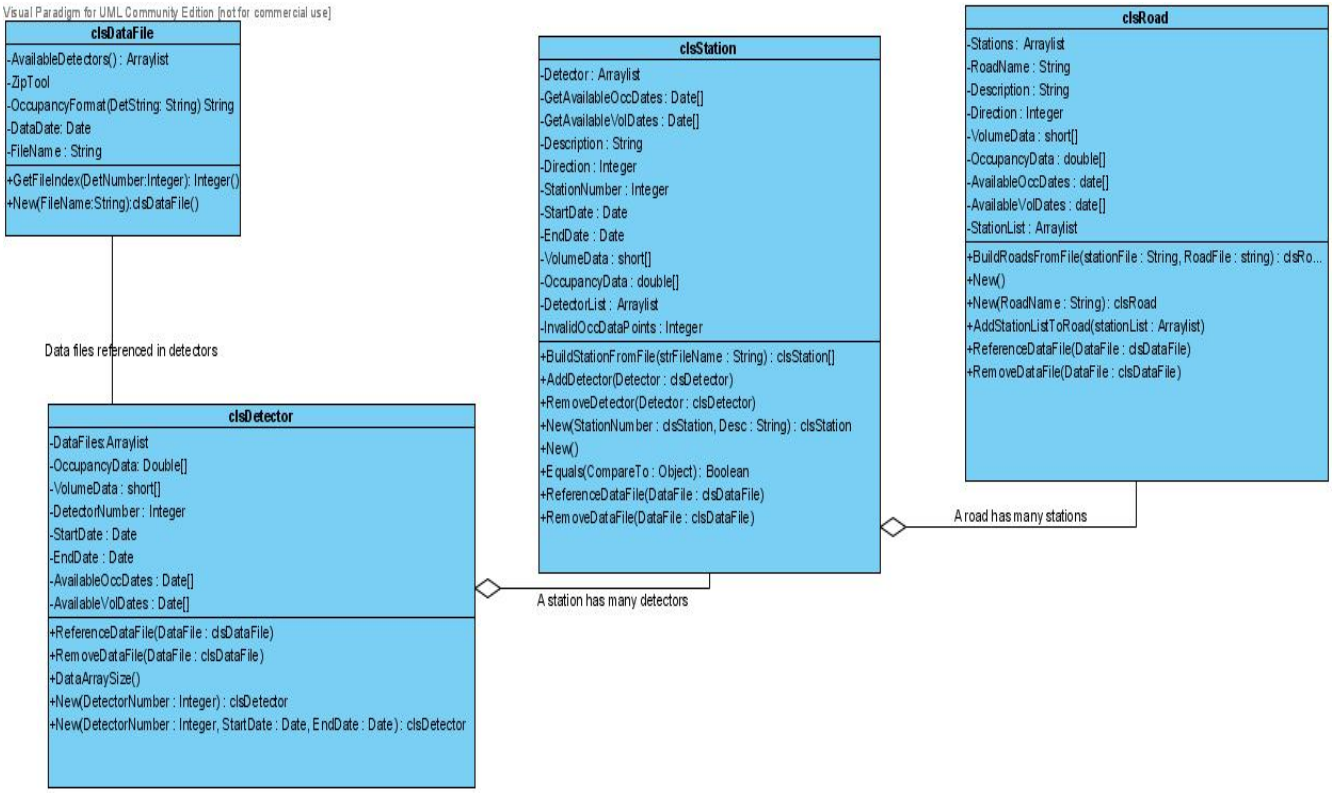

Figure 1- TDRL_SDK Class Diagram

As shown in the diagram, objects are built up from more primitive objects. Roads are built from Stations, and Stations are built from Detectors. All data extraction is ultimately done at the Detector level. That data is interpreted and presented at other levels. A description of each class follows.

Section 2.1- Class Descriptions

clsDataFile- This class is perhaps the most important class of this library. It maintains a link to a datafile, that is used for data extraction. It also keeps a list of detectors whose data is included in a particular datafile. Full documentation on this class may be found within the html SDK documentation.

clsDetector- The clsDetector class represents a single detector. The main ability for this object is the ability to extract data. There is also the ability for an object of this class to report a list of dates it knows are valid based upon the datafiles that have been referenced with it. Full documentation on this class and it's members may be found

within the html SDK documentation.

clsStation- The clsStation class is an encapsulation of multiple clsDetector objects. To add a detector, it can be added manually, or all detectors can be loaded automatically via text file specification. There is also a function to report available dates where data is known to exist for the station. Full documentation on this class and it's members may be found within the html SDK documentation.

When data is retrieved from a station, the following rules apply:

- All detectors in a level must at least have a valid datafile referenced to them for the desired date (the data itself need not be completely valid).

- Volume data points are calculated by the sum of each individual data point from each detector in the level.
  For Example, if the station data point were for 3:10:30 PM, it would be the sum of each detector's 3:10:30 PM data point.

- If an invalid detector data point is found (-1 entry), the station object then falls back on secondary and tertiary detector levels to fill the data point. (Applies to volume data only).

- Occupancy data points are calculated by the mean of all the individual data points from the detectors in a single level. Since a valid station result can occur even with some invalid detector data, a level number must be supplied from the user. A function is included that will determine the detector level with the least number of errors, and will produce the most accurate results.

clsRoad- The clsRoad class represents a single road. It is made up of multiple station objects. All functions called on the road class are reciprocated down to the station objects, and ultimately the detector objects. Stations can be added to a road in a couple of different ways. First, a function is included to add an array of stations to a road. Second, a function to build roads from a text file specification is also included. Third, Stations may be added one at a time. When data is retrieved from a road, an arraylist of all the data from all stations is returned. Full documentation for the clsRoad class may be found in the html version of the TDRL SDK documentation.

clsMissingDataException- This exception is thrown whenever data is requested from a detector/station/road, and the corresponding datafile does not exist, or has not been referenced. A detailed debug message is included with this exception to help pinpoint where the error occurred.

clsCustomException- This exception is thrown whenever any other exception occurs. It will give a detailed description of where the error occurred. If possible, a short explanation of why the error occurred (and how to fix it) is given.

Section 3.0- Using TDRL SDK

This section will give instructions on how to use the main features of TDRL SDK.

Obtaining data from a road, station, or detector-
There are two things that MUST be done before data can be extracted from any object. First, a clsDataFile object MUST be referenced with the detector/station/road object. This is how the object obtains data. Without a valid data file, a clsMissingDataException is thrown. Second, a start date and an end date including time must be provided. They are the beginning and end points for the data extraction. If there are multiple days between the start and end date, the data files for each date must be referenced. When data is retrieved, an array of type "short" is returned for volume and array type double is returned for occupancy. Short stands for short integer, and is a 16-bit signed integer.

Obtaining available dates from a road, station, or detector-
When obtaining available dates for a given detector/station/road, simply call the corresponding "getAvailableDates" function (exact function names can be found the html SDK documentation). Keep in mind that in order for a date to be valid, a datafile must exist for that date. For accuracy, datafiles must be referenced before requesting available dates.

Building a list of Stations via text specification-
TDRL SDK allows automatic generation of station objects through text file. An example of this text file is provided with your package. To run the auto generate feature, simply call the clsStation.BuildStationsFromFile function. A String filename is all that is required. An arraylist of stations is returned.

Building a list of Roads via text specification-
TDRL SDK also allows the automatic generation of roads via text file. To do this, simply call the clsRoad.BuildRoadsFromFile. The return is an arraylist of clsRoad objects. Since roads contain more complex objects (stations) than stations do (detectors), it is required that both a road specification file and a station specification file are given, so that the program can automatically load all the stations that are to be added to the given road.

Section 3.1- Interpreting Exceptions

There are two exceptions that may be thrown by TDRL SDK (described in section 2.1).

clsMissingDataException-
This exception is only thrown when a data is requested, and the corresponding clsDataFile object doesn't exist. Check to be sure you have created a link to a .traffic file, and that you have referenced this clsDataFile object with your detector/road/station object.
The Referencedatafile function only creates the reference if there exists data for that

particular detector in the file. For example, you have a detector, number 5000. If you try to reference a data file that has no data for detector 5000, the data file is simply not referenced. When this happens, the return of Referencedatafile is false.

clsCustomException-

This exception can be thrown for any reason. For more predicable/common problems, a problem cause statement is included with the exception. The exception's DebugMessage property contains this explanation.

Section 4.0- Final Notes

TDRL SDK was written by Dan Cinnamon, working with Dr. Taek Mu Kwon at the University of Minnesota- Duluth. Example code slices provided below was written and tested by Lalit Nookala. It is intended to be a community service contribution, and therefore is free to distribute and modify.

AnalyzeThis!- A sample application that gives some examples on how to use the TDRL SDK package. It does not use all functionality, and it not complete, but it is still a good starting point for learning to use the SDK.

Example Code Slices

Example 1: Extract the 30 sec volume and occupancy values for a single detector for one day. It is assumed that the data file is "20030618.traffic".

```
Dim datafile As New clsDataFile("20030618.traffic") 'declare the traffic file from
which the data has to be extracted
Dim vol() As Short
Dim occ() As Double
Dim i As Integer, buf As String
Dim det As New clsDetector(100) ' declare the new detector with detector no. 100
det.ReferenceDataFile(datafile) 'reference the .traffic file from which the data
has to be accessed
det.StartDate = #6/18/2003# 'set the start date and time
det.EndDate = #6/18/2003 11:59:59 PM# 'set the end date and time

vol = det.VolumeData 'retrieve the volume data for the detector
occ = det.OccupancyData 'retrieve the occupancy data for the detector

buf = "Volume and Occupancy of the detector on " + det.StartDate.ToShortDateString
+ vbCrLf
buf += "# ," + det.DetectorNumber.ToString + ".vol, " + det.DetectorNumber.ToString
+ ".occ" + vbCrLf
For i = 0 To vol.Length - 1
    buf += CStr(i + 1) + ".  " + CStr(vol(i)) + vbTab + ", " +
occ(i).ToString("0.00") + vbCrLf
Next

If buf Is Nothing Then
       TextBox1.Text = "No data returned"
Else
       TextBox1.Text = buf
```

```
End If
```
Example 2: Extract the 30 sec volume and occupancy values for a single station for a single day. The station comprises of detectors 10, 100, and 1000.

```
Dim datafile As New clsDataFile("20030618.traffic") 'declare and instantiate the
traffic file from which the data has to be extracted
Dim sta As New clsStation(555, "Test station") 'declare and instantiate the station
for which the volume and occupancy data is extracted

Dim det1 As New clsDetector(10)
Dim det2 As New clsDetector(100)
Dim det3 As New clsDetector(1000)
'add one detector at a time to the station
sta.AddDetector(det1)
sta.AddDetector(det2)
sta.AddDetector(det3)
sta.ReferenceDataFile(datafile) 'reference the traffic data file for which the data
has to be extracted.

sta.StartDate = #6/18/2003# 'set the start date and time for which the data has to
be extracted
sta.EndDate = #6/18/2003 11:59:59 PM# 'set the end date and time for which the data
has to be extracted

Dim vol() As Short
Dim occ() As Double

vol = sta.VolumeData 'retrieve the volume for the station
occ = sta.OccupancyData ' retireve the occupancy for the station. Both arrays will
contain 2,880 values.
```

Example 3: Extract the 30 sec volume and occupancy values for a road for a single day. The station comprises of stations 111 (10, 100), 222 (20, 200), 333 (30, 300), and 444 (40, 400) where (#, #) are detectors of the station.

```
Dim vol As New ArrayList
Dim occ As New ArrayList
Dim i As Integer, buf As String
Dim stations As New ArrayList ' the station arraylist is required to build the road
Dim testRoad As New clsRoad(6)
Dim datafile As New clsDataFile("20030618.traffic") 'declare and instantiate the
traffic file from which the data has to be extracted

'declare all the detectors that will be used in building the stations that are
present on the road
Dim sta1Det1 As New clsDetector(10, CDate("#6/18/2003#"), CDate(#6/18/2003 11:59:59
PM#))
Dim sta1Det2 As New clsDetector(100, CDate("#6/18/2003#"), CDate(#6/18/2003
11:59:59 PM#))
Dim sta2Det1 As New clsDetector(20, CDate("#6/18/2003#"), CDate(#6/18/2003 11:59:59
PM#))
Dim sta2Det2 As New clsDetector(200, CDate("#6/18/2003#"), CDate(#6/18/2003
11:59:59 PM#))
Dim sta3Det1 As New clsDetector(30, CDate("#6/18/2003#"), CDate(#6/18/2003 11:59:59
PM#))
Dim sta3Det2 As New clsDetector(300, CDate("#6/18/2003#"), CDate(#6/18/2003
11:59:59 PM#))
```

```
Dim sta4Det1 As New clsDetector(40, CDate("#6/18/2003#"), CDate(#6/18/2003 11:59:59
PM#))
Dim sta4Det2 As New clsDetector(400, CDate("#6/18/2003#"), CDate(#6/18/2003
11:59:59 PM#))

'Add stations to the arraylist of stations that will comprise the road
stations.Add(New clsStation(111, "TestStation1"))
stations.Add(New clsStation(222, "TestStation2"))
stations.Add(New clsStation(333, "TestStation3"))
stations.Add(New clsStation(444, "TestStation4"))

'Add detectors to each station
CType(stations(0), clsStation).AddDetector(sta1Det1)
CType(stations(0), clsStation).AddDetector(sta1Det2)
CType(stations(1), clsStation).AddDetector(sta2Det1)
CType(stations(1), clsStation).AddDetector(sta2Det2)
CType(stations(2), clsStation).AddDetector(sta3Det1)
CType(stations(2), clsStation).AddDetector(sta3Det2)
CType(stations(3), clsStation).AddDetector(sta4Det1)
CType(stations(3), clsStation).AddDetector(sta4Det2)

testRoad.AddStationListToRoad(stations) 'Add station list to the road
testRoad.ReferenceDataFile(datafile) ' reference the .traffic datafile
testRoad.StartDate() = CDate("#6/18/2003#") 'set the start date and time
testRoad.EndDate = CDate(#6/18/2003 11:59:59 PM#) 'set the end date and time
vol = testRoad.VolumeData 'retrieve the volume for the road
occ = testRoad.OccupancyData 'retrieve the occupancy for the road

The rest of the codes are simply printing the data
buf = "Volume and occupancy of the road on " + testRoad.StartDate.ToShortDateString
+ vbCrLf
buf += "index,"
For i = 0 To stations.Count - 1
   If i = stations.Count - 1 Then
       buf += CType(stations(i), clsStation).StationNumber.ToString + ".vol," +
CType(stations(i), clsStation).StationNumber.ToString + ".occ"
   Else
       buf += CType(stations(i), clsStation).StationNumber.ToString + ".vol," +
CType(stations(i), clsStation).StationNumber.ToString + ".occ,"
   End If
Next

For j As Integer = 0 To vol(0).Length - 1
   For i = 0 To stations.Count - 1
       If i = 0 Then
          buf += vbCrLf + CStr(j + 1) + "," + CType(vol(i)(j), Short).ToString +
"," + CType(occ(i)(j), Double).ToString("0.00")
       Else
           buf += "," + CType(vol(i)(j), Short).ToString + "," + CType(occ(i)(j),
Double).ToString("0.00")
       End If
    Next
 Next


If buf Is Nothing Then
    TextBox1.Text = "No data returned"
```

```
Else
     Dim objFileWriter As New IO.StreamWriter("testRoad.csv")
     objFileWriter.Write(buf)
     objFileWriter.Flush()
     objFileWriter.Close()
     Shell("notepad " & Application.StartupPath & "\testRoad.csv")
            TextBox1.Text = "Data sent out to file 'testRoad.csv' "
 End If
```

Retrieving a large amount of station and road data can be much more conveniently by providing station and detector numbers through an input data file. Example codes are includes in the sample code titled SimpleExample.