

***CONNECTED VEHICLES CORE SERVICES (CVCS)***

***– Data Distribution Subsystem (DDS)***

**Operations Manual**

Version *<1.1>*

*<09/15/2017>*

|  |  |
| --- | --- |
| Client: | *FHWA Intelligent Transportation Systems Joint Program Office –*  *Ariel Gold* |
| Date: | *9/15/17* |
| Project Number: | *DTFH-61-15-R-00037* |
| Work Location(s): | *Michigan* |
| Team | *Leidos* |

**VERSION HISTORY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version #** | **Implemented By** | **Date** | **Approved By** | **Date** |
| 1.0 | *Matt Vahlberg, Erik Hamilton, Diane Newton* | *07/14/2017* | *<name>* | *<mm/dd/yy>* |
| 1.1 | *Matt Vahlberg, Erik Hamilton, Diane Newton* | *09/15/2017* | *<name>* | *<mm/dd/yy>* |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1 Introduction 4](#_Toc493575863)

[2 System Overview 4](#_Toc493575864)

[3 Software Summary 4](#_Toc493575865)

[3.1 Software Description 4](#_Toc493575866)

[3.2 Commercial Services and Software Components 5](#_Toc493575867)

[3.3 Developed Components for integration into a CV Environment 6](#_Toc493575868)

[3.4 System Architecture 8](#_Toc493575869)

[3.4.1 Rackspace 8](#_Toc493575870)

[3.4.2 Situation Data Clearinghouse (SDC) System 10](#_Toc493575871)

[3.4.3 Situation Data Warehouse (SDW) System 11](#_Toc493575872)

[3.4.4 Distribution System 13](#_Toc493575873)

[3.4.5 Web Applications 14](#_Toc493575874)

[3.4.6 Standalone Applications 18](#_Toc493575875)

[4 Running the Systems 22](#_Toc493575876)

[4.1 Initiate Forwarders 22](#_Toc493575877)

[4.1.1 Start the Virtual Machines (VMs) 22](#_Toc493575878)

[4.2 Building the Systems 23](#_Toc493575879)

[4.2.1 Upload Plugins 23](#_Toc493575880)

[4.2.2 Setup Data Models 23](#_Toc493575881)

[4.2.3 Build the System 23](#_Toc493575882)

[4.2.4 Start the System 24](#_Toc493575883)

[4.2.5 Verify 24](#_Toc493575884)

[4.3 Complete Forwarders 24](#_Toc493575885)

[4.3.1 Configure Addresses 24](#_Toc493575886)

[5 Monitoring 24](#_Toc493575887)

[5.1 Management Console and System Monitor 24](#_Toc493575888)

[5.2 Warehouse Tools 24](#_Toc493575889)

[5.3 Test Routine: Data Type ISD, VSD, etc. 25](#_Toc493575890)

[5.4 Notification Scheduling 25](#_Toc493575891)

[6 Troubleshooting 25](#_Toc493575892)

[6.1 Ticketing System 25](#_Toc493575893)

[6.2 Uptime Dashboard 25](#_Toc493575894)

[6.3 DigitalEdge Console 25](#_Toc493575895)

[6.4 AWS Console 26](#_Toc493575896)

[6.5 Rackspace Console 26](#_Toc493575897)

[6.6 Maintenance 26](#_Toc493575898)

[Appendix A: List of Acronyms 27](#_Toc493575899)

[Appendix B: Maintenance Plan 28](#_Toc493575900)

[B.1 Software Updates 28](#_Toc493575901)

[B.1.1 Updating Plugins 28](#_Toc493575902)

[B.1.2 Updating Software Libraries Directly 28](#_Toc493575903)

[B.1.3 Rebuild and Cycle Affected System 29](#_Toc493575904)

[B.1.4 Update Forwarders 29](#_Toc493575905)

[B.2 Continued Operations & Infrastructure Maintenance 29](#_Toc493575906)

[B.2.1 AWS VM Retirement 29](#_Toc493575907)

[B.3 Periodic Diagnostics 29](#_Toc493575908)

[B.3.1 DigitalEdge Management Console 29](#_Toc493575909)

[B.3.2 DigitalEdge System Monitor 30](#_Toc493575910)

[B.3.3 AWS Console 30](#_Toc493575911)

[B.4 Backup Procedure for the Situation Datawarehouse 30](#_Toc493575912)

[B.4.1 Volume Snapshots 30](#_Toc493575913)

[B.5 Maintenance Issues 30](#_Toc493575914)

[B.6 Security Monitoring and Patching 30](#_Toc493575915)

[B.6.1 Software 30](#_Toc493575916)

[B.6.2 Instance Access 31](#_Toc493575917)

[Appendix C: Background 31](#_Toc493575918)

[Appendix D: DigitalEdge™ 33](#_Toc493575919)

[D.1 Team 33](#_Toc493575920)

[D.2 Licensing 33](#_Toc493575921)

[D.3 Documentation 33](#_Toc493575922)

List of Tables and Figures

[Table 1. Dynamic Plugin Parameter Values 22](#_Toc493575672)

[Figure 1- Cloud Environments for the SDC and SDW 4](#_Toc493575928)

[Figure 2- Relationship between Rackspace and AWS 8](#_Toc493575929)

[Figure 3- Situation Data Clearinghouse DigitalEdge System 9](#_Toc493575930)

[Figure 4- Situation Data Warehouse DigitalEdge System 10](#_Toc493575931)

[Figure 5- Distribution DigitalEdge System 12](#_Toc493575932)

[Figure 6- ISD Message Creator Application 14](#_Toc493575933)

[Figure 7- TIM Message Creator Application 14](#_Toc493575934)

[Figure 8- Warehouse Tools Set 15](#_Toc493575935)

[Figure 9- Warehouse Tools Map Application 16](#_Toc493575936)

[Figure 10- Message Validator Application 17](#_Toc493575937)

[Figure 11- Advisory Situation Data Poller Test application 19](#_Toc493575938)

[Figure 12- Intersection Situation Data Poller Test Application 21](#_Toc493575939)

[Figure 13- SEMI Test Bed Architecture - Physical View 32](#_Toc493575940)

# 1 Introduction

This document provides information concerning operations of the Data Distribution Subsystem for the United States Department of Transportation (USDOT) Southeast Michigan Connected Vehicles Core Services (CVCS) project. Operating these systems requires periodic interaction with the underlying server-side systems, which run in Amazon Web Services (AWS). This document reflects the Version 2.3 CVCS systems developed by Leidos.

# 2 System Overview

The Situation Data Clearinghouse (SDC) and Situation Data Warehouse (SDW) are systems designed to accept, process, and disseminate vehicle situation data, intersection data, and traveler information data to authorized users.

The SDC is designed to provide situation and intersection data to subscribers in near real-time. The SDC validates data message formats and verifies that messages fall within the geographic region associated with the SDC. For messages satisfying the input criteria, the SDC makes them available via Java Messaging Service (JMS) topic subscriptions to authorized consumers for data falling within the subscriber’s desired geographic region.

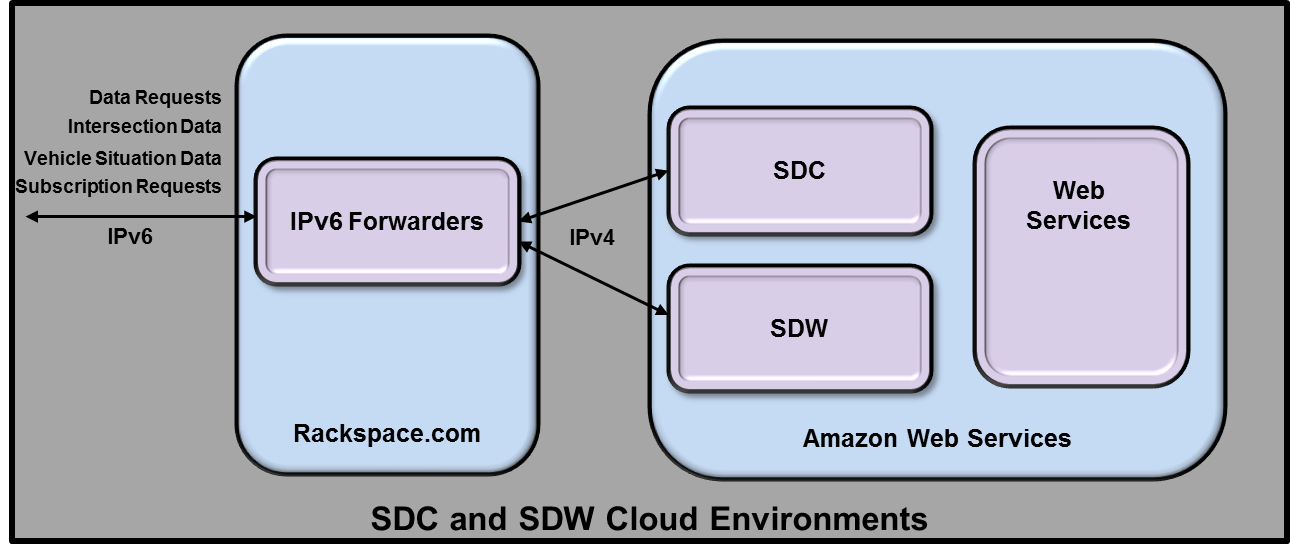
The SDW accepts the same message types as the SDC. In addition, the SDW accepts traveler information data. The SDW differs from the SDC in that the valid input messages are stored in a short-term database rather than made immediately available to subscribers. Messages are stored for a configurable amount of time before expiring. The SDW is currently set to age off messages after 30 minutes. Validated users can query stored messages from the database according to message type and geographic region.

A number of web applications and services are used to support the SDC and SDW. The applications support visualization of subscription data from the SDC and data queries into the SDW. The applications are enabled by web services running in AWS that connect to the SDC and SDW systems. These tools are useful for demonstrations and for monitoring and troubleshooting purposes.

# 3 Software Summary

## 3.1 Software Description

The SDC and SDW systems operate in the Amazon Web Services (AWS) and Rackspace Cloud environments as shown in Figure 1. Rackspace is used to accept, translate, forward, and distribute User Datagram Protocol (UDP) packets over Internet Protocol Version 6 (IPv6). This is the interface to the SDC and SDW from roadside units (RSUs), which receive vehicle situation data and intersection data from devices within range of the RSU. The forwarders running in Rackspace translate the packets from IPv6 to the more commonly used Internet Protocol Version 4 (IPv4) for data ingestion into the AWS cloud. Within AWS, the SDC and SDW are systems that run on the Leidos DigitalEdge framework[[1]](#footnote-1). DigitalEdge provides general-purpose ingest, processing, and data storage services for large, complex systems. Additionally, AWS provides web services for client applications that consume data from the SDC and SDW.

****

**Figure 1- Cloud Environments for the SDC and SDW**

## 3.2 Commercial Services and Software Components

As noted above, the Situation Data Clearinghouse (SDC) and Situation Data Warehouse (SDW) run in a public cloud infrastructure. The SDC/SDW environment also contains libraries and certificates that enable encoding and decoding per the ASN.1 UPER format and encrypting and decrypting IEEE1609.2 standard messages. This section will describe an overview and introduction for each of the various commercial services and software components, for more details refer to the Software Architecture section of this document.

*Amazon Web Services (AWS)* – Enables the operations of the SDC, SDW, and web applications via cloud infrastructure. The cloud resources utilized through AWS are Elastic Computing (EC2), Elastic Block Storage (EBS), Simple Storage Service (S3), and Route53.

* EC2 – Handles processing dialogs and situation data messages
* EBS – Stores the situation data before distribution in the SDC or stores the situation data for long-term uses by the SDW
* S3 – Stores the SDC and SDW configuration files, software components, and other miscellaneous artifacts, such as the Java keystore and truststore files to connect to the SDC
* Route53 – Provides DNS access to the services

*Rackspace* – AWS does not support UDP traffic over the IPv6 network, which is a requirement for receiving intersection and vehicle situation data. As such, Rackspace is used to forward situation data messages from the IPv6 network into the CVCS systems running in AWS on an IPv4 network. The cloud resource used here is the virtual cloud server, and it provides a dual stack network interface that supports both IPv6 and IPv4.

*DigitalEdge1 –* DigitalEdge is a big data platform developed by Leidos that runs in AWS. DigitalEdge runs the four systems that comprise the CVCS data distribution and support services subsystems. Included within DigitalEdge are management tools that support data modeling, system configuration and construction, system operation and status, and system performance monitoring. To meet the needs of a CV environment, the Leidos team created multiple running DigitalEdge systems.

*OSS ASN.1 Studio* – The SDC and SDW use a software library provided by OSS Nokalva to handle encoding and decoding the Connected Vehicle dialog and situation data messages in UPER format. The studio is a desktop application that comes with the OSS Nokalva software package and is used to generate the Java source which the SDC and SDW use in AWS. In order to use the studio, users must acquire the Connected Vehicle and Dedicated Short-Range Communication (DSRC) ASN.1 specification files.

* DSRC ASN.1 Specification - <http://standards.sae.org/j2735_201603/>
* Connected Vehicle ASN.1 Specification - <https://gitlab.com/connectedvehicles/fedgov-cv-asn1/blob/master/src/main/resources/SEMI_v2.3.0_070616.asn>

*1609.2 Certificates* – All messages into and out of the SDC and SDW will be encrypted following the IEEE 1609.2-2016 standard. Certificates for both the SDC and the SDW will need to be obtained from the Security Certificate Management System (SCMS) in order to decrypt any ASN.1 UPER messages coming into the SDC and SDW.

## 3.3 Developed Components for integration into a CV Environment

The native DigitalEdge platform alone does not meet the specific system needs for the SDC and SDW, thus, several plug-in components were developed using the DigitalEdge software development kit (SDK). DigitalEdge supports integration of custom components into the platform using a simple upload feature. Additionally, a number of web applications were developed for the Southeast Michigan Connected Vehicles project that integrated with the SDC and SDW. Below is a description of the various software components developed to ensure integration of SDC, SDW, and web tools into a real-world CV Test Bed environment.

*CVCS Components – These are components specifically developed for the USDOT Southeast Michigan project. While the following components in this subsection may be useful for others implementing similar services, it is important to note that they were developed for integration with a specific CV Test Bed. The components run in DigitalEdge to support specific functionality required for the SDC and SDW in AWS.*

* fedgov-cv-asn1: This plugin is generated from using the OSS Studio desktop tool using the Connected Vehicle and DSRC ASN.1 specification files. It also contains common utility classes used to assign the encoding and decoding of the Connected Vehicle messages.
* fedgov-cv-common: This plugin contains common Java classes used by the other projects.
* fedgov-cv-transport-udp: This plugin contains the data receiver component of the SDC and SDW. The transport handles all Connected Vehicle dialog messages.
* fedgov-cv-transport-listener: This plugin is used by the SDW to connect to the SDC’s Publish/Subscribe Server to listen for all vehicle and intersection situation data.
* fedgov-cv-parser-j2735: This plugin is for the SDC and SDW to parse vehicle, intersection, and traveler information situation data messages (Note: the traveler information messages go to the SDW only) and translate them according to mappings specified in the DigitalEdge data models.
* fedgov-cv-processor-broadcast-instructions: This plugin is used by the SDW to enrich the traveler information input data with broadcast instructions if the traveler information is of type RSU.
* fedgov-cv-processor-geojson: This plugin is for the SDW to enrich intersection and traveler information input data with GeoJSON information before it gets stored in the MongoDB database.
* fedgov-cv-datasink-router: This plugin is for the SDC to route vehicle and intersection situation data to the appropriate subscriber topic.
* fedgov-cv-datasink-subscription: This plugin is for the SDC to create subscriptions. The subscription information is stored on the DigitalEdge internal H2 database.
* fedgov-cv-mongodb: This plugin is for the SDW to store the vehicle, intersection, and traveler information situation input data in the MongoDB database.
* fedgov-cv-query-processor: This plugin is for the SDW to service query requests for intersection and traveler information messages.

*Standalone Applications – This is a set of test applications developed to send and receive data from the SDC and SDW for integration testing and demonstration purposes.*

* Java Swing applications
  + Vehicle Swing application: used to send mock vehicle situation data to the SDC and plots them on a map
  + Vehicle subscription Swing application: used to dynamically subscribe to the vehicle situation data from the SDC and to plot them on a map
  + Vehicle replay Swing application: used to directly query from the SDW MongoDB for vehicle situation data and to plot it on a map
  + Subscription Swing application: allows a user to send a subscription data request to the SDC and get back a subscriber id
* Java command-line applications
  + An application to poll for advisory broadcast messages from the SDW
  + An application to poll for intersection messages from the SDW
  + An application to send intersection situation data to the SDC
  + Forwarder application: installed on a Rackspace instance for forwarding UDP traffic over the IPv6 and IPv4 into and out of the SDC and SDW
  + System Monitor application: installed on a Rackspace instance to monitor the health and uptime of all the system’s components
  + Warehouse Client application: performs deposits/queries or subscribes for a running stream of data from a Warehouse This client can be embedded directly in Java code as well

*Web Applications: This is a set of web applications developed to visualize the situation data routed and stored in the SDC and SDW.*

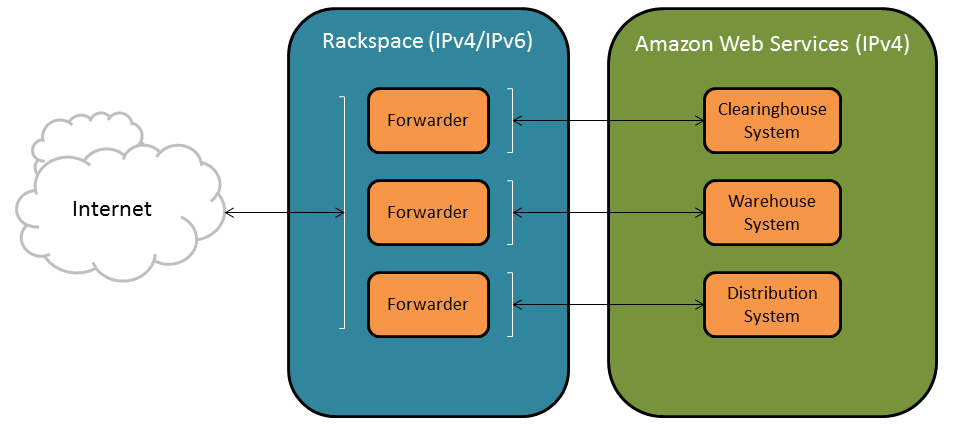
* Single Sign-on: This is the Central Authentication Service (CAS) web application. The application runs in the background and has no operational interaction. It provides a single sign-on mechanism for the other web applications.
* ISD Message Creator: This web application allows for the creation, editing, and encoding of Intersection MAP and SPaT messages.
* TIM Message Creator: This web application allows for the creation, editing, encoding, and deposit of Traveler Information messages.
* Warehouse Tools Set:
  + Warehouse Query: This web application provides a way to query the SDW for traveler information and situation data.
  + SDC Subscription: This web application provides a way to subscribe to the SDC and receive near-real time situation data.
  + Data Deposit: This web application allows direct deposit of messages into the SDC or SDW.
  + Map Application: This web application subscribes to the SDC for vehicle situation data and plots them on a map.
* Message Validator: This web application offers a way for third party integrators to check if their Connected Vehicle message is encoded and formatted correctly before sending to the SDC and SDW.

## 3.4 System Architecture

In order to meet the requirements of the SDC and SDW, three DigitalEdge systems run in the AWS cloud environment. The first system (Clearinghouse) ingests and processes vehicle and intersection situation data. The second system (Warehouse) ingests, processes, and stores vehicle, intersection, and traveler information situation data. The third system (Distribution) handles subscription requests and traveler information and intersection query data requests. The reason for the third system is to separate data requests from situation data deposits, so subscription and query requests can be fulfilled in a timely manner. Components developed specifically for the Southeast Michigan project are denoted in italic font. Native DigitalEdge components are identified in regular font. It should be noted that there is overlap on the various instances used per system. For ease of system use, each system has been separated out below.

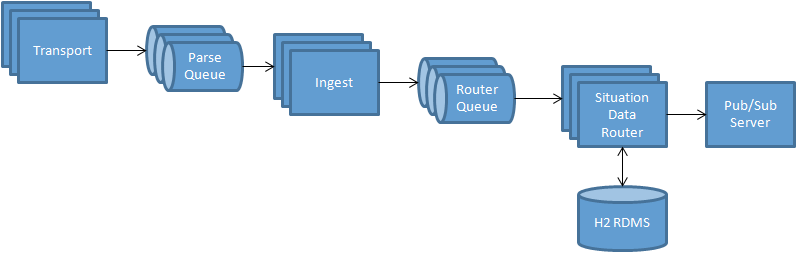
### 3.4.1 Rackspace

One of the requirements for the SDC and SDW is to receive situation data deposits and data requests via IPv6 networks using UDP. Because AWS does not support this capability, Rackspace.com is utilized since its virtual instance provides a dual stack IPv6/IPv4 network interface. As shown in Figure 2, Rackspace runs custom developed applications that handle data deposit and data request messages from both IPv6 and IPv4 networks and forwards them to the appropriate DigitalEdge system transport. In return, the application forwards responses back out, providing a single IP Address for third party applications to send and receive. All IPv6 input and output traffic must go through the Rackspace forwarder application. Optionally, the DigitalEdge systems in AWS can be configured to send directly to third party applications if the response is going out on the IPv4 network, for example the web applications utilize this approach.



**Figure 2- Relationship between Rackspace and AWS**

### 3.4.2 Situation Data Clearinghouse (SDC) System



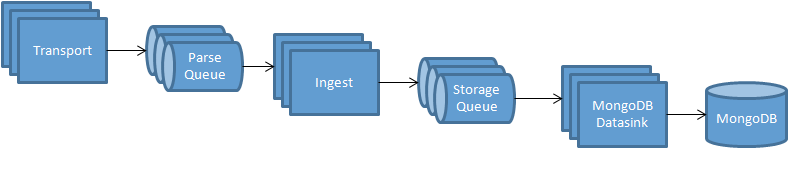
**Figure 3- Situation Data Clearinghouse DigitalEdge System**

The Clearinghouse System, shown in Figure 3, is a DigitalEdge system. Each box and cylinder in the diagram above represents a virtual DigitalEdge instance running in the AWS environment. The boxes and cylinders that are stacked indicate that one or more instance of the same type can run in parallel depending upon the configuration of the Clearinghouse system. The ability to create additional instances of the same type provides a way to horizontally scale the system to meet the performance requirements.

The Clearinghouse system receives vehicle and intersection situation data messages and routes them to the appropriate subscriber. Below is a description of each instance.

* Transport – This instance runs the *fedgov-cv-transport-udp* custom component. It receives and responds to vehicle and intersection situation data ASN.1 UPER dialog messages. When a VehSitDataMessage or IntersectionSituationData message is received, the transport will decrypt it and validate it to ensure that it is properly formed and within the specified service region of the system before placing it on the parse queue for processing. One or multiple Transport instances can be configured to run at once in the Clearinghouse.
* Parse Queue – This instance runs an ActiveMQ JMS Server and is responsible for storing the vehicle and intersection situation data until the Ingest instance picks it up for processing. This ensures that no messages are lost when the Ingest instance is busy. One or three of this type of instance can be configured to run at once in the Clearinghouse.
* Ingest – This instance type runs the *fedgov-cv-parser-j2735* and *fedgov-cv-processor-geojson* custom components, parses vehicle and intersection situation data messages, and maps extracted data fields into JavaScript Object Notation (JSON) records as configured in the appropriate DigitalEdge data model. Afterwards, a GeoJSON structure will be added to each data record. One or multiple Ingest instances can be configured to run at once or the Clearinghouse can be configured to scale the number of Ingest instances dynamically depending on the load of the system.
* Router Queue - This instance type runs an ActiveMQ JMS Server and is responsible for storing the vehicle and intersection situation data records until a Situation Data Router instance accepts it for processing. This ensures that no messages are lost when a Situation Data Router instance is busy. One or three Router Queues can be configured to run in the Clearinghouse.
* Situation Data Router – This instance type runs the *fedgov-cv-datasink-router* custom component and routes vehicle and intersection situation data to the appropriate subscriber topic on the Pub/Sub Server. It compares each situation data record to all active subscriptions and determine which ones match. When a subscription matches, the Situation Data router will encrypt the data using the subscriber’s 1609.2 certificate and place the encrypted message on the subscriber’s JMS topic. All subscription information is stored in the H2 RDMS instance, and the Situation Data Router periodically polls H2 to see if new subscriptions have been added or existing subscriptions removed. If a new subscription is added, the Situation Data Router creates the subscriber’s unique JMS topic on the Pub/Sub Server. One or multiple Situation Data Router instances can be configured to run at once, or the Clearinghouse can be configured to scale the number of instances dynamically depending on the load of the system. If more than one instance of this type exists, only the first instance will create or remove JMS topics for subscriptions.
* H2 RDBMS – This instance type runs the H2 relational database management system used internally by DigtalEdge and stores all the subscription information. Only one instance of this type exists for the Connected Vehicle tenant. The H2 RDBMS also supports the other two DigitalEdge systems.
* Pub/Sub Server – This instance type runs an ActiveMQ JMS Server and is responsible for pushing vehicle and intersection situation data to subscribers connected to a topic. Only one instance of this type exists in the Clearinghouse system.

### 3.4.3 Situation Data Warehouse (SDW) System

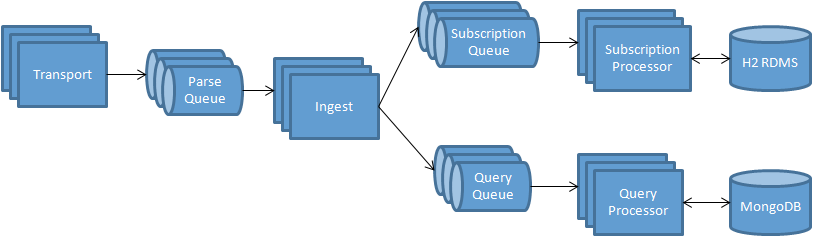


**Figure 4- Situation Data Warehouse DigitalEdge System**

The Warehouse System is the second DigitalEdge system, shown in Figure 4. Each box and cylinder in the diagram above represents a virtual DigitalEdge instance running in the AWS environment. The boxes and cylinders that are stacked indicate that one or more instance of the same type can run in parallel depending upon the configuration of the Warehouse system. The ability to create additional instances of the same type provides a way to horizontally scale the system to meet performance requirements for large amounts of input data.

* Transport - This instance type runs the *fedgov-cv-transport-udp* and *fedgov-cv-transport-listener* custom components. It receives and responds to traveler situation data ASN.1 UPER dialog messages. In addition, it handles vehicle and intersection situation data from the Clearinghouse system for storage into the MongoDB database. Additionally, the Transport processes AdvisorySituationData messages. When an AdvisorySituationData message is received, the transport decrypts and validates it to ensure that it is properly formed and within the specified service region of the system before placing it on the parse queue for processing. One or multiple Transport instances can be configured to run at once in the Warehouse.
* Parse Queue – This instance runs an ActiveMQ JMS Server and is responsible for storing the vehicle and intersection situation data until the Ingest instance picks it up for processing. This ensures that no messages are lost when the Ingest instance is busy. One or three of this type of instance can be configured to run in the Warehouse.
* Ingest – This instance type runs the *fedgov-cv-parser-j2735*, *fedgov-cv-processor-broadcast-instructions*, and *fedgov-cv-processor-geojson* custom components. It parses the vehicle, intersection, and traveler information situation data and maps extracted data fields into JSON records as configured in the appropriate DigtalEdge data model. If the message contains traveler information situation data and the distribution type is RSU, a broadcast instruction JSON structure will be added to the data record. If the message contains vehicle or intersection situation data, a GeoJSON structure will be added to the data record. One or multiple Ingest instances can be configured to run at once, or the Warehouse can be configured to scale the number of Ingest instances dynamically depending on the load of the system.
* Storage Queue - This instance type runs an ActiveMQ JMS Server and is responsible for storing vehicle, intersection, and traveler information situation data messages until the MongoDB data sink instance accepts them for processing. This ensures that no messages are lost when the Data sink instance is busy. One or three Storage Queues can be configured to run in the Warehouse system.
* MongoDB Datasink – This instance type runs the *fedgov-cv-datasink-mongodb* custom component, and it handles storage of the vehicle, intersection, and traveler information situation data messages into the MongoDB database. This instance creates the time-to-live and geospatial indexes for the “expireAt” and “region” fields of the data model. One or multiple MongoDB Datasinks can be configured to run at once, or they can be configured to scale up dynamically depending on the load of the system. If more than one MongoDB Datasinks exist, only the first instance will monitor and create the time-to-live and geospatial indexes.
* MongoDB – This instance runs the actual MongoDB database. The geospatial and time-to-live functionality provided by the database are used to meet the geospatial query and automatic removal requirements of the Connected Vehicle project.

### 3.4.4 Distribution System



**Figure 5- Distribution DigitalEdge System**

The third and final DigitalEdge system is the Distribution System, shown in Figure 5. Each box and cylinder in the diagram above represents a virtual instance running in the AWS environment. The boxes and cylinders that are stacked indicate that one or more instance of the same type can be configured to run depending upon the configuration of the Distribution system. The ability to create additional instances of the same type provides a way to horizontally scale the system to meet the performance requirements.

* Transport - This instance type runs the *fedgov-cv-transport-udp* custom component, and it receives and responds to subscription and data requests from ASN.1 UPER dialog messages. When a DataSubscriptionRequest, RsuAdvisorySituationDataRequest, or IntersectionSituationDataRequest message is received, the Transport will decrypt and validate it to ensure that it is properly formed and within the specified service region of the system before placing it on the parse queue for processing. One or multiple Transport instances can be configured to run at once in the Distribution system.
* Parse Queue - This instance type runs an ActiveMQ JMS Server and is responsible for storing the subscription and data request messages until the Ingest instance picks them up for processing. This ensures that no messages are lost when the Ingest instance is busy. One or three Parse Queue instances can be configured to run in the Distribution system.
* Ingest - This instance type runs the *fedgov-cv-parser-j2735* custom component. It parses the subscription and data request messages and maps the extracted information into JSON data records as per the associated DigitalEdge data model. One or multiple Ingest instances can be configured to run at once, or the Distribution system can be configured to scale the number of Ingest instances dynamically depending on the load of the system.
* Subscription Queue - This instance runs an ActiveMQ JMS Server and is responsible for storing subscription request JSON data records until a Subscription Processor instance can accept those records for processing. This ensures that no messages are lost when the Processor instance is busy. One or three Subscription Queue instances can be configured to run in the Distribution system.
* Subscription Processor – This instance type runs the *fedgov-cv-processor-subscription* custom component. It adds new subscriptions or actively and passively removes subscriptions from the H2 RDBMS. Active removal pertains to an action initiated by a subscriber via the DataSubscriptionCancel ASN.1 UPER message. Passive removal pertains to expired subscriptions either by the subscription end time reached or by expiration of the subscriber certificate. When a new subscription is added, the Subscription Processor assigns the subscriber an ID between 10,000,000 and 99,999,999; this is also the name of the topic the subscriber will connect to on the Pub/Sub Server. Only one Subscription Processor is recommended, but more than one can be configured.
* H2 RDBMS – This is referring to the same instance described for the Clearinghouse system. The Subscription Processor places a new subscription and actively (initiated by the client) or passively (expired) removes the subscription from the H2 RDBMS instance.
* Query Queue - This instance type runs an ActiveMQ JMS Server and stores data request situation JSON records until a Query Processor instance accepts them for processing. This ensures that no messages are lost when a Query Processor instance is busy. One or three Query Queue instances can be configured to run in the Distribution system.
* Query Processor - This instance type runs the *fedgov-cv-datasink-query-processor* custom component. It queries the MongoDB database for vehicle and intersection situation data. After receiving the query response, the Query Processor packages the result into one or more RsuAdvisorySituationDataBundle or IntersectionSituationDataBundle ASN.1 UPER messages depending on the query type. It encrypts the messages using the requestor’s 1609.2 certificate, and sends it back to the requestor over UDP. One or multiple Query Processor instances can be configured to run at once, or the Distribution system can be configured to scale the number of Query Processor instances dynamically depending on the load of the system.
* MongoDB – This is referring to the same MongoDB database instance specified in the Warehouse system. The Query Processor connects to the MongoDB instance to perform queries for advisory and intersection situation data.

### 3.4.5 Web Applications

#### Intersection Data (ISD) Message Creator

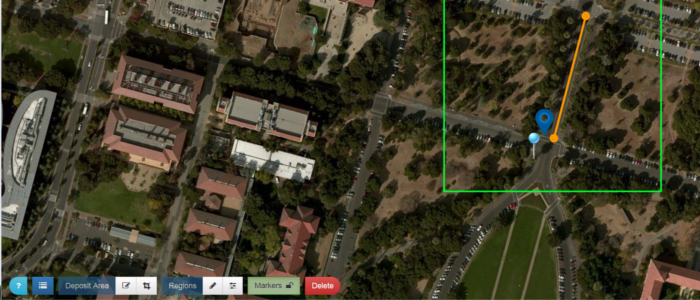
The ISD Message Creator allows a user to define the lanes and approaches of an intersection using a graphical interface. Once designed, the user can convert the message to ASN.1 or Hex.



**Figure 6- ISD Message Creator Application**

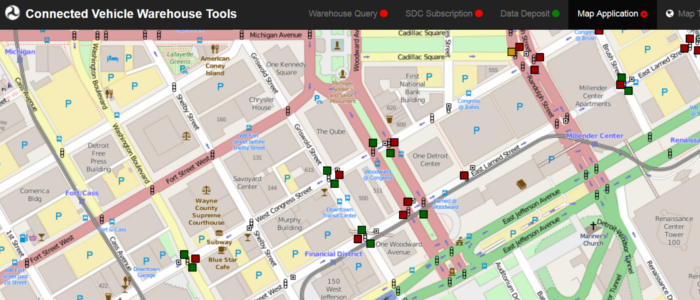
#### Traveler Information Message (TIM) Message Creator

The TIM Message Creator tool allows users to build traveler information messages regarding sign and work zone details using a graphical interface. Once designed, the user can convert the message to ASN.1 or Hex and deposit it to the SDW.



**Figure 7- TIM Message Creator Application**

#### The Warehouse Tool Set



**Figure 8- Warehouse Tools Set**

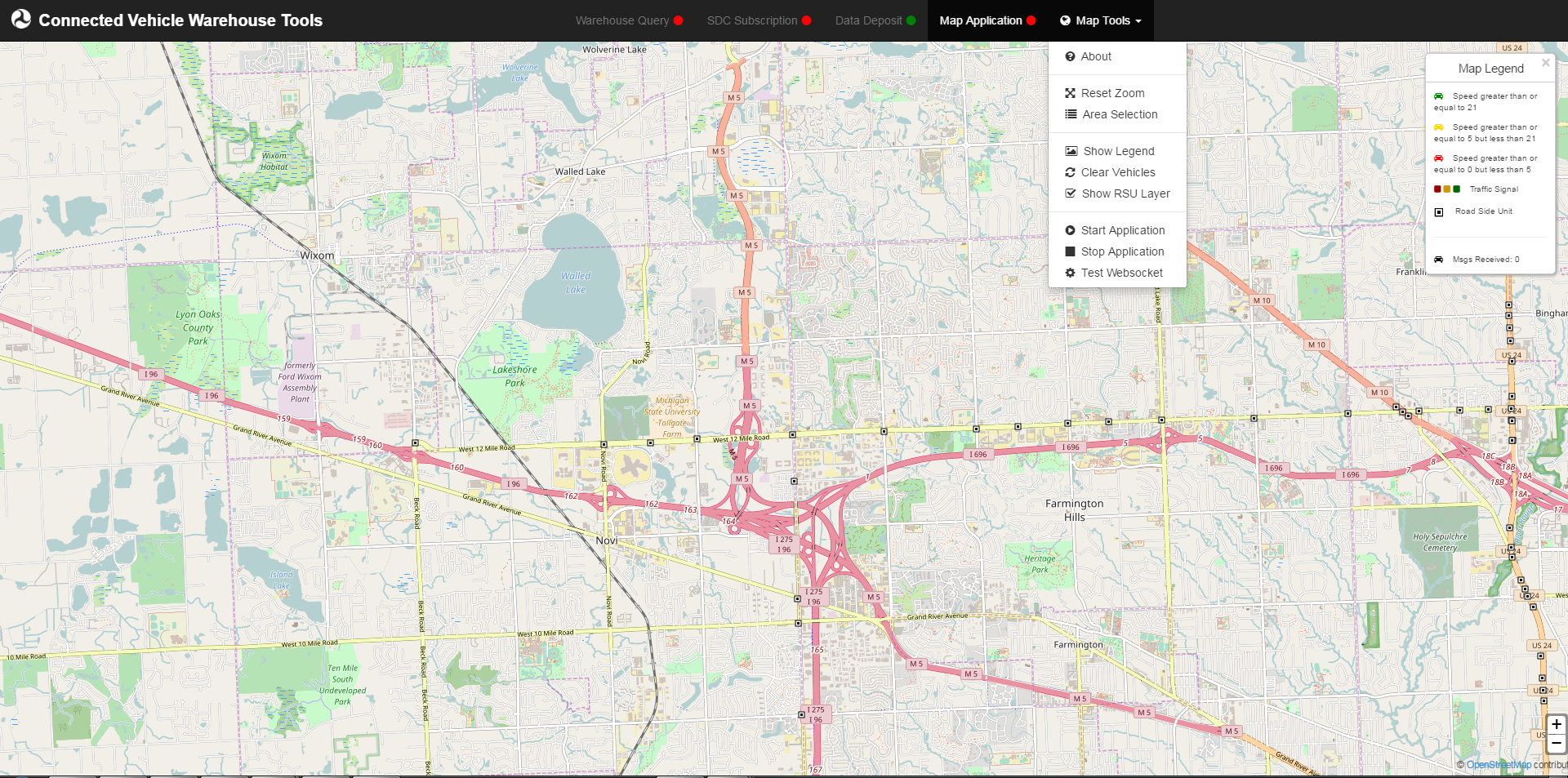
The Warehouse Tools set is a full suite that allows a user to subscribe, query, and deposit messages to the warehouses. There is also a mapping application that displays vehicle and signal information in real-time.

##### Warehouse Tools Map Application

The Map Application visualizes vehicle situation and intersection data from the SDC. The user interface (UI) connects to a WebSocket server running in AWS. The WebSocket server is a subscriber to the SDC that implements a special listen-all subscription that receives every deposited data input into the SDC. The UI can select information to display according to user-specified geospatial regions.

To use the UI, connect a web browser to the URL of the WebSocket server. The URL is: *https://webapp2.connectedvcs.com/whtools23/*. Upon connecting, the UI requires a valid username and password. Once valid credentials are entered, the application is listed as *Map Application* along the top navigation bar.

A screen shot of the application follows in Figure 9. Vehicle situation data is plotted on the map once a second for all vehicles transmitting situation data bundles falling within the assigned geographic region. Each one-second bundle is plotted on the map as a color-coded dot. Each dot disappears off the map 30 seconds later to avoid the display from becoming cluttered with old information. The 30-second period is the default setting but is customizable on the WebSocket server. The dot colors are red, yellow, or green according to the speed of the vehicle, which is also customizable on the WebSocket server.



**Figure 9- Warehouse Tools Map Application**

The default dot values, and those depicted on the above figure are as follow:

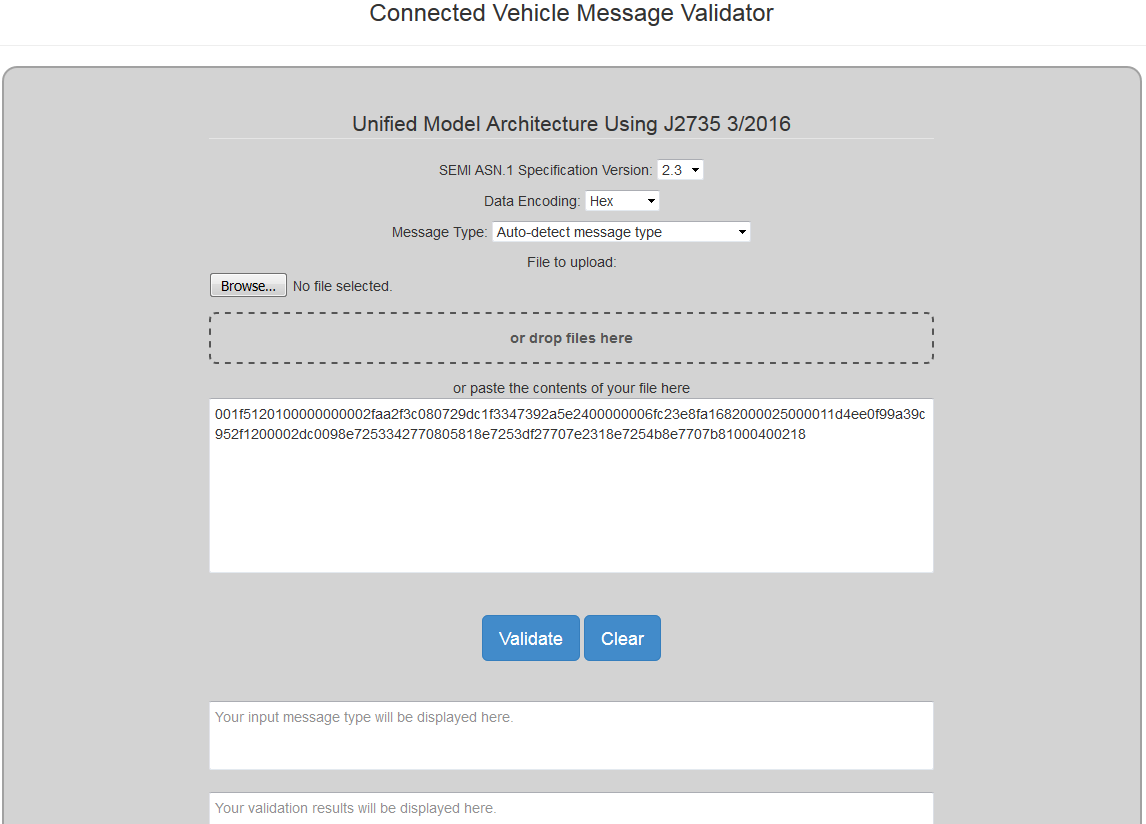
* Red for speeds less than 5 mph
* Yellow for speeds greater than or equal to 5 mph and less than 21 mph
* Green for speeds greater than or equal to 21 mph

Intersection data from traffic signals appear as color-coded squares by lane plotted at the geospatial coordinated provided in the metadata. The color of the square corresponds to the state of the traffic signal for each lane.

A single pull-down menu, the Map Tools menu, supports operation of the application. This menu allows users to start and stop the application, clear all data from the map, and to enable or disable the map layer contain the fixed-location RSUs. The Show/Hide Legend function toggles between whether or not the legend in the upper right corner of the screen shot is visible or not. The Area Selection menu option will open a dialog which allows a user to choose a custom geographic area in which to display SDC data. By default, the UI shows the greater Detroit area. The user has the option to switch to other pre-defined geographic areas or to specify a custom rectangular area defined by latitude and longitude of the rectangle corners.

#### Message Validator

The Message Validator tool is used to check UPER/ASN.1/HEX versions of messages for accuracy against the specifications and standards before depositing into a warehouse.



**Figure 10- Message Validator Application**

### 3.4.6 Standalone Applications

#### RSUAdvSitDataPoller

The RSUAdvSitDataPoller was developed to assist obtaining the advisory message and instructions on how to broadcast to the roadside unit where they are broadcast out to nearby vehicles. It also helps to test and demonstrate the ad-hoc query functionality of the SDW. The application was written in Java 1.6 and runs on the command line.

This application is located in the fedgov-cv-apps project and building the project produces the fedgov-cv-apps-<version>-jar-with-dependencies.jar file. Before running the application the settings.properties file needs to be configured and placed in the same folder as the jar file. Below is an example of the settings.properties file.

# The IP address of the situation data warehouse.

WarehouseIP=10.11.148.213

# The port number of the situation data warehouse.

WarehousePort=46754

# The port number on your localhost to receive warehouse response,

# must be different from the WarehousePort.

ResponsePort=46761

# The service region you are interested in advisory message(s).

NWLatitude=48.997519

NWLongitude=-125.595703

SELatitude=24.790723

SELongitude=-68.203125  
# The maximum age in minutes of data to return, this property is optional .

TimeBound=30

# The directory to write the output file on your localhost, this directory

# must be created before the application runs.

# SavePath=/tmp – example on a linux environment

SavePath=C:\\tmp

# The name of the output file.

Filename=out.txt

# The time in seconds between each poll requests. If this value is set too low

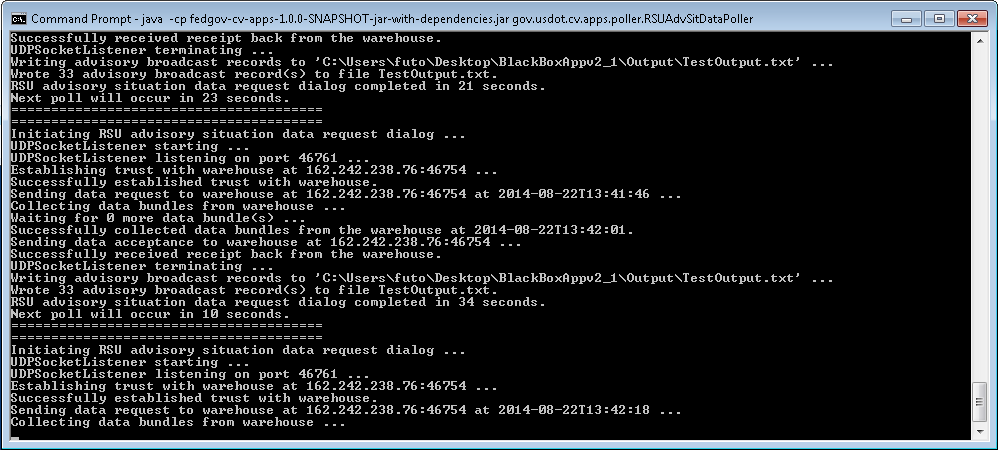
# the application will just poll immediately after finishing the current run sp

# set this value appropriately.

PollInterval=45

To run the RSUAdvSitDataPoller execute the following. Sample output appears in Figure 11.

java -cp fedgov-cv-apps-1.0.0-SNAPSHOT-jar-with-dependencies.jar gov.usdot.cv.apps.poller.RSUAdvSitDataPoller



**Figure 11- Advisory Situation Data Poller Test application**

The application will start up by loading in the configuration file and then submits an advisory situation data request to the SDW and continue to do so until the application is stopped. The result returned from the query will be written to the configured file with a single line representing an advisory message and its broadcast instruction. Below is an example of a single line output. The Payload property is the HEX encoded value of the traveler information message from the DSRC ASN.1 specification.

Version=0.5;Type=TIM;PSID=00008003;Priority=2;TxMode=ALT;TxChannel=178;TxInterval=1;DeliveryStart=07/21/2014, 20:06;DeliveryStop=07/28/2014, 20:06;Signature=true;Encryption=false;Payload=3081c68001108109000000000000003714830101a481ae3081ab800102a11ba119a0108004194fba1f8104ce45ce2382020a0681020006820102820207de830301c17084027d00850102a6108004194fc1988104ce45da4082020a008702016e880100a92430228002000ea21ca01aa31804040ce205a104040ada04f70404068004d60404034d0704aa3aa0383006a004800235293006a0048002010c3006a004800231283006a004800222113006a0048002010c3006a004800231203006a0048002221185021001;

#### IntersectionSitDataPoller

The IntersectionSitDataPoller was developed to test and demonstrate the ad-hoc intersection situation data query functionality of the SDW. The application was written in Java 1.6 and runs on the command line.

This application is located in the fedgov-cv-apps project and building the project produces the fedgov-cv-apps-<version>-jar-with-dependencies.jar file. Before running the application the settings.properties file needs to be configured and placed in the same folder as the jar file. Below is an example of the settings.properties file.

# The IP address of the situation data warehouse.

WarehouseIP=10.11.148.213

# The port number of the situation data warehouse.

WarehousePort=46754

# The port number on your localhost to receive warehouse response,

# must be different from the WarehousePort.

ResponsePort=46761

# The service region you are interested in intersection situation data.

NWLatitude=48.997519

NWLongitude=-125.595703

SELatitude=24.790723

SELongitude=-68.203125  
# The maximum age in minutes of data to return, this property is optional.

TimeBound=30

# The directory to write the output file on your localhost, this directory

# must be created before the application runs.

# SavePath=/tmp – example on a linux environment

SavePath=C:\\tmp

# The name of the output file.

Filename=out.txt

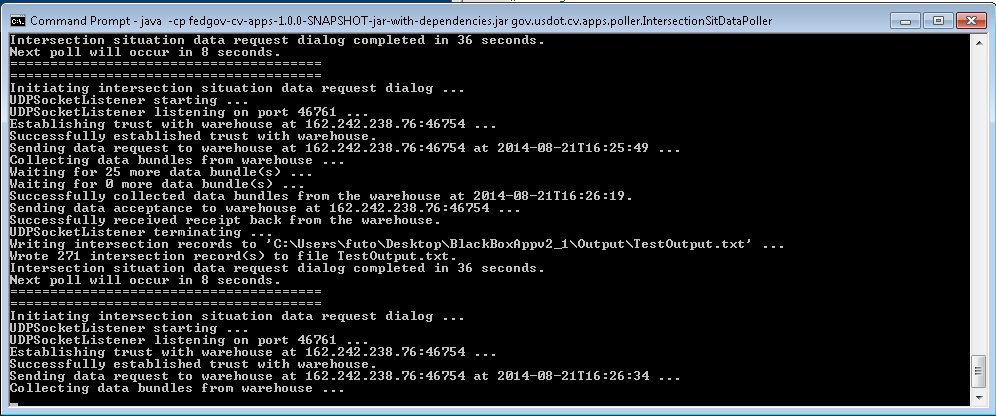
# The time in seconds between each poll requests. If this value is set too low

# the application will just poll immediately after finishing the current run sp

# set this value appropriately.

PollInterval=45

To run the IntersectionSitDataPoller execute the following. Sample output appears in Figure 12.

****java -cp fedgov-cv-apps-1.0.0-SNAPSHOT-jar-with-dependencies.jar gov.usdot.cv.apps.poller.IntersectionSitDataPoller

**Figure 12- Intersection Situation Data Poller Test Application**

The application will start up by loading in the configuration file and then submits an intersection situation data request to the SDW and continue to do so until the application is stopped. The result returned from the query will be written to the configured file with a single line representing the HEX encoded value of the intersection record. Below is an example of a single line output.

*3060a00a80010781010187020000a152a013800207dd81010c82010983010984011e85011ea13b801753616d706c6520496e74657273656374696f6e205b305d81040000035982010183022710840102a50a30088202010286022715860101870102*

# 4 Running the Systems

## 4.1 Initiate Forwarders

### 4.1.1 Start the Virtual Machines (VMs)

To initiate the forwarders, bring up two VMs in Rackspace: one for the SDC/ORDS, if it does not exist already, and the other for SDW. **Note the IPv4 addresses of each virtual machine (VM) as they will be required to configure the DigitalEdge Transport nodes later in the Building the System, System Builder section of this manual.** To log into the Rackspace account go to <https://mycloud.rackspace.com/>. A Rackspace account is required.

4.1.2 Install the Software

To install the forwarder software and scripts on the SDC/ORDS forwarder, create the following directories: /usr/local/cvData23 and /usr/local/cvData23\_secure. Upload fedgov-cv-apps-1.0.0-SNAPSHOT-jar-with-dependencies.jar, runcvDataNOSCREEN.sh, runcvData.sh, and the settings.properties file associated with the SDCForwarder-Nonsecure to the /usr/local/cvData23 directory. Upload fedgov-cv-apps-1.0.0-SNAPSHOT-jar-with-dependencies.jar, runcvDataNOSCREEN.sh, runcvData.sh, and the settings.properties file associated with the SDCForwarder-Secure to the /usr/local/cvData23\_secure directory.

To start the forwarder processes and setup the forwarder auto restart process on the SDC/ORDS forwarder, upload cvDataV23.conf and cvDataV23\_secure.conf to the /etc/init folder. Then run the following commands: “sudo start cvDataV23” and “sudo start cvDataV23\_secure”.

To install the forwarder software and scripts on the SDW forwarder, create the following directories: /usr/local/SDWv23 and /usr/local/SDWv23\_secure. Upload fedgov-cv-apps-1.0.0-SNAPSHOT-jar-with-dependencies.jar, runSDW23NOSCREEN.sh, runSDW23.sh, and the settings.properties file associated with the SDWForwarder-Nonsecure to the /usr/local/SDWv223directory. Upload fedgov-cv-apps-1.0.0-SNAPSHOT-jar-with-dependencies.jar, runSDW23\_secureNOSCREEN.sh, runSDW23.sh, and the settings.properties file associated with the SDWForwarder-Secure to the /usr/local/SDWv23\_secure directory.

To start the forwarder processes and setup the forwarder auto restart process on the SDW forwarder, upload SDWv23.conf and SDWv23\_secure.conf to the /etc/init folder. Then run the following commands: “sudo start SDWv23” and “sudo start SDWv23\_secure”.

## 4.2 Building the Systems

### 4.2.1 Upload Plugins

Please refer to the plugins tab of *CVCS Configuration Report 20170912.xlsx**[[2]](#footnote-2)* document for the full list of plugins required for the CVCS DigitalEdge Systems. Ensure that all the plugins have been uploaded into DigitalEdge using the DigitalEdge Management Console. Refer to the DigitalEdge Operations Guide[[3]](#footnote-3), Chapter 5, section *Managing the component repository* for instructions on uploading plugins to DigitalEdge.

### 4.2.2 Setup Data Models

For each CVCS system to be built (SDC, ORDS, etc.), please refer to the corresponding named System tab of the *CVCS Configuration Report 20170912.xlsx* document for the list of Data Models required for that system. Ensure that all required Data Models are uploaded to DigitalEdge. For instructions on uploading Data Models, refer to the DigitalEdge Configuration Guide[[4]](#footnote-4), Chapter 4, section *Upload and Download Data Models Between System.* After all required Data Models have been uploaded; open the Data Model using the DigitalEdge Data Model Editor to ensure that all Field Types and Translations match what is listed in the *CVCS Configuration Report 20170425.xlsx* document for each Data Model. Please refer to the DigitalEdge Configuration Guide2, Chapter 4, *Defining the Data Model* for detailed information about working with Data Models in DigitalEdge.

### 4.2.3 Build the System

For each CVCS system to be built, refer to the corresponding named System tab of the *CVCS Configuration Report 20170912.xlsx* document for the list of required plugins and system build parameters. For instructions on configuring and building a system, refer to the DigitalEdge Configuration Guide2, Chapter 5, *Building the System*. Please note that the systems should be built in the following order: SDC, ORDS, SDW, and finally SDPC.

Most plugin parameter values are static and can be entered into DigitalEdge System Builder exactly as they are shown in *CVCS Configuration Report 20170912.xlsx*, but the following values are dynamic and must be updated in DigitalEdge prior to building each system.

Table 1. Dynamic Plugin Parameter Values

|  |  |  |  |
| --- | --- | --- | --- |
| System | Plugin | Field | Value |
| SDC | CvUDPTransportService | forwader-host-name | IPv4 DNS address of SDC/ORDS Forwarder |
| SDW | JMSTopicListenerTransportService | target-host | IPv4 DNS address of PubSub Server |
| SDW | CvUDPTransportService | forwader-host-name | IPv4 DNS address of SDW Forwarder |
| SDPC | JMSTopicListenerTransportService | target-host | IPv4 DNS address of PubSub Server |

### 4.2.4 Start the System

After building a system, go to the Management Console, find the newly built system and press the start button. The systems cannot be started simultaneously, start one system at a time in the following order: SDC, then ORDS, then SDW, and finally SDPC. For instructions on using Management Console and starting a system, please refer to the DigitalEdge Operations Guide3, Chapter 5, *Managing DigitalEdge from the Management Console*.

### 4.2.5 Verify

When starting subsequent systems verify that the status of the system, that was just started, is OK before starting the next one. For details on system status, please see the DigitalEdge Operation Guide1, Chapter 6, *System Monitoring*.

## 4.3 Complete Forwarders

### 4.3.1 Configure Addresses

The settings.properties files on the Rackspace forwarders now have to be updated with the IP addresses of the Transport nodes of the new CVCS DigitalEdge systems.

On the SDC/ORDS forwarder update the /usr/local/cvData\*/settings.properties files. Set the “TransportIP” and “TransportIPPool” to the new public IP Address of the SDC Transport node.

On the SDC/ORDS forwarder update the /usr/local/cvDistribution\*/settings.properties files. Set the “TransportIP” and “TransportIPPool” to the new public IP Address of the ORDS Transport node.

On the SDW forwarder update the /usr/local/ SDW\*/settings.properties files. Set the “TransportIP” and “TransportIPPool” to the new public IP Address of the SDW Transport node.

After all settings.properties have been updated, restart the Forwarder VMs for the changes to take effect. The forwarders are configured to startup on boot.

# 5 Monitoring

## 5.1 Management Console and System Monitor

To manage and monitor the DigitalEdge systems, please refer to the DigitalEdge Operations Guide3 Chapter 5, *Managing DigitalEdge from the Management Console* and Chapter 6, *System Monitoring.*

## 5.2 Warehouse Tools

The *Warehouse Tools Set* allows a user to subscribe, query, and deposit messages into the warehouses. This also includes a mapping application that displays vehicle and signal information in real-time. The tools provide a way to send a unique test message through the systems to a specified warehouse. The tools can then be used to query for the test message to determine if a part of the data path is down, or if data is flowing completely through to the end. The tools themselves rely on a Websocket connection to the system. No connection is indicated by a red dot next to the tool’s name in the navigation panel at the top of the page, and a successful connection is indicated by a green dot. If the tools are unable to connect prior to deposit or query, that may indicate an issue with the Websocket server components of the systems.

## 5.3 Test Routine: Data Type ISD, VSD, etc.

To test ISD and VSD messages go to the Warehouse Tools Set and deposit a new unique message in the Data Deposit tab. For verification, check if the message has gotten through either by doing a Warehouse Query, an SDC Subscription, or just check on the Map Application.

To test TIM messages go to the Warehouse Tools Set and deposit a new unique message (Data Deposit). Check if the message has gotten through by doing a Warehouse Query.

## 5.4 Notification Scheduling

The System Monitor application will monitor message flow via test messages and detect downtime/system failures. When a failure is detected it will automatically create a Samanage incident report and distribute alert emails to a configurable list of users defined in the System Monitor’s configuration. For a detailed description of the System Monitor and it’s configuration options, please refer to the System Monitor Operations Manual. Upon recovery, it will send recovery emails to a different configurable list of emails. On the first day of every month, a report of components uptimes will be generated for the previous month and emailed to a third configurable list of emails.

# 6 Troubleshooting

## 6.1 Ticketing System

If a problem is detected, visit the Samanage portal and view the current system status by opening the incident report indicated on the alert email sent out.

<https://cvcs.samanage.com/login>

## 6.2 Uptime Dashboard

Go to CVCS monitoring dashboard to view the system status. When this is available, this should be the first step in troubleshooting. In the meantime, DigitalEdge console, and AWS console and Rackspace console should be used for reference.

## 6.3 DigitalEdge Console

One way to monitor the state of each system is to go to the Tenant Console <https://default.tms.aws-dev.deleidos.com/tenantconsole/>. Verify that the status of a system is OK; if in the Warning state please refer to the DigitalEdge Operations Guide3 under the sections *System Monitoring->Viewing system status* and *Troubleshooting*.

Another way to monitor each system is in DigitalEdge System Monitor <https://default.tms.aws-dev.deleidos.com/systemmonitor/>. Generally if there is zero for the Average Throughput or a very steep incline, the system is in a bad state. For more information please refer to the DigitalEdge Operations Guide3 under the section *System Monitoring->Checking system metrics*

## 6.4 AWS Console

In AWS Console <https://console.aws.amazon.com/console/home>, instances can be monitored by checking the Instance State of each Instance to verify that it is running. If an instance is in the warning state or if there is a scheduled downtime please refer to either the support center <https://console.aws.amazon.com/support/home?region=us-east-1#/> or the documentation for ec2 instances <https://aws.amazon.com/documentation/ec2/>.

## 6.5 Rackspace Console

Login to Rackspace console <https://mycloud.rackspace.com/> to verify that all of the servers are green. If there are any warning signs (not showing up green) or if there is any scheduled downtime, please click on the *Support* link on the console.

## 6.6 Maintenance

For maintenance of the systems, please refer to Appendix B: Maintenance Plan.

# Appendix A: List of Acronyms

|  |  |
| --- | --- |
| ARC-IT | Architecture Reference for Cooperative and Intelligent Transportation |
| AWS | Amazon Web Services |
| CVCS | Connected Vehicles Core Services |
| DB | Database |
| DSRC | Dedicated Short-Range Communications |
| EBS | Elastic Block Store |
| EC2 | Elastic Compute Cloud |
| IEEE | Institute of Electrical and Electronics Engineers |
| IP | Internet Protocol |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |
| ISD | Intersection Data |
| JMS | Java Messaging Service |
| RSU | Roadside Unit |
| SAE | Society of Automotive Engineers |
| SCP | Secure Copy Protocol |
| SDC | Situation Data Clearinghouse |
| SDK | Software Development Kit |
| SDW | Situation Data Warehouse |
| SSH | Secure Socket Shell |
| USDOT | United States Department of Transportation |
| UPD | User Datagram Protocol |
| VM | Virtual Machine |

# Appendix B: Maintenance Plan

## B.1 Software Updates

The software systems running within CVCS are operational systems. Therefore, determining reasonable software updates is and should be biased toward ensuring maximum uptime.

### B.1.1 Updating Plugins

Before updating plugins, please note that plugins are Java based software components and familiarity with Java will be required.

If an existing DigitalEdge plugin needs to be updated, or a new plugin needs to be created, refer to the DigitalEdge Software Development Kit (SDK) Guide[[5]](#footnote-5) for details on developing and updating a plugins. Once the updated or new plugin is ready it must be uploaded to the DigitalEdge repository before it can be used in a DigitalEdge system. Refer to the DigitalEdge Operations Guide3, Chapter 5, section *Managing the component* repository for instructions on uploading plugins.

### B.1.2 Updating Software Libraries Directly

Sometimes a software update is required for a library that isn’t a DigitalEdge plugin. In these situations it is necessary to access the DigitalEdge nodes directly. The nodes are accessible via a Secure Socket Shell (SSH). In order to do so, the security group for the instance needs to be opened via the AWS Console for the Internet Protocol (IP) of the machine accessing the instance[[6]](#footnote-6) and a copy of the public key from the key pair used to start to instance must be obtained. To upload new software onto the instance, any Secure Copy Protocol (SCP) client can be used. Once the instance is accessed and the new software library files are uploaded, the software update can be performed.`

The location of the software depends on which software is being updated. Any DigitalEdge software libraries can be found in the sub-directories under /usr/local/rtws/\*. The libraries will be located in “lib” directories and there will be scripts to stop and start the various services located in corresponding “bin” directories. On transport instances, DigitalEdge transport specific software is located in /usr/local/rtws/transport/\*. DigitalEdge datasink and enrichment specific software is located on their corresponding ingest instances under the directory /usr/local/rtws/ingest/\*. Non-DigitalEdge datasink software locations depend on how the instance is configured to install the software. For example, a MongoDB Datasink instance is configured to store the Database (DB) on a mounted volume under /mnt/rdafs/mongodb/. JMS instances key software is Apache ActiveMQ located at /usr/local/apache-activemq/\* with scripts to start and stop the service located under /usr/local/rtws/commons-core/bin/boot/.

In order to perform the update, after locating the correct software library, the service needs to be stopped, the correct library files need to be replaced with new copies, and the service needs to be started again. It is recommended to create backup copies of the original library files before replacing them with the new version. This will ensure old versions can be restored if there is an issue with the new software.

### B.1.3 Rebuild and Cycle Affected System

After new or updated plugins have been uploaded or a virtual machine (VM) has been retired, the effected DigitalEdge system(s) must be rebuilt and restarted for the software updates to take effect. To do this, first open the target system in System Builder. Add and/or update the required plugins (including plugin parameters) for the system and give the system a new unique name. This system, with the software updates and new name, will run in parallel with the old system until it can be confirmed the changes were successful. Build the new system and start it in the Management Console. For detailed instructions on building and starting a DigitalEdge system, please refer to the DigitalEdge Configuration Guide[[7]](#footnote-7), Chapter 5, *Building the System*.

### B.1.4 Update Forwarders

After starting the new system, the respective forwarder’s settings.properties files need to be updated. Refer to the *CVCS Configuration Report 20170912.xlsx2* file for the names and locations of these files. Change the TransportIP in each of the files from the public IP of the Transport of the old system to the public IP of the Transport of the new system. After doing this, restart the forwarder. To verify that data is getting through, use the webtools (such as Warehouse Tools Set). It is recommended to track Intersection Data (ISD) messages through the system to verify the software update has worked. If successful the old system can be stopped.

## B.2 Continued Operations & Infrastructure Maintenance

### B.2.1 AWS VM Retirement

Periodically, AWS needs to retire (shutdown, restart, etc.) one or more of the virtual machines (nodes) that make up a DigitalEdge system. If there is ever an issue with an AWS node, an email will be sent to the AWS account administrator with details about the node in question. In addition to the email, EC2 instances which require maintenance will have a scheduled event listed for them on the AWS Console. To view which EC2 instances have scheduled events, please refer to the AWS documentation[[8]](#footnote-8). When this happens, use the DigitalEdge or AWS Console to determine the DigitalEdge system(s) that will be affected. Rebuild and restart the affected system as discussed in sections 4.3 Rebuild and Cycle Affected System and 4.4 Update Forwarders.

## B.3 Periodic Diagnostics

There are a few places where diagnostics can be done for a running system. To view the status the DigitalEdge system as a whole, there are two tools available: the DigitalEdge Management Console and the DigitalEdge System Monitor. To view the health of individual instances that compose the systems, the AWS Console can be used.

### B.3.1 DigitalEdge Management Console

The DigitalEdge Management Console provides two ways to perform diagnostics for a system. The first is the system view list which provides a quick snapshot of a running system, including what process groups are running, the number of instances and types, and the number of volumes and sizes. The second tool is the log viewer which allows access to the logs of each node in a DigitalEdge system. The use of both of these is described in the DigitalEdge Operations Guide3, Chapter 6, in sections *Viewing system status* and *Checking log files*.

### B.3.2 DigitalEdge System Monitor

The DigitalEdge System Monitor is a dynamic console tool for monitoring system health and potential problems in real-time by visually depicting system activity and resource scaling. It provides the capability to see the throughput of ingest instances, the amount of data on the internal system queues, and the current storage capacity of the datasink instances. The use of the DigitalEdge System Monitor is outlined in the DigitalEdge Operations Guide3, Chapter 6, section *Checking system metrics*.

### B.3.3 AWS Console

The AWS Console can be used to check the health and status of the individual instances that compose the DigitalEdge systems. Using the AWS Elastic Compute Cloud (EC2) service, a web service that provides secure, resizable compute capacity in the cloud, each instance can be selected in order to show: its current state, a detailed description of the instance and its attributes, status checks, monitoring, and much more. Detailed documentation on how to use AWS EC2 is available through AWS EC2 Documentation[[9]](#footnote-9). For ease of system use, the AWS Elastic Block Store (EBS) is also utilized to provide persistent block storage volumes for use with AWS EC2 instances.

## B.4 Backup Procedure for the Situation Datawarehouse

In the SDW, the technology used to store the data is the NoSQL document database, MongoDB. The database is stored on RAIDed EBS volumes attached to the DigitalEdge datasink instance. Since the database is completely stored on volumes separate from the instance, this makes using an AWS EBS snapshot the ideal way to create a backup of the data.

### B.4.1 Volume Snapshots

Creating a volume snapshot is an easy process using the AWS console and is documented by AWS EC2 Documentation[[10]](#footnote-10). Since multiple volumes are used and RAIDed by the DigitalEdge system, it is necessary to complete the snapshot process for each volume on the instance. Once the process is completed, the snapshots will remain stored by AWS indefinitely until manually deleted from the AWS EC2 Console.

## B.5 Maintenance Issues

Maintenance issues are currently reported by customers using the ticketing system from the Samanage Service Desk[[11]](#footnote-11). The service desk is accessible from the “Support” link in the top right corner of the Connected Vehicles Tool Library page. Once a ticket is submitted, it will be addressed as quickly as possible and closed upon resolution.

## B.6 Security Monitoring and Patching

### B.6.1 Software

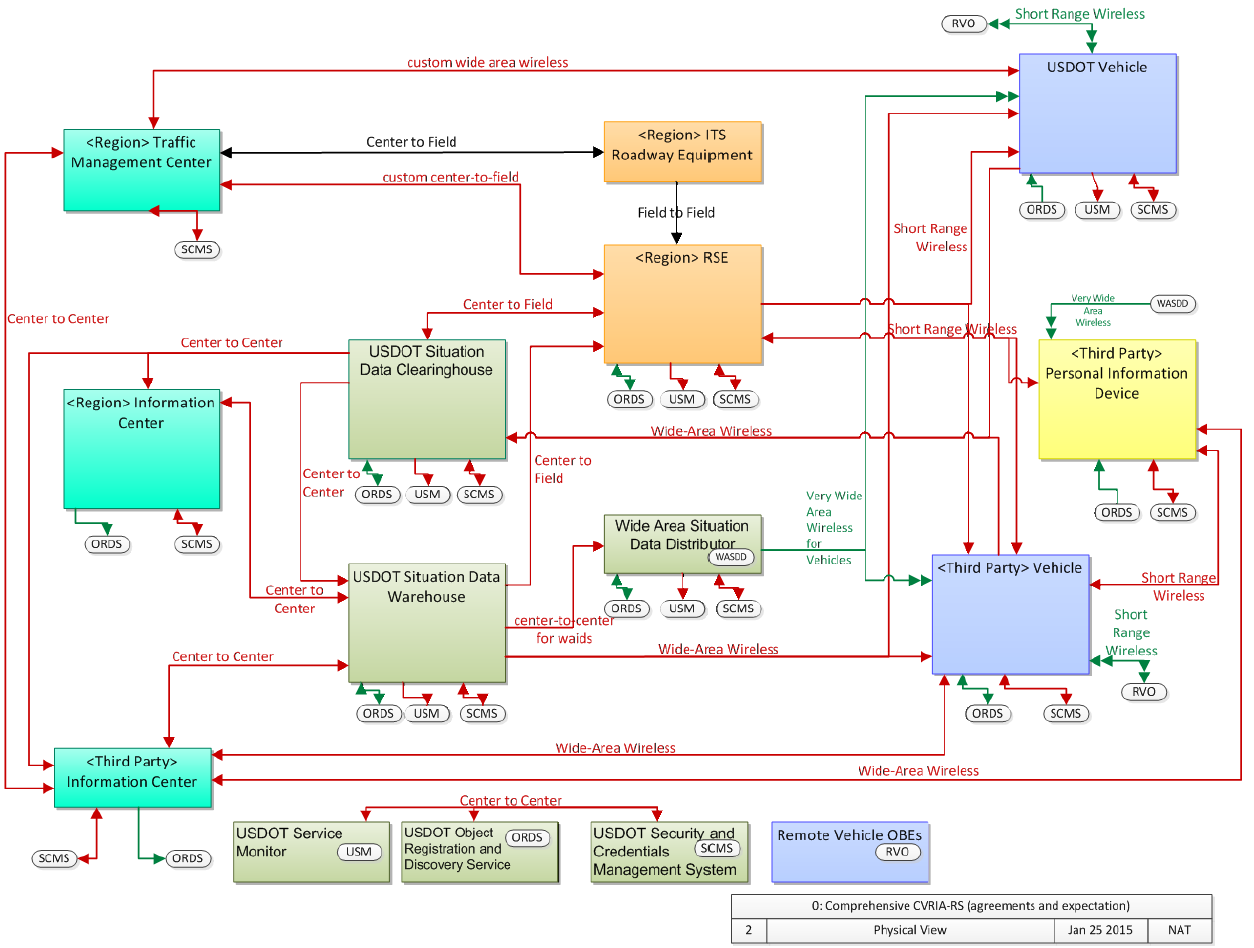
Due to the nature of Connected Vehicle environments, security must be built into every facet of a Connected Vehicle network. The SDC and SDW were developed using the security libraries, recommended by the USDOT and industry experts, in the following industry standards: IEEE 1609.2-2016 and SAE J2735. Because security is built directly into the software, any patches or updates to the security of the system can be applied using the, above, instructions for software updates in Section B.1.

### B.6.2 Instance Access

Security for the access of the DigitalEdge instances is provided by multiple means. When a new instance is launched, it is done so using the public key from an RSA key pair. This public key is controlled by AWS and not accessible to anyone else. In order to access an EC2 instance launched using the public key, a copy of the private key that was generated with the public key must be used for authentication. Without the private key, it is not possible to access the instance. In addition to the use of key pairs, AWS implements Security Groups. A Security Group is a virtual firewall that can be used to control traffic to instances. The Security Group maps IPs, IP ranges, or other Security Groups to specific port or port ranges. Each instance must be part of a Security Group, and only specified IPs or other instances in specified Security Groups will be allowed to access the instance on the specified ports. Therefore, the Security Group needs to be maintained to only ensure specific access to the instance. Description on how to configure the Security group can be found in AWS Documentation2.

# Appendix C: Background

In 2007, the United States Department of Transportation (USDOT) established a Connected Vehicle (CV) Test Bed in Oakland County, Michigan (known as both the Unified Implementation of the Reference Architecture and the Southeast Michigan Test bed – referred herein as the SEMI Test Bed) as a test facility for Proof of Concept (POC) testing of Dedicated Short-Range Communications (DSRC) using allocated bandwidth within 5.9 GHz. The SEMI Test Bed comprises all of the required elements for a functional V2V and V2I/I2V, 2016 standards compliant, real-world deployment. To date the SEMI Test Bed has been used as a testing facility for stakeholders to evaluate prototype equipment, applications, and services. In order to meet the evolving needs of the CV industry and remain a leader in the implementation of best practices and advances in technology and architecture, the Test Bed has implemented numerous enhancements, such as various approaches and improvements in data distribution systems, advances in mapping tools, and updates to the most up to date standards.

The Connected Vehicle Core Services discussed throughout this document can be found within the SEMI Test Bed. For further information on these or any other services within the SEMI Test Bed please refer to the Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT). This reference will provide the communications, physical, functional, and enterprise views of the SEMI Test Bed architecture. The below image, shows the physical view of the SEMI Test Bed architecture, capturing all of the physical components and their high-level interactions. The ARC-IT supporting tool, System Engineering Tool for Intelligent Transportation (SET-IT), also provides the SEMI Test Bed architecture as a sample project. 

**Figure 13- SEMI Test Bed Architecture - Physical View**

# Appendix D: DigitalEdge™

DigitalEdge is a cloud-based platform for creating big data solutions. It offers:

* *Scalable, pluggable, flexible, pre-integrated, data management platform* to enable quick implementation and simplified management of near real time (or batch) big data applications
* *Software* that leverages the power and dynamism of cloud computing to support rapid out-of-the-box creation and integration of IT systems
* *Data enrichment engine* that standardizes, integrates, and enhances diverse data sources to clean up imperfect data and transform it into information, rich with meaning.
* *Administration tools* that support rapid creation and spawning of systems that perform pre- ingest and enrichment

## D.1 Team

The DigitalEdge platform was developed and is maintained by Leidos. The DigitalEdge Team is comprised mostly of software engineers, based in the National Security Sector of Leidos, and contains a depth of experience ranging from college interns to 30+ years creating large scale enterprise systems.

The DigitalEdge Team prefers a Linux based environment, leveraging Docker in Amazon Web Services, Microsoft Azure, Eucalyptus and Rackspace. Favorite programming languages include Java and AngularJS.

## D.2 Licensing

DigitalEdge is provided for use under the Apache 2.0 Open Source License (https://raw.githubusercontent.com/deleidos/digitaledge-platform/master/LICENSE). The source information for DigitalEdge is available on GitHub at [https://github.com/deleidos/digitaledge-platform](https://webmail.leidos.com/owa/redir.aspx?SURL=hBHXcWQeF-nMmF0Jz8jDe4XkE0szysv0nJzt3mHNg9_KDPaNC-rSCGgAdAB0AHAAcwA6AC8ALwBnAGkAdABoAHUAYgAuAGMAbwBtAC8AZABlAGwAZQBpAGQAbwBzAC8AZABpAGcAaQB0AGEAbABlAGQAZwBlAC0AcABsAGEAdABmAG8AcgBtAA..&URL=https%3a%2f%2fgithub.com%2fdeleidos%2fdigitaledge-platform).

## D.3 Documentation

The below documentation may have been referenced in this manual.

* DigitalEdge Overview Guide
* https://github.com/deleidos/digitaledge-platform/raw/1.2.1/docs/Overview\_Guide.pdf
* DigitalEdge Operations Guide
  + https://github.com/deleidos/digitaledge-platform/raw/1.2.1/docs/Operations\_Guide.pdf
* DigitalEdge Configuration Guide
  + https://github.com/deleidos/digitaledge-platform/raw/1.2.1/docs/Configuration\_Guide.pdf

Additional documentation can be found at:

https://github.com/deleidos/digitaledge-platform/tree/1.2.1/docs

1. <https://github.com/deleidos/digitaledge-platform/> [↑](#footnote-ref-1)
2. <https://gitlab.com/connectedvehicles/fedgov-cv-documents/blob/master/CVCS%20Configuration%20Report%2020170912.xlsx> [↑](#footnote-ref-2)
3. <https://github.com/deleidos/digitaledge-platform/raw/1.2.1/docs/Operations_Guide.pdf> [↑](#footnote-ref-3)
4. <https://github.com/deleidos/digitaledge-platform/raw/1.2.1/docs/Configuration_Guide.pdf> [↑](#footnote-ref-4)
5. <https://github.com/deleidos/digitaledge-platform/raw/1.2.1/docs/Ingest_Pipeline_SDK.pdf> [↑](#footnote-ref-5)
6. <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/authorizing-access-to-an-instance.html> [↑](#footnote-ref-6)
7. <https://github.com/deleidos/digitaledge-platform/raw/1.2.1/docs/Configuration_Guide.pdf> [↑](#footnote-ref-7)
8. <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring-instances-status-check_sched.html#viewing_scheduled_events> [↑](#footnote-ref-8)
9. <https://aws.amazon.com/documentation/ec2/> [↑](#footnote-ref-9)
10. <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-creating-snapshot.html> [↑](#footnote-ref-10)
11. <https://cvcs.samanage.com/> [↑](#footnote-ref-11)