
This project designed and developed a system for web-enabling an existing technological device that captures data from commercial vehicle engines. A company based in South Carolina will use the prototype generated by this project for demonstration purposes and as a basis for future product development.

Development of the demonstration system required defining processes for packing and transporting data from the vehicle to the web server, coding a custom TCP server, and constructing a database and web-based user interface for viewing the data.

Headings:

Database – Management – Systems

Information – System – Design

Web Databases
THE DEVELOPMENT OF A WEB-BASED COMMERCIAL VEHICLE TRACKING AND MAINTENANCE SYSTEM

by

Bradley C. Argue

A Master's paper submitted to the faculty of the School of Information and Library Science of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Master of Science in Information Science.

Chapel Hill, North Carolina
May, 2001

Approved by:

___________________________
Advisor
Table of Contents

Project Description .............................................................................................................. 2
  Background ....................................................................................................................... 2
  Scope ............................................................................................................................... 3
  Resources ......................................................................................................................... 4
  Deliverables ...................................................................................................................... 5
  Timeline .......................................................................................................................... 6

Analysis .............................................................................................................................. 7
  Data Sequence Overview ............................................................................................... 7
  Process Description and Key Decisions ......................................................................... 9

Development ..................................................................................................................... 12
  Database ......................................................................................................................... 12
  TCP Server ..................................................................................................................... 13
  Website Interface ......................................................................................................... 13

Testing .............................................................................................................................. 15

Evaluation ......................................................................................................................... 16

Future Considerations ...................................................................................................... 17

Lessons Learned ............................................................................................................. 19

Bibliography .................................................................................................................... 20

Appendix A – Data Dictionary ....................................................................................... 21

Appendix B – ER Diagram ............................................................................................... 24

Appendix C – Database Schema .................................................................................... 25

Appendix D – Interface State Transition Diagram ....................................................... 26

Appendix E – SQL DDL Scripts (included on disk) ...................................................... 27

Appendix F – ASP scripts (included on disk) ................................................................. 27
Project Description

Background

Vehicle Enhancement Systems, Inc. of Rock Hill, South Carolina (VES) is an innovative developer of technological devices for commercial vehicles. One of VES’ products is VESplex, a commercial vehicle recorder that collects engine and related vehicle data, such as the governor setting, fluid levels, and fuel efficiency. Used in conjunction with a patented technology known as “IRIS” (Infrared Information System), VESplex uploads this information to a computer during servicing of the vehicle. Mechanics utilize custom software applications to analyze the raw data and draw conclusions about the vehicle’s condition and the habits and safety of its driver.

VES would like to further exploit the potential of these technologies by building an Internet-accessible application that stores vehicle data over time and generates customized, dynamic reports. Uploading vehicle data to a centralized server and storing it over time has a number of benefits. First, it simplifies and standardizes the process mechanics use to analyze vehicles in the shop. Currently, they must install and learn to use a separate software program for each engine maker (and in some cases, for each engine model). Secondly, by collecting data more frequently (e.g., utilizing wireless transfer capabilities to collect vehicle information every time a truck fills up at a truck stop) and tracking it over time, more intelligent preventative maintenance is possible.
For example, a truck that is consistently getting poor fuel efficiency may be brought in for maintenance sooner than its regularly scheduled interval.

VES plans to market their new Internet-based service under the name, “DataPlex.” VES has targeted two potential customers, a truck line and a fleet services company, for whom they would like to develop a pilot version of the product. VES hopes to sell these companies on the technology and possibly share development costs for marketing and rolling out the production service.

Scope

I agreed to develop a database and sample web interface for VES’ pilot product during the 2001 spring semester. I consulted with VES to define the parameters of the data that is stored in the DataPlex database and the web-based user interface requirements. Procedures for packing the vehicle data as well as unpacking and importing this information into the database were established in cooperation with an engineer at VES. A database structure that supports the information provided by the truck engine simulator developed by VES was created. Efforts were also made to develop the database so that it can scale to incorporate vehicle data that may be added in the future. For example, driver and trailer information is not currently available from the data on the engine bus. VES would like DataPlex to be able to track this information in the future, so the database structure I developed supports this information as well. Procedures for exporting information in a standard format were provided as well as a password-protected web application that generates some of the most commonly needed reports. Figure 1 represents the scope of the project in the context of the entire DataPlex system.
Figure 1 – DataPlex System and Project Scope

Resources

A variety of resources were required for the successful completion of this project. Once the scope of the project was determined, it was estimated that approximately 100 hours would be needed to develop and document the portions of the system for which I was responsible. Additional VES human resources were required to concurrently build a prototype VESPlex processor capable of packing and sending data according to agreed upon specifications. Several VES associates gave input during development of system specifications. In particular, the VES engineer assigned to create the DataPlex-capable VESPlex prototype was instrumental in helping to determine what truck data should be included in the prototype system and how it should be packed for transport between the truck and web server.
An NT-based file/web server at the UNC School of Information and Library Science (SILS) was utilized as a development machine for this project. Technologies used in development on this server included (ActiveState) Perl for building a TCP server to parse incoming data, Microsoft SQL Server, and Microsoft Internet Information Server (IIS) and Active Server Pages (ASP) for creating a web-based user interface. TAL Technologies’ TCP/COM software application was chosen as a means of redirecting data on the VESPlex Base.

**Deliverables**

A number of deliverables were promised at the conclusion of this project. They included:

- Specifications for packing and transporting data from truck to database.
- Specifications for design of a scaleable database that could hold truck data.
- A database and associated DDL scripts necessary to create it.
- A functional real-time data transfer process including the application code for receiving and importing incoming data on the server.
- A web interface incorporating the following minimum components/functionality:
  - Web Pages
    - Login page that accepts username (email) and password.
    - Options for selecting various reports per truck and per company.
  - Reports
    - Table displaying all data from the most recent data upload for a particular truck.
- List of a particular company’s trucks that are not currently set at the correct MaxSpeed.
- List of a particular company’s trucks that have an average fuel economy of less than a specified number of miles per gallon.
- Ability to download data for an individual truck in comma-separated values (CSV) format.

- All documentation and code associated with the items above.

**Timeline**

All work on this project was completed during the 2001 spring semester. The timeline shown in Table 1 was approved in early January 2001.

**Table 1 – Development Timeline**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
<th>Completed Task</th>
<th>Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial</td>
<td>Approved project scope and timeline</td>
<td>Jan 9, 2001</td>
</tr>
<tr>
<td>2</td>
<td>Specification</td>
<td>Finalize procedures for importing data; database specifications; list of web forms &amp; reports needed</td>
<td>Feb 5, 2001</td>
</tr>
<tr>
<td>3</td>
<td>Database Build</td>
<td>Database build complete with sample data inserted</td>
<td>Feb 26, 2001</td>
</tr>
<tr>
<td>4</td>
<td>Prototype</td>
<td>Functioning real-time data input and simple web interface</td>
<td>Mar 11, 2001</td>
</tr>
<tr>
<td>5</td>
<td>Interface Build</td>
<td>All web forms, reports, and export functions created</td>
<td>Apr 16, 2001</td>
</tr>
<tr>
<td>6</td>
<td>Documentation</td>
<td>All documentation and final deliverables completed</td>
<td>May 12, 2001</td>
</tr>
</tbody>
</table>
Analysis

Data Sequence Overview

The first step in preparing for development of the DataPlex system was to clearly determine and analyze the necessary sequence through which data would flow from the truck to the server. The only assumptions at the start of the project were that the data would be packed by and sent from a VESPlex and would be transferred in real time to a web server database. The methods for transfer and number of steps along the way were determined during the course of the project.

Data on the backend of the DataPlex system passes through a sequence of five steps on its way from the truck to the web server database (see Figures 1 and 2). The VESPlex is located on the truck and reads data off of the engine bus. The VESPlex then transfers its captured data via infrared to a local VESPlex Base. The VESPlex Base contains a processor that converts the data to standard RS232 format that is read in through a PC serial port. From there, the data is routed across the Internet using TCP/IP. A server listening at a specific IP address parses the incoming data and inserts it into the database using an ODBC connection. A variety of factors contributed to the decisions made when adopting this particular procedure. A description of the chosen process and rationale behind each key decision follows.

Process Description and Key Decisions

In the DataPlex system, the VESPlex reads data off of the truck engine bus. Data traveling on the engine bus uses the J1708 protocol. For the sake of the prototype, it was determined that data would be captured every three seconds to impress observers with the
real-time capabilities of the system. Some of the major decisions that were faced early on in the project related to what data should be captured and how it should be packed for transport to the web server. VES ultimately settled on twenty-one data elements to be included in each upload. It was agreed that these would be packaged in the following order: VIN, trip distance, total distance, battery, total hours, speed, idle speed, rated speed, total revs, oil pressure, coolant temp, coolant, air temp, oil temp, fuel temp, fuel, instantaneous economy, average economy, brake status, max speed, and pedal position
User Interface Data Sequence

User requests/views web page

HTTP Request

Web server (IIS)

Request for ASP page

Web Application Server (ASP)

Database query

Database Server (MS SQL)

HTTP Response

HTML

Recordset

Figure 3 – User Interface Data Sequence Diagram

(see Appendix A for data definitions). Other data that will be needed in the future but that were deemed impractical or impossible to include in the prototype included active and inactive error codes, trailer VIN, and driver ID. Accommodations were made for these items, however, in the database design.

The VESPlex sends packed data to a VESPlex Base using IRIS infrared technology. On the base, the data is converted to RS232 format and subsequently received by the PC through a serial port. TCP/IP was chosen as the transport method for sending data to the web server because of the existing Internet communications infrastructure. The question of who would be responsible for sending the data to the server via TCP/IP was unclear early on. In initial discussions with VES, there was an assumption that their technicians would be responsible for sending data to a web server IP address. Ultimately, the scope of my project was expanded somewhat to incorporate the PC software that converts data from RS232 to TCP/IP packets as well.
I researched several possible methods of handling the serial port data on the PC. Solutions considered included development of a custom application written in C/C++ or Visual Basic as well as existing software packages. Given time constraints and the complexity of serial port programming, I recommended that VES use TCP/COM, an existing product made by TAL Technologies that provides flexible, automatic redirection of COM port data to IP addresses. TCP/COM retails for $259 (2 licenses).

The decision to simply redirect the packed data to an IP address resulted in a transfer of the burden of unpacking data to the server side. If a custom application would have been written for the PC, some of processing power could have been offloaded from the server. Additionally, this would have opened up the possibility of sending the data to the web server as an HTTP POST request that could then be processed by an ASP page, DLL, or some other web-accessible application on the server side. Instead, a TCP server was needed to listen on a dedicated port and process the incoming data. The advantage of centralizing the majority of the processing load was the ability to have relatively “dumb” terminals (i.e., the VESPlex Base and PC running TCP/COM) and a “smarter” server that can be flexibly modified without affecting the distributed pieces of the system.

Perl was selected for building the custom TCP server solution. Perl was chosen due to my prior programming experience with it, its platform independence, and the availability of existing code that could be customized for these purposes. Code from several Perl TCP servers was used as a basis for development (Cavalier, 2000; Golden, 1998; Wall, Christiansen & Schwartz, 1996, p. 349-351). The TCP server listens on an assigned TCP/IP port and unpacks the incoming data based on a data scheme template stored in the database. The unpacked data is then inserted into the database as a new
record. For the purposes of the prototype, only one scheme is used. In the future, the appropriate scheme will be selected based on the truck VIN (the VIN prefixes the packed data for this reason). Data scheme templates are provided for purposes of system scalability. Truck engine models offer varying sets of data that the VESPlex can capture. Thus, it is likely that in the future different configurations of data will be packed depending on the engine model and possibly company preferences as well.

Data template schemes are stored in the database in the following format:

\[\text{FieldName1,Type1,Length1,Resolution1,FieldName2,Type2,Length2,Resolution2,\ldots FieldNameN,TypeN,LengthN,ResolutionN}\]

where “FieldName” equals the name of the associated field in the database, “Type” refers to the data type, “Length” defines the length of the data in bytes, and “Resolution” specifies the resolution (used for numerical data only). The template lists the data in the order received. Supported data types are “ASCII”, “HexLSB” (Least Significant Byte first format), and “HexLSBSigned” for signed numbers. ASCII values may use Length to specify their length or indicate “var” if the length is variable and will be specified by a hex value preceding the string. ASCII values include “na” for Resolution to indicate that it does not apply.

Microsoft SQL Server was chosen as the database management system for a number of reasons. First, only SQL Server and Oracle were both robust enough to handle the quantity of data that will be produced by the DataPlex system and available in the SILS test environment. Secondly, a key business partner that was identified as a potential partner in future DataPlex development uses Microsoft products including Windows NT and SQL Server. Finally, SQL Server was chosen for its relatively low cost vis-à-vis Oracle. ODBC was chosen as the method for connecting to the database from the Perl
server due to its prevalent use on Microsoft Windows server platforms and the availability of Perl/ODBC libraries such as Perl DBI.

On the user interface side, DataPlex utilizes ASP and VBScript. ColdFusion was also seriously considered as an option for the interface application development. ASP was ultimately chosen because of its inclusion with IIS (and consequent cheaper cost) and because it is the tool most frequently used by the business partner mentioned above.

**Development**

Development of the DataPlex prototype occurred on three fronts. Database development took place during phase 3 of the development timeline; development of the TCP Server was accomplished in phase 4; and the user interface was built during phases 4 and 5 (see Table 1).

**Database**

Development of the DataPlex SQL Server database was based on the data dictionary, entity-relationship diagram, and database schema shown in Appendices A, B and C, respectively. Efforts were made to ensure the scalability of the database. For example, separate driver, trailer, and (active/inactive) code tables were created for future use despite the fact that they are not utilized in the prototype system. Also, company and scheme tables were separated out even though there is only need for a single company and scheme for purposes of the demonstration system. Incorporation of additional single and multi-value Data table fields will be easily accomplished as more engine data and data template schemes are required. Microsoft SQL Enterprise Manager client software was used to create and manipulate the database tables, relationships, and views. The
DDL SQL scripts needed to recreate the database structure will be submitted with this documentation in electronic format.

TCP Server

As previously noted, development of the DataPlex TCP Server was initially based on code from several publicly available Perl TCP Servers (Cavalier, 2000; Golden, 1998; Wall, Christiansen & Schwartz, 1996, p. 349-351). The TCP server listens for incoming data on a defined TCP/IP port (e.g., 2345 was used during testing) and forks a new process for each package of data received. The forked process unpacks the data based on the data template scheme associated with the particular truck and inserts it into the database. The program is able to unpack ASCII data of variable lengths as well as data that are in hexadecimal least-significant-byte-first (LSB) or most-significant-byte-first (MSB) format. Due to issues with the VESPlex, most data are packed in LSB format. The server also accepts a delimited list of “trusted” IP addresses for security purposes. Any connections attempted from non-trusted IP addresses are immediately terminated. The Perl code used in this application will be submitted with this documentation in electronic format.

Website Interface

A number of minimum user interface requirements were agreed upon after consultation with VES during the specification phase of development. These included:

- Web Pages
  - Login page that accepts username (email) and password.
  - Options for selecting various reports per truck and per company.
• Reports
  o Table displaying all data from the most recent data upload for a particular truck.
  o List of a particular company’s trucks that are not currently set at the correct MaxSpeed.
  o List of a particular company’s trucks that have an average fuel economy of less than a specified number of miles per gallon.
  o Ability to download data for an individual truck in CSV format.

Concurrent with this project, I was enrolled in the SILS User Interface Design course. The DataPlex user interface was the subject of a major usability project undertaken in the class. A user analysis, task analysis, expert evaluation, test plan, and extensive associated documentation were the product of this course (Argue, 2001a).

The DataPlex interface essentially consists of five “pages” or states (see Appendix D). Upon request, all pages except “Help/About DataPlex” check to see if the requester is logged in. If not, the requested page is noted and the user is presented with the login screen. After successful login, the user is taken to the page originally requested or, by default, the “Home” page. On the Home page users see a list of up to 20 of their company’s trucks. Commonly-referred-to truck data is displayed in a table format, including truck identifying information, time and date of the last data upload, the current governor (maximum speed) setting, and average fuel economy. All visible columns can be sorted ascending and descending by clicking on the column title. Clicking on the truck VIN brings up a page with all of the data included in the last upload for that truck. From this page, the user can download all historical data that was collected during a
specified period in CSV format. From the Home page, users are able to click on a link to modify preferences. The Preferences page shows suggested options that users will be able to set in the future. Among the options are the ability to select the columns of data displayed, record sort order, and number of records that appear by default on the Home page. At this time there is no ability to save these modifications.

Tools used in development of the DataPlex user interface included Adobe Photoshop, Macromedia Dreamweaver, and Allaire Homesite.

**Testing**

A round of testing was conducted on the Perl TCP Server at the end of the Prototype phase (see Table 1). Custom Perl TCP client software was used to send sample packed data to the server in real time. Testing was conducted with a single server and multiple client applications located at two separate physical locations. Testing included sending a single pack of data from a client to the server as well as sending multiple packs of data to the server in the course of one client session. Ultimately, up to three client sessions were initiated with the server simultaneously, with multiple packs of data being sent by each. The server output was monitored to verify proper functioning and successful insertion of the data was verified via the database after each test. It is imperative that testing of the data upload procedure takes place with a truck simulator and subsequently with an actual truck equipped with a VESPlex. Originally, this was expected to take place during the Prototype phase of the project, but as of this writing, a DataPlex-enabled VESPlex and base have not been completed by VES.
User interface testing was conducted on a variety of PCs with Microsoft Windows operating systems and both Internet Explorer 4.x+ and Netscape Navigator 4.x+ installed. Multiple users, including numerous students from the SILS User Interface Design class and VES employees, browsed the site searching for usability problems and provided both formal and informal feedback. Additional testing with the targeted user group is recommended. In the course of my User Interface Design class, I created a usability test plan that would be appropriate to employ as a means of further testing. In this plan, I outlined a qualitative study that would involve an interface walkthrough and “think aloud” with 10-12 targeted users at three different onsite locations (Argue, 2001b).

**Evaluation**

Evaluation of the success of this project will come from client feedback and implementation of a usability study. I plan to travel to VES headquarters in May 2001 to present the software and documentation produced during this project. Informal feedback will be gathered at that meeting, including a list of enhancements to be added in the next iteration of development. In addition, I am recommending that VES conduct the usability study mentioned above prior to further system development. The goal of this test plan is to uncover all major usability problems with the prototype DataPlex interface and to generate a list of potential enhancements to the system based on user feedback and suggestions. The findings of the study will be used to revise the user interface and aid in the system’s continuing iterative development process (Argue, 2001b).
Future Considerations

A number of considerations must be addressed before the DataPlex system is deemed ready for production. First, scaling the system is an issue of concern. While the database infrastructure should scale easily, many of the other system components need to be tested under real-world conditions in order to gauge how robust they are. In particular, the TCP Server may need to be rewritten given that Perl, an interpreted language, is generally slower than applications built with compiled languages. Depending on the frequency with which data is collected and the number of trucks stored in a single database, a more robust database management system such as Oracle may need to be considered in the future as well. All aspects of the server’s performance will need to be tested once there is a clearer estimate of the load that will be put on it in a production scenario. Ultimately, the need for separate, dedicated database and web application servers is likely. Where the server or servers will be hosted and maintained will need to be determined based on the financial and business-process tradeoffs associated with outsourcing versus internally supporting the DataPlex infrastructure. Whoever is selected to maintain the system will need to institute procedures for backing up the database and archiving old data, among other things.

Additional backend modifications and enhancements should be considered as well. The use of TCP/COM may be reconsidered. A custom written application that transfers some of the processing requirements from the server to remote hosts may be beneficial. Alternatively, there would be advantages to enhancing the VESPlex and/or VESPlex base so that they are TCP/IP capable and can directly upload information to the server via wireless communications. The decision about the format in which data is
packed will need to be revisited as well. Additional information preceding the packed data may add needed flexibility as support for multiple truck engines and companies are added to the DataPlex system in the future.

As mentioned previously, further testing needs to be conducted on the user interface. Focus groups or individual interviews with end users will give further insight into the appropriateness of design decisions and potential future interface enhancements. Security, from the user interface end, must be carefully considered before rolling out the production system. For example, the use of Secure Sockets Layer (SSL) is likely warranted for this application. Additional functionality, such as an administrative interface and ability for users to modify their personal and company information, needs to be constructed as well. Ultimately, users may want to tie the DataPlex system into existing truck maintenance and personnel systems. Whether DataPlex can be offered in conjunction with common, related information systems or if this work would have to be done on an ad hoc basis will need to be determined.

Other business-related issues remain outstanding as well. A competitive analysis needs to be conducted as well as careful consideration of potential business partners in this endeavor. VES will need to determine whether they want to pay for all of the development and marketing or share in both the upfront costs and the eventual profits. Intellectual property rights, profitability, capital expenditure, and risk will all come into play when considering these issues.
Lessons Learned

I learned a great deal through this project. Overall, it was a very positive experience and is one I would certainly choose if I were placed in the same position again. The timeline set at the beginning of the project was very helpful in gauging my progress along the way and was successfully followed throughout. The scope of the project was broad and yet turned out to be manageable. The variety of skills required throughout made the project very interesting and working with a complex data process from truck to server to browser gave me an excellent opportunity to be involved with many aspects of a technical product’s development. The hands-on, practical nature of the project was very appealing to me initially and served to motivate me and hold my interest throughout. Finally, this project provided a great deal of freedom, ability to explore skills I had not used extensively before, and the ability to work independently.

Most of the negative aspects of this project were due in large part to my physical distance from VES headquarters. Communication was difficult at times, particularly late in the project timeline. There were a number of extenuating circumstances on their end that made it more difficult to stay in close contact during the later stages of development. This had the most detrimental effects during development of the interface. Increased communication and more timely feedback from the client would have been helpful particularly during the Interface Build phase. Contact with potential end users throughout the entire project would have been appreciated as well. If I were to begin the again, I would be most likely to change this aspect of the project.
Bibliography


http://www.rocketaware.com/perl/perlipc/Using_open_for_IPC.htm#Internet_TCP_Clients_and_Servers

http://www.cs.uno.edu/~golden/Teach/server

Appendix A – Data Dictionary

**Company**—Table holding company information

- **CompanyID**—Unique company identifier
- **CompanyName**—Company name

**Data**—Table holding information sent in VESPlex upload

- **DataID**—Unique data upload identifier
- **AirTemp** (*Ambient air temp*)—Outside air temperature measured to .25 F°
- **AveEcon** (*Average fuel economy*)—Average fuel economy (over the past < minute) measured to 1/256 mpg
- **Battery** (*Battery voltage*)—Level at which battery is charged measured to .05 volts
- **BrakeStatus** (*Parking brake switch status*)—Parking brake status: on/off
- **Coolant** (*Coolant level*)—Coolant level measured as a percentage of full (.5% resolution)
- **CoolantTemp** (*Engine coolant temp*)—Temperature of engine coolant measured to 1.0 F°
- **DateTime**—Date and time data is inserted into the database
- **Fuel** (*Fuel level*)—Fuel level measured as a percentage of full (.5% resolution)
- **FuelTemp** (*Fuel temp*)—Fuel temperature measured to .25 F°
- **IdleSpeed** (*Idle engine speed*)—Number of revolutions per minute engine is programmed to idle at (.25 resolution)
- **InstEcon** (*Instantaneous fuel economy*)—Instantaneous fuel economy measured to 1/256 mpg
- **MaxSpeed** (*Max road speed limit*)—Maximum speed truck is programmatically allowed to go (i.e., governor setting) measured to .5 mph
- **OilPressure** (*Engine oil pressure*)—Engine oil pressure measured to .5 psi
- **OilTemp** (*Engine oil temp*)—Engine oil temperature measured to .25 F°
PedalPosition (Percent accelerator pedal position)—Accelerator pedal position measured in terms of percentage depressed (.4 resolution)

RatedSpeed (Rated engine speed)—Number of revolutions per minute engine is rated for (.25 resolution)

Speed (Engine speed)—Engine revolutions per minute (.25 resolution)

TotalDist (Total vehicle distance)—Total number of miles truck has traveled (i.e., odometer reading) measured to .1 miles

TotalHours (Total engine hours)—Total number of hours engine has run measured to .5 hours

TotalRevs (Total engine revs)—Total number of engine revolutions measured in 1,000s of revolutions

TripDist (Trip distance)—Number of miles truck has traveled on current trip measured to .1 miles

Driver—Table containing driver information

DriverID—Unique driver identifier

DriverName—Driver name

Fault—Table holding fault codes

FaultID—Unique fault code identifier

FaultDesc—Description of fault code meaning

FaultType—Possible values are Active or Inactive

Scheme—Table containing map for parsing VESPlex upload

SchemeID—Unique scheme identifier

SchemeDesc—Description of scheme

SchemeTemp—Scheme template.

(SchemeTemp is in format, FieldName1,Type1,Length1,Resolution1,FieldName2,Type2,Length2,Resolution2,...FieldNameN,TypeN,LengthN,ResolutionN, where FieldName equals the name of the associated field in the Data table, Type refers to the data type, Length defines the length of the data in bytes, and Resolution specifies the resolution (used for numerical data only). The template lists the data in the order received. Supported data types are “ASCII”,
“HexLSB” (Least Significant Byte first format), and “HexLSBSigned” for signed numbers. ASCII values may use Length to specify their length or indicate “var” if the length is variable and will be specified by a hex value preceding the string. ASCII values should include “na” for Resolution to indicate that it does not apply.)

**Trailer**—Table containing trailer information

- **TrailerVIN**—Unique trailer Vehicle Identification Number (VIN)
- **TrailerMake**—Trailer manufacturer
- **TrailerModel**—Trailer model
- **TrailerYear**—Trailer production year

**Truck**—Table containing truck information

- **VIN**—Unique Vehicle Identification Number (VIN)
- **TruckEngine**—Truck engine type/model
- **TruckMake**—Truck manufacturer
- **TruckMaxSpeed**—Truck’s correct governor setting
- **TruckModel**—Truck model
- **TruckYear**—Truck production year

**User**—Table containing web interface user information

- **Email**—User's unique email address
- **Password**—Password for accessing online system (6 characters minimum)
- **UserName**—Real name of user
Appendix B – ER Diagram
Appendix C – Database Schema

<table>
<thead>
<tr>
<th>Underline</th>
<th>primary key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Italic</td>
<td>foreign key</td>
</tr>
</tbody>
</table>

Company (CompanyID, CompanyName)

Data (DataID, VIN, TrailerVIN, DriverID, DateTime, AirTemp, AveEcon, Battery,
BrakeStatus, Coolant, CoolantTemp, Fuel, FuelTemp, IdleSpeed, InstEcon,
MaxSpeed, OilPressure, OilTemp, PedalPosition, RatedSpeed, Speed, TotalDist,
TotalHours, TotalRevs, TripDist)

Driver (DriverID, DriverFName, DriverLName)

Fault (FaultID, FaultDesc)

Generates (DataID, FaultID, FaultType)

Scheme (SchemeID, SchemeDesc, SchemeTemp)

Trailer (TrailerVIN, TrailerMake, TrailerModel, TrailerYear)

Truck (VIN, CompanyID, SchemeID, TruckEngine, TruckMake, TruckMaxSpeed,
TruckModel, TruckYear)

User (Email, CompanyID, Password, FName, LName)
Appendix D – Interface State Transition Diagram

Notes:
- All transitions via mouse click
- Browser navigation buttons can also be used
- Invalid login submission will result in an error message and return to Login page
- Logout/Login navigation on Help page is context sensitive
- All pages except Help redirect to Login if user has not yet logged in
Appendix E – SQL DDL Scripts (included on disk)

Appendix F – ASP scripts (included on disk)