

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

***– System Monitor (SM) for Connected Vehicle Environment***

**Operations Manual**

Version *<1.2>*

*<09/28/2017>*

|  |  |
| --- | --- |
| Client: | *FHWA Intelligent Transportation Systems Joint Program Office –*  *Ariel Gold* |
| Date: | *9/28/17* |
| 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* | *09/20/2017* | *<name>* | *<mm/dd/yy>* |
| 1.2 | *Matt Vahlberg, Erik Hamilton* | *09/28/2017* | *<name>* | *<mm/dd/yy>* |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1 Introduction 1](#_Toc493675112)

[2 System Overview 1](#_Toc493675113)

[3 Software Summary 1](#_Toc493675114)

[3.1 Software Description 1](#_Toc493675115)

[3.2 Commercial Services and Software Components 2](#_Toc493675116)

[3.3 Developed SM Components for Integration into a CV Environment 3](#_Toc493675117)

[3.4 System Architecture 3](#_Toc493675118)

[3.4.1 Rackspace 4](#_Toc493675119)

[3.4.2 System Monitor (SM) System 4](#_Toc493675120)

[4 System Monitor 5](#_Toc493675121)

[4.1 Description 5](#_Toc493675122)

[4.2 Deployment 5](#_Toc493675123)

[4.2.1 Automatic Deployment 5](#_Toc493675124)

[4.2.2 Manual Deployment 6](#_Toc493675125)

[4.3 Custom Configuration 6](#_Toc493675126)

[4.4 Monitor Message Format 6](#_Toc493675127)

[4.4 Starting the Monitor 7](#_Toc493675128)

[Appendix A: Message Flow Diagram 8](#_Toc493675129)

[Appendix B: Monitor Messages Group ID Definitions 9](#_Toc493675130)

[Appendix C: Instances Accessed By SM 10](#_Toc493675131)

[Appendix D: Test Path Coverage 11](#_Toc493675132)

[D.1 Sending Messages 11](#_Toc493675133)

[D.2 Subscriptions 11](#_Toc493675134)

[D.3 Querying Databases via WebSocket 11](#_Toc493675135)

[Appendix E: List of Acronyms 12](#_Toc493675136)

[Appendix G: Background 13](#_Toc493675137)

List of Tables and Figures

[Table 1. Monitor Messages Group ID Definitions 9](#_Toc493675012)

[Table 2. Instances Accessed by SM 10](#_Toc493675013)

[Table 3. Test Path Coverage for Sending Messages 11](#_Toc493675014)

[Table 4. Test Path Coverage for Subscriptions 11](#_Toc493675015)

[Table 5. Test Path Coverage for Querying Databases via WebSocket 11](#_Toc493675016)

[Figure 1- Cloud Environments for the SM 2](#_Toc493675232)

[Figure 2- System Monitor System 4](#_Toc493675233)

[Figure 3- Message Flow Diagram 8](#_Toc493675234)

[Figure 4- SEMI Test Bed Architecture - Physical View 14](#_Toc493675235)

# 1 Introduction

This document provides information concerning operations of the System Monitor for the United States Department of Transportation (USDOT) Southeast Michigan Connected Vehicles Core Services (CVCS) project. Operating this system requires periodic interaction with the underlying server-side system, which runs in the Rackspace Cloud environment. This document reflects the Version 2.3 CVCS system monitor developed by Leidos.

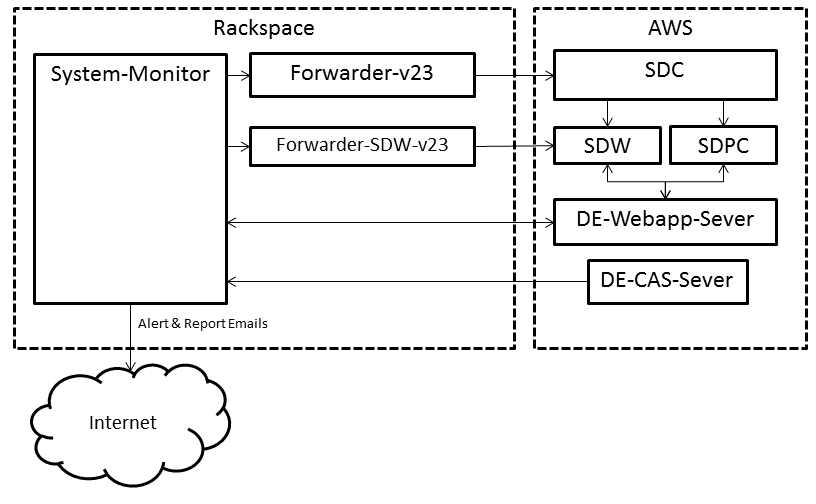
# 2 System Overview

The System Monitor (SM) is a system designed to monitor, track, and report the uptime of Data Distribution Subsystems (DDS) and Support Services Subsystems (SSS) components. It is designed to issue an email alert to a configured list of users and to generate a ticket to an alert monitoring system when an error is detected so that the issue can be inspected and addressed. For the purposes of CVCS, the Samanage platform was used for the alert monitoring system. The SM allows for configuration on how quickly it can detect an issue, the default configuration is to detect an issue within 3 minutes of the problem arising. Additionally, the SM maintains a history of the monitored components’ uptime. This history is distributed to a configured list of users on the first day of each month.

# 3 Software Summary

## 3.1 Software Description

The SM operates in the Rackspace cloud environment and interacts with DSS and SSS components in the Amazon Web Services (AWS) cloud environments as shown in Figure 1. The SM exists in Rackspace in order to monitor the Forwarder’s capability to receive packets using both Internet Protocol Version 6 (IPv6) and Internet Protocol Version 4 (IPv4). The packets are then sent to and processed by the subsystems per standard operation. The SM then uses the Webapp Server to verify that the packets were received and processed correctly. It does the verification by using the webapps to query the data warehouses to ensure that the test messages have correctly passed through the system. In order to communicate with the Webapp Server, the SM must also communicate with the CAS Server which provides Single Sign-on (SSO) capability for the Webapps. This ensures that the SSO capability of the system is also functional. The SM sends out alerts via email for any detected issues with data passing through the system. The SM will continue to test the system even after it discovers an issue and if it detects that the data successfully going through the system again, it will generate an email to announce the system’s recovery and the amount of time the system was down. In addition to emailing alerts and recovery messages, at the beginning of each month the SM will email a report detailing the uptime of the system’s components for the previous month.



**Figure 1- Cloud Environments for the SM**

## 3.2 Commercial Services and Software Components

As noted above, the System Monitor (SM) runs in a public cloud infrastructure. The SM also contains libraries and certificates that enable encoding and decoding per the ASN.1 UPER format and encrypting and decrypting IEEE 1609.2 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 System Architecture section of this document.

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

* EC2 – Handles processing dialogs and situation data messages
* EBS – Stores the situation data before distribution in the DSC 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

*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.

*DigitalEdge*- 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 generated by OSS Nokalva ASN.1 Studio to handle encoding and decoding the Connected Vehicle dialog and situation data messages in BER 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 SM uses in Rackspace and the SDC and SDW use in AWS. In order to use the studio, users must acquire the Connected Vehicle and Dedicated Short Range Communication ASN.1 specification files. The license for ASN.1 Studio is specifically limited for use with the Southeast Michigan Connected Vehicles.

* 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 standard. A certificate for the SM will need to be obtained from the Security Certificate Management System (SCMS) to decrypt any ASN.1 UPER messages coming into the SDC and SDW.

## 3.3 Developed SM Components for Integration into a CV Environment

In order to best observe a CV environment a standalone server was developed to perform the tasks necessary to meet the requirements of the System Monitor. Below is a description of the various software components developed to ensure integration of a SM into a real-world CV Test Bed environment.

*Standalone Application – This is a set of test applications developed for USDOT to monitor the CVCS subsystems.*

* System Monitor Java command-line application
  + Sends test data to the CVCS Subsystem components
  + Polls for test data from the CVCS Subsystems via web applications
  + Alerts on issues detected with the CVCS Subsystem components
  + Sends recovery message when an issue is detected to be resolved
  + Reports monthly on uptime of CVCS Subsystem components

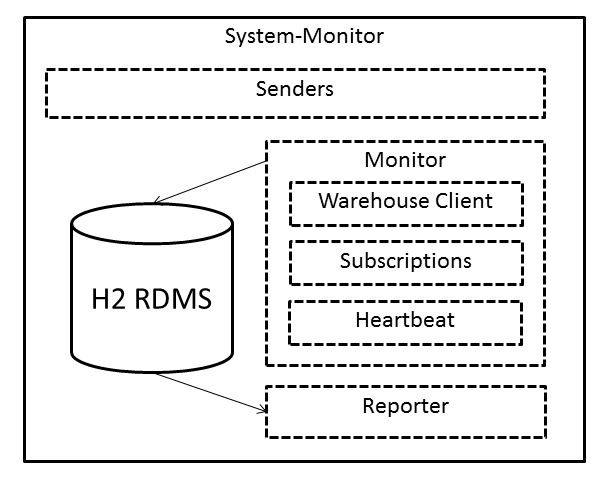
## 3.4 System Architecture

In order to meet the requirements of the SM, the system runs in the Rackspace cloud environment in order to monitor the abilities of the Forwarders to receive IPv6 traffic. The system communicates directly with both the forwarders in Rackspace and the Webapp/CAS servers in AWS. This allows the monitor to ensure all components of the system are properly receiving and passing data to the proper destinations.

### 3.4.1 Rackspace

One of the requirements for the CV systems 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. Rackspace runs the SM and custom developed forwarding 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.

### 3.4.2 System Monitor (SM) System



**Figure 2- System Monitor System**

The System Monitor, shown in Figure 3, is a standalone server. Only one instance of the monitor needs to be configured and started to monitor the CV environment data distribution and support service subsystems. The SM can be broken down into 4 components, as described below.

* Senders – The senders are the component of the SM that encode and deposit test messages into the systems via the forwarders. There can be any number of senders and they can be configured in any manner to send different types of messages, with different encryption options, using different Internet Protocols. The default configuration for the senders includes sending advisory situation data (ASD), intersection data (ISD), and vehicle situation data (VSD) monitor messages. These messages are sent in both secured and unsecured formats, over both IPv4 and IPv6. If this component stops working, it will become evident when the monitor component stops detecting monitor messages coming through the system.
* Monitor – The monitor component is where the SM verifies that the system components are operational. It is the component that communicates with the Webapp server in AWS to query the data from the DDS and ensure that all the monitor messages sent by the Senders are coming through to their destination. When a change in status is detected, the monitor will generate an alert email and deliver it to a configured list of users. The monitor component itself can be broken into 3 subcomponents:
  + Warehouse Client – The warehouse client is the tool used to connect, via WebSockets, to the Webapp server and perform the querying of monitor messages.
  + Subscriptions – Subscriptions are another component of the DDS that allow for a constant stream of data instead of performing a query at a time. The subscriptions subcomponent ensures that the DDS is capable of handling subscriptions properly.
  + Heartbeat – The heartbeat subcomponent provides a pulse on the monitor component of the SM. It will send emails at a specified rate to a list of configured users to ensure that the monitoring component is functional.
* Reporter – The reporter is the component which generates and formats the reports. Reports are distributed to a configurable list of users on the first day of every month.
* H2 RDBMS – The SM runs a H2 relational database management system to store the status of each of the components over time. When the monitor component detects a change in status of a component, it will write an entry to the H2 DB recording the time of status change. In order to prevent a stale history due to monitor subcomponent failure, a new record reporting the last known status of a component will be written to the DB at a configurable interval of time. When the reporter component generates a report, it extracts, processes, and formats each component’s status for the specific report period.

# 4 System Monitor

## 4.1 Description

The System Monitor (SM) will create one sender for every type of message it is configured to send. The senders will continuously send their messages to the CV Systems via the forwarders.

The monitor component will also be aware of the defined senders so that it is aware of what data to watch for. It will perform queries against the DDS and SSS, using WebSockets, at the configured rate to check for the monitor messages it is expecting from the senders. If the monitor fails to detect a message enough time times to cross the threshold of the configured tolerance, it will then send off an alert email to the specified alert email recipients. The monitor will also record in the H2 DB that the status of the message is down. Once an alert has been generated for a specific message, the monitor will continue to check for that message. If the monitor does not detect the message again within 24 hours, it will generate a new alert. This process will continue until the message is detected again. When the message is detected, the monitor will send a recovery email to the specified recovery email recipients and record in the H2 DB that the status of the message is up.

On the first day of every month, the reporter will generate a report for the previous month’s uptimes. The report is saved on the local disk of the SM server and distributed over email to a configurable list of recipients. The report is a CSV giving a summary of the uptime of each of the CVCS system’s components as well as a detailed listen of the uptime of each message type sent by the SM.

## 4.2 Deployment

### 4.2.1 Automatic Deployment

To automatically deploy the system monitor using default configuration, the ZIP file generated during the build process should be unzipped to the desired directory. On Linux systems, the install\_system\_monitor.sh script can then be run. This script will modify all the configuration files to contain the correct directory path based on the monitor’s current directory. It will also configure the server to run the System Monitor via upstart, allowing the server to automatically restart the monitor if it fails.

### 4.2.2 Manual Deployment

To manually deploy the monitor using default configuration, the configuration scripts must all be modified. The configuration files contain the special variable %PATH\_TO\_CV-SYSTEM-MONITOR-DIR%. Each file that contains this variable must be modified so that the variable is replaced with the correct, fully qualified directory path to the file the variable is referencing. Once that is completed, the monitor will be configured.

## 4.3 Custom Configuration

To customize the configuration there are multiple configuration files for the separate components of the system. All the configuration files have README files provided in the docs/ directory of the ZIP. The README files describe the configuration and each field’s purpose, default value, and valid values.

The main configuration file is the system\_monitor\_config.json file. This file is where the senders’ configurations, the monitor configuration, and the reporter configuration are defined. The values in this file will control the general behavior of the monitoring as well as tell the monitor where each component’s required configuration files is located.

Each sender must have its own configuration file. These files are the standard configuration files used by the CV Sender class and contain which messages are used, the format of the message, where to send the message, and other information required by the CV Sender class. The default configuration files are located in the config/senders/ directory. The message files that the default sender configuration files point to are located in the config/messages/ directory.

The monitor component requires the path to the warehouse\_config.json file to properly configure the warehouse client subcomponent. The warehouse\_config file is the standard configuration file used by the CV WarehouseClient class and contains information to connect to the warehouse via WebSockets. Additionally, the monitor requires the definition of subscriptions for the subscription subcomponent. The monitor configuration is also where the heartbeat is defined. Lastly, the configuration requires a list of email recipients for both alert and recovery emails.

The reporter component is called the “uptimeConfig” in the system\_monitor\_config file. It is where the configuration for the H2 DB is set up as well as where reports are stored and a list of recipient’s emails which the report will be delivered to.

## 4.4 Monitor Message Format

There are a few key settings that make a message sent by the senders a valid monitor message.

* The monitor message must be an AdvisorySituationData (ASD) (Dialog ID = 156), IntersectionSituationData (ISD) (Dialog ID = 162), or VehSitDataMessage (VSD) (Dialog ID = 154) message.
* The monitor message must have the correct Group ID to indicate it as a SM Monitor Message. There are 4 valid Group IDs to that can be used. They are defined in Appendix B.
* The position in the monitor message must be defined so that it falls within the bounding box of
  + Northwest Latitude = 28.049
  + Northwest Longitude = -81.653
  + Southeast Latitude = 28.047
  + Southeast Longitude = -81.651.

This is the hard coded, specific region in which the monitor will query for records as it is a water location where it would not be possible for vehicles to drive.

Other than these constraints, any other fields within the data message can be set to any standard, valid value that would allow the message to successfully be encoded and pass through CV systems.

## 4.4 Starting the Monitor

The monitor is started using the scripts provided in the initial ZIP file

If the monitor was deployed using the method described in Section 4.2.1 Automatic Deployment, then the monitor should be automatically started. Additionally, using the automatic deployment method will configure the server to create an upstart job so that if the monitor dies or is stopped, it will restart automatically.

If the monitor is deployed as described in the Section 4.2.2 Manual Deployment, then the monitor must be started by hand. To start the monitor, run the provided start\_system\_monitor.sh script. This script requires a single parameter: the path to the system\_monitor\_config.json file.

# 5 Updating OSS Licensing

1. Install new OSS liscense (“ossinfo” file) into the base folder where the OSS executable is located
2. Restart OSS Studio
3. Compile the ASN to Java using OSS Studio
4. Delete the old compiled code from the fedgov-cv-asn1-generated-j2735 Git repository: src/main/java/gov/usdot/asn1/generated/j2735
5. Copy the new compiled code to the same location in the Git repository
6. Build the fedgov-cv-asn1-generated-j2735 project
7. Build the rest of the projects that have dependencies on the generated code in the following order:
   1. fedgov-cv-asn1
   2. fedgov-cv-common
   3. fedgov-cv-apps
   4. fedgov-cv-system-moitor
8. Deploy the system monitor to the VM and restart:

< SCP fedgov-cv-system-monitor-1.0.0-SNAPSHOT-bin.zip to /cv-system-monitor >

< Save a copy of the existing system monitor configuration files if you want to maintain the current

configuration >

cd /cv-system-monitor

unzip fedgov-cv-system-monitor-1.0.0-SNAPSHOT-bin.zip

< Modify the unzipped configuration files or replace them with the saved copies of the old

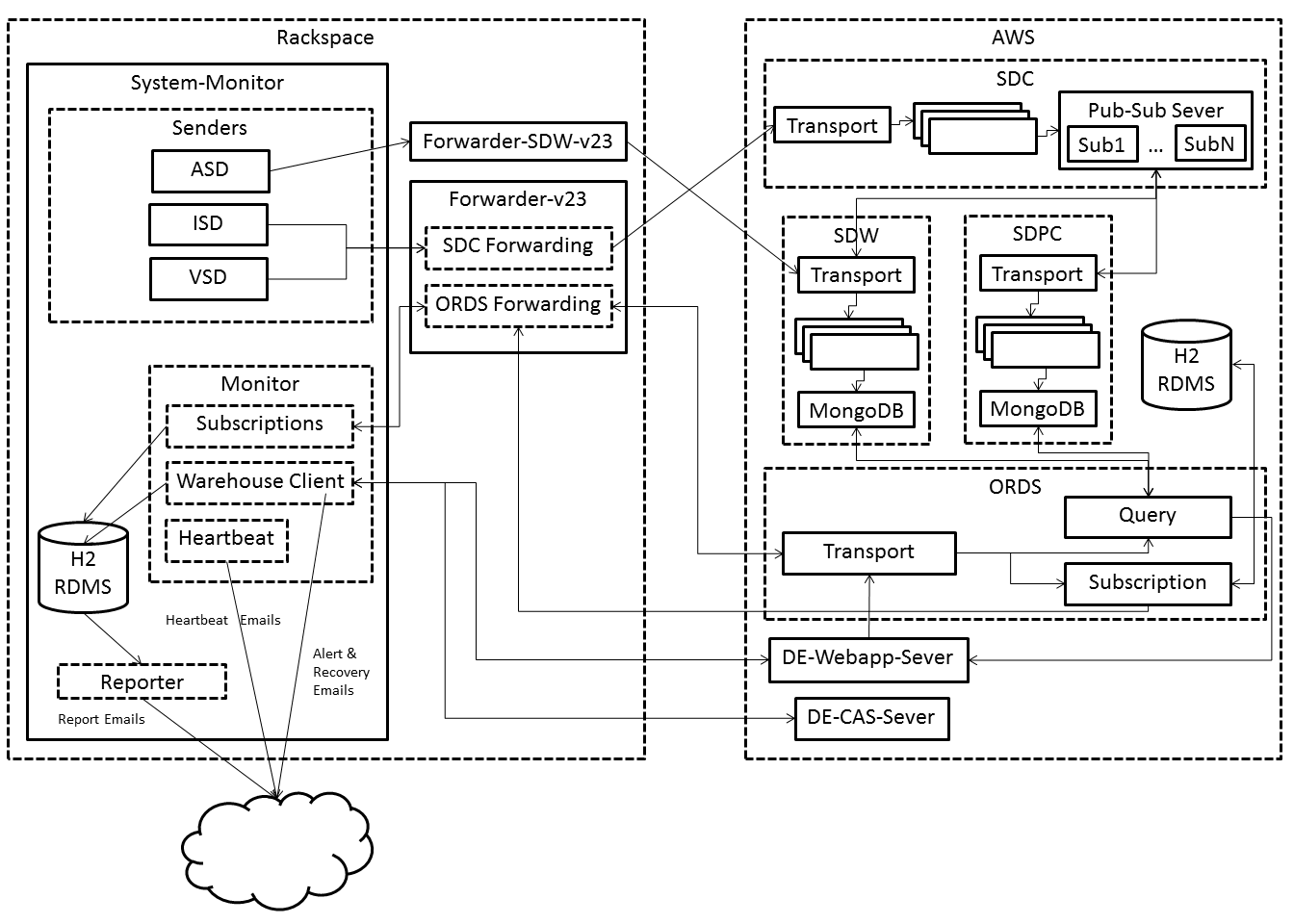
configuration >

ps aux | grep system\_monitor

sudo kill -9 <system\_monitor PID>

nohup ./start\_system\_monitor.sh config/system\_monitor\_config.json 2>&1 &

# Appendix A: Message Flow Diagram



**Figure 3- Message Flow Diagram**

# Appendix B: Monitor Messages Group ID Definitions

**Table 1. Monitor Messages Group ID Definitions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Group ID (Hex)** | **Group ID (Decimal)** | **IP Type** | **Secure** |
| 0x10B01000 | 279973888 | IPv4 | True |
| 0x10B02000 | 279977984 | IPv4 | False |
| 0x10B03000 | 279982080 | IPv6 | True |
| 0x10B04000 | 279986176 | IPv6 | False |

# Appendix C: Instances Accessed By SM

**Table 2. Instances Accessed by SM**

|  |  |  |
| --- | --- | --- |
| **Instance** | **IP Type** | **IP Address** |
| System-Monitor |  |  |
|  | IPv4 | 104.130.2.93 |
| Forwarder-v23 |  |  |
|  | IPv4 | 104.130.170.234 |
|  | URLs-IPv4 | sdc2.connectedvcs.com  ords2.connectedvcs.com |
|  | IPv6 | 2001:4802:7803:104:be76:4eff:fe20:bfb2 |
|  | URLs-IPv6 | sdc6-2.connectedvcs.com  ords6-2.connectedvcs.com |
| Forwarder-SDW-v23 |  |  |
|  | IPv4 | 104.130.28.59 |
|  | URL-IPv4 | sdw2.connectedvcs.com |
|  | IPv6 | 2001:4802:7803:104:be76:4eff:fe20:afe7 |
|  | URL-IPv6 | sdw6-2.connectedvcs.com |
| DE-SDC-Transport |  |  |
|  | IPv4 | 54.80.143.114 |
| DE-SDW-Transport |  |  |
|  | IPv4 | 54.198.152.161 |
| DE-Webapp-Server |  |  |
|  | URL | webapp2.connectedvcs.com |
|  | IPv4 | 54.198.59.102 |
| DE-CAS-Server |  |  |
|  | URL | cas.connectedvcs.com |
|  | IPv4 | 23.21.230.64 |
| DE-SDW-MongoDB |  |  |
|  | IPv4 | 54.196.111.250 |
| DE-SDPC-MongoDB |  |  |
|  | IPv4 | 54.162.80.160 |

# Appendix D: Test Path Coverage

## D.1 Sending Messages

**Table 3. Test Path Coverage for Sending Messages**

|  |  |  |  |
| --- | --- | --- | --- |
| **IP** | **Secure** | **Message** | **IP Test Path Coverage** |
| IPv4 | False | ASD | SM → Forwarder-SDW-v23 (URL-IPv4) → DE-SDW-Transport (IPv4) |
| IPv4 | True | ASD | SM → Forwarder-SDW-v23 (URL-IPv4) → DE-SDW-Transport (IPv4) |
| IPv6 | False | ASD | SM → Forwarder-SDW-v23 (URL-IPv6) → DE-SDW-Transport (IPv4) |
| IPv6 | True | ASD | SM → Forwarder-SDW-v23 (URL-IPv6) → DE-SDW-Transport (IPv4) |
| IPv4 | False | ISD | SM → Forwarder-v23 (URL-IPv4) → DE-SDC-Transport (IPv4) |
| IPv4 | True | ISD | SM → Forwarder-v23 (URL-IPv4) → DE-SDC-Transport (IPv4) |
| IPv6 | False | ISD | SM → Forwarder-v23 (URL-IPv6) → DE-SDC-Transport (IPv4) |
| IPv6 | True | ISD | SM → Forwarder-v23 (URL-IPv6) → DE-SDC-Transport (IPv4) |
| IPv4 | False | VSD | SM → Forwarder-v23 (URL-IPv4) → DE-SDC-Transport (IPv4) |
| IPv4 | True | VSD | SM → Forwarder-v23 (URL-IPv4) → DE-SDC-Transport (IPv4) |
| IPv6 | False | VSD | SM → Forwarder-v23 (URL-IPv6) → DE-SDC-Transport (IPv4) |
| IPv6 | True | VSD | SM → Forwarder-v23 (URL-IPv6) → DE-SDC-Transport (IPv4) |

## D.2 Subscriptions

**Table 4. Test Path Coverage for Subscriptions**

|  |  |  |
| --- | --- | --- |
| **IP** | **Secure** | **IP Test Path Coverage** |
| IPv4 | False | SM → Forwarder-v23 (URL-IPv4) → DE-ORDS-Transport (IPv4) |
| IPv4 | True | SM → Forwarder-v23 (URL-IPv4) → DE-ORDS-Transport (IPv4) |
| IPv6 | False | SM → Forwarder-v23 (URL-IPv6) → DE-ORDS-Transport (IPv4) |
| IPv6 | True | SM → Forwarder-v23 (URL-IPv6) → DE-ORDS-Transport (IPv4) |

## D.3 Querying Databases via WebSocket

**Table 5. Test Path Coverage for Querying Databases via WebSocket**

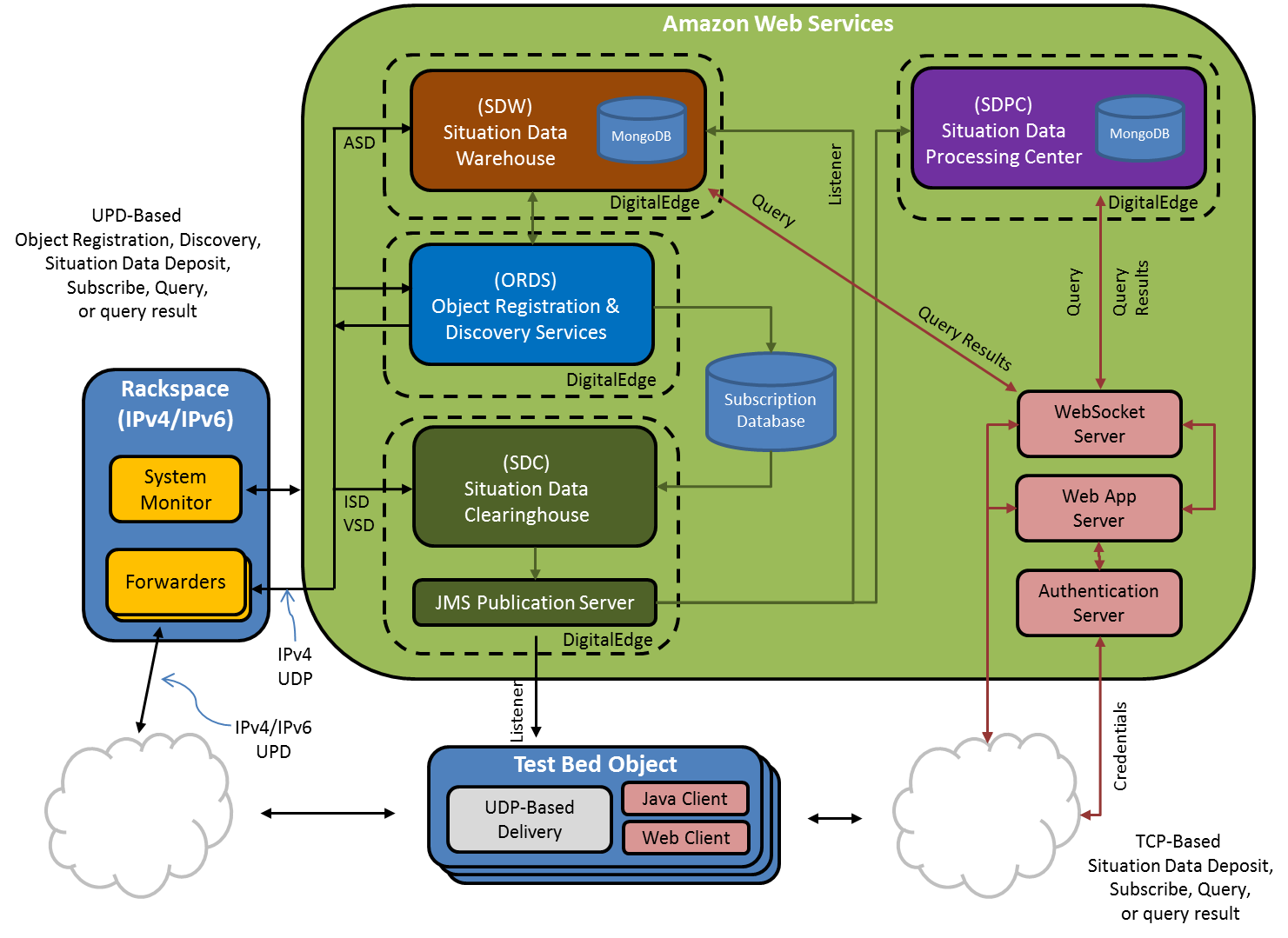
|  |  |
| --- | --- |
| **DE Component** | **Test Path Coverage** |
| DE-CAS-Server | SM → DE-CAS-Server (URL) |
| DE-SDW-MongoDB | SM → DE-Webapp-Server (URL) → DE-SDW-MongoDB (IPv4) |
| DE-SDPC-MongoDB | SM → DE-Webapp-Server (URL) → DE-SDPC-MongoDB (IPv4) |

# Appendix E: 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 G: 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.



**Figure 4- System Architecture**