

Southeast Michigan Test Bed Advanced Data Capture Field Testing

Task 4: System Design Document

www.its.dot.gov/index.htm

Draft Version 3.1 — July 28, 2016

FHWA-JPO-15-227



U.S. Department of Transportation

Produced by Booz Allen Hamilton
U.S. Department of Transportation
Federal Highway Administration (FHWA)

Notice

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

The U.S. Government is not endorsing any manufacturers, products, or services cited herein and any trade name that may appear in the work has been included only because it is essential to the contents of the work.

Quality Assurance Statement

The Federal Highway Administration (FHWA) provides high-quality information to serve Government, industry, and the public in a manner that promotes public understanding. Standards and policies are used to ensure and maximize the quality, objectivity, utility, and integrity of its information. FHWA periodically reviews quality issues and adjusts its programs and processes to ensure continuous quality improvement.

Technical Report Documentation Page

| | | | | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|------------------------------------------------------|--|-----------------------------------------------|--|
| 1. Report No. FHWA-JPO-15-227 | | 2. Government Accession No. | | 3. Recipient's Catalog No. | |
| 4. Title and Subtitle Southeast Michigan Test Bed Advanced Data Capture Field Testing Task 4: System Design Document | | | | 5. Report Date July 28, 2016 | |
| | | | | 6. Performing Organization Code | |
| 7. Author(s) Hamid Musavi , Paul Valdetaro, Mathew Kowalsky | | | | 8. Performing Organization Report No. | |
| 9. Performing Organization Name And Address Booz Allen Hamilton 8283 Greensboro Drive McLean, VA 22102 | | | | 10. Work Unit No. (TRAIS) | |
| | | | | 11. Contract or Grant No. | |
| 12. Sponsoring Agency Name and Address U.S. Department of Transportation Federal Highway Administration 1200 New Jersey Avenue, SE Washington, DC 20590 | | | | 13. Type of Report and Period Covered | |
| | | | | 14. Sponsoring Agency Code FHWA | |
| 15. Supplementary Notes | | | | | |
| 16. Abstract This document presents the architectural and high level design of the Operational Data Environment (ODE) for the Southeast Michigan (SEMI) Test Bed. The SEMI-ODE is a data provisioning service that is configurable by its users via a web application component. It operates between one or more CV (or other ITS) related data sources and any category/type of client application that requires one of these types of data to function. The core capability of the <i>initial release</i> of the SEMI-ODE to ingest data is enabled by the SEMI-ODE's ability to actively manage subscriptions to the US DOT Situation Data Clearinghouse (SDC), the US DOT Situation Data Warehouse (SDW) and the USDOT Situation Data Processing Center (SDPC). SEMI ODE also performs necessary security / credential checks and, as needed, data valuation, aggregation, integration, sanitization and propagation functions. | | | | | |
| 17. Key Words ODE, Operational Data Environment, aggregation, integration, valuation, propagation, v2i, connected vehicles, architectures, big data, real-time processing, API, ASN.1, CV, DDS, DSRC, EVSD, Hadoop, ITS, Yarn, HDFS, ISD, JDK, Kafka, Liferay, Portal, j2735, SDC, SDPC, SDW, Spark, SPaT, SSL, TSD, VM, HTTP WebSocket | | | | 18. Distribution Statement No Restrictions | |
| 19. Security Classif. (of this report) Unclassified | | 20. Security Classif. (of this page) Unclassified | | 21. No. of Pages 100 | |
| | | | | 22. Price N/A | |

This page intentionally left blank

ACCEPTANCE / APPROVAL PAGE

// // _____
Name
Author's Title

Prepared _____
by Date

// // _____
Name
Quality Assurance

Reviewed _____
by Date

// // _____
Name
Project Manager

Approved _____
by Date

DOCUMENT CHANGE HISTORY

| Version Number | Date | Description |
|----------------|------------|------------------------------------------------------|
| 0.95 | 03/13/2015 | Annotated outline release |
| 0.98 | 04/15/2015 | Draft release |
| 1.1 | 8/18/2015 | Release to reflect US DOT and Noblis review comments |
| 2.0 | 9/15/2015 | Revised with additional content |
| 3.0 | 3/31/2016 | Updated with latest implementation changes |
| 3.1 | 7/28/2016 | Release to reflect US DOT and Noblis review comments |

Table of Contents

| | |
|--------------------------------------------------|-----------|
| Chapter 1. Introduction | 1 |
| 1.1. Project Background..... | 1 |
| 1.2. System Overview | 2 |
| 1.3. Document Overview | 3 |
| 1.4. Reference documents | 4 |
| Chapter 2. System Architecture | 6 |
| 2.1. System Design Considerations | 8 |
| 2.2. Architectural Design | 10 |
| 2.3. Interfaces | 13 |
| 2.3.1. DDS Interfaces..... | 13 |
| 2.3.2. ODE Interfaces..... | 16 |
| 2.4. ODE Database Design | 23 |
| 2.5. System Security | 24 |
| 2.5.1. Network Level Security | 24 |
| 2.5.2. Database Level Security | 25 |
| 2.5.3. Application Level Security | 25 |
| 2.6. ODE Process Flow | 25 |
| Chapter 3. Components..... | 29 |
| 3.1. Request Processing Components | 30 |
| 3.1.1. Service API | 32 |
| 3.1.2. Identity and Access Management (IAM)..... | 33 |
| 3.1.3. Request Manager | 34 |
| 3.1.4. Connector | 36 |
| 3.2. Data Processing Components | 37 |
| 3.2.1. Overview | 37 |
| 3.2.2. Collector..... | 40 |
| 3.2.3. Integrator | 41 |
| 3.2.4. Validator (Data Valuation) | 42 |

| | | |
|------------------------------------------------------------------------------------|-------------------------------------------------------------------|-----------|
| 3.2.5. | Sanitizer | 44 |
| 3.2.6. | Aggregator | 46 |
| 3.2.7. | Distributor | 49 |
| 3.3. | Data Consumption | 49 |
| 3.4. | Logging and Monitoring | 50 |
| 3.5. | Space Estimates | 51 |
| 3.6. | Impact to Existing Components..... | 51 |
| Chapter 4. Requirements Traceability | | 53 |
| Chapter 5. Database..... | | 65 |
| 5.1. | Overview | 65 |
| 5.2. | Entity Relationship Diagram | 65 |
| 5.3. | Data Dictionary | 65 |
| Chapter 6. Interface Specifications | | 67 |
| 6.1. | Outbound Requests placed by SEMI ODE to DDS..... | 67 |
| 6.1.1. | Enhanced Vehicle Situation Data (EVSD) Subscription Request | 67 |
| 6.1.2. | Intersection Situation Data (ISD) Subscription Request | 68 |
| 6.1.3. | Traveler Situation Data (TSD) Query | 68 |
| 6.2. | Inbound Data Streams from the DDS to the ODE..... | 68 |
| 6.2.1. | Enhanced Vehicle Situation Data (EVSD) Messages..... | 69 |
| 6.2.2. | Intersection Situational Data (ISD) Messages | 72 |
| 6.2.3. | Traveler Situation Data (TSD) Messages | 74 |
| 6.2.4. | 3 rd party/ancillary data providers | 77 |
| 6.3. | Outbound Data Streams from SEMI ODE to Client Applications | 77 |
| 6.3.1. | Vehicle Data..... | 78 |
| 6.3.2. | Intersection Data | 78 |
| 6.3.3. | SPaT Data | 80 |
| 6.3.4. | Map Data..... | 81 |
| 6.3.5. | Advisory Data | 83 |
| 6.3.6. | Aggregate Data | 83 |
| 6.3.7. | User | 85 |
| APPENDIX A: Connected Vehicle Reference Implementation Architecture (CVRIA) | | |
| Physical Object Definitions (for Reference ONLY)..... | | 87 |
| 6.4. | Archived Data Center | 87 |

| | | |
|-------------------------------------------------------------------------------------------------------|-----------------------------------------------------|-----------|
| 6.5. | Center | 87 |
| 6.6. | Cooperative ITS Credentials Management System | 88 |
| 6.7. | Data Distribution System..... | 88 |
| 6.8. | ITS Roadway Equipment..... | 88 |
| 6.9. | Object Registration and Discovery Service | 88 |
| 6.10. | Personal Information Device | 88 |
| 6.11. | Roadside Equipment | 89 |
| 6.12. | Service Monitor System..... | 89 |
| 6.13. | Transportation Information Center | 89 |
| 6.14. | Vehicle On-Board Equipment (OBE)..... | 90 |
| APPENDIX B: CVRIA System Architecture Application Object Definitions (for Reference ONLY)..... | | 93 |
| 6.15. | Operational Data Environment | 93 |
| 6.16. | Archived Data Center | 93 |
| 6.17. | Archived Data User Systems | 94 |
| 6.18. | Authorizing Center | 94 |
| 6.19. | Center | 94 |
| 6.20. | Transportation Information Center | 94 |

List of Figures

| | |
|-----------------------------------------------------------------------|----|
| Figure 1 - System Architecture Physical View – Level 0..... | 6 |
| Figure 2 - System Architecture Physical View – Level 1..... | 11 |
| Figure 3 - SEMI ODE Technology Stack | 12 |
| Figure 4 - DDS Account Request Form..... | 16 |
| Figure 5 - SEMI ODE Process Flow Diagram..... | 26 |
| Figure 6 - SEMI ODE Application Components..... | 29 |
| Figure 7 - Request Processing Sequence of Operations | 31 |
| Figure 8 - Request Manager Subscription Consolidation Algorithm | 35 |
| Figure 9 - Data Processing Sequence of Operations..... | 38 |

List of Tables

| | |
|--------------------------------------------------------------------------------|----|
| Table 1 – DDS Interface Selection Decision Matrix | 15 |
| Table 2 – SEMI ODE REST API | 18 |
| Table 3 – ODE Streaming API | 20 |
| Table 5 - Vehicle Data Validation Ranges | 44 |
| Table 6 - Requirements Traceability Matrix..... | 53 |
| Table 7 – Enhanced Vehicle Situation DataEnhanced Vehicle Situation Data | 69 |
| Table 7 – Intersection Situation Dara | 72 |
| Table 9 – Traveler Situation Data | 77 |

CHAPTER 1. INTRODUCTION

This System Design Document (SDD) details the software architecture and related design decisions needed to implement the Operational Data Environment (ODE) component as part of the Connected Vehicle (CV) Program.

The intended audience for this SDD is an individual possessing a working knowledge of Information Technology (IT). This SDD document is a high-level design and implementation description and does not specify the low level details of the design nor how the design is/will be specifically implemented.

As described in the Performance Work Statement (PWS), this project leverages Agile project management / software development techniques, which provide for an iterative approach to development and release management. This approach permits the development effort to be broken into discrete and relatively small “chunks” of development, which are referred to as “Sprints”. The system at the start of the Agile development cycle is deliberately understood at different levels of details, with deeper knowledge of specific aspects of the system gained as the items packaged for a specific sprint are identified and studied. This implies that this SDD document is treated as a “living document”, with progressive updating of the relevant document sections as necessary during each sprint. This approach allows design decisions to be postponed until the last responsible moment in time when the decision needs to be made. This results in better decisions since the factors impacting the decision are more fully understood as the development effort progresses. A side effect is that the documentation is also more accurate whenever read. This can be contrasted to the classic waterfall methodology, which requires a fully developed document at the outset. Systems developed using the waterfall approach, in contrast, are rarely implemented as initially documented and the accuracy of the document is commensurately lower at any given point in time during the development cycle.

1.1. Project Background

The promise of Connected Vehicle technology has been the focus of many research efforts, both public and private in nature. These efforts span various types of institutions including academia, government, and private industry – many of which belong to the automotive and communication sectors. Given the complex nature of the infrastructure that will eventually be needed to support this technology, a series of connected vehicle test beds has been created. These test beds allow researchers to further advance the state of the art and the state of the practice for connected vehicle technology. Research and development performed using the test beds allows for the identification and distribution of lessons learned to support the continued development of this technology. To enable the use of connected vehicle applications, it has become clear through such research efforts that having a mechanism to ingest, process, and output pertinent data from

connected vehicles and connected infrastructure to client (transportation) applications is imperative. It is with this concept in mind that the SEMI ODE is being developed.

To date, extensive work is on-going to develop a connected vehicle test bed in Southeast Michigan. Emphasis has been placed on the secure and trusted communication of broadcast safety messages and peer-to-peer encrypted information exchanges. Hyper-current, and hyperlocal messages, which are high frequency messages transmitted at ~10Hz, are broadcast to other vehicles for real-time use by safety applications. Basic Safety Messages (BSMs) are captured as Enhanced Vehicle Situation Data (EVSD) and Signal Phase and Timing (SPaT) messages are captured as Intersection Situation Data (ISD). Both EVSD and ISD are sent to the US DOT Situation Data Clearinghouse (SDC) for real-time distribution. Travel advisory messages are captured as Traveler Situation Data (TSD) and are available from the US DOT Situation Data Warehouse (SDW), for a configurable time period. In addition to storing TSD, the SDW also stores EVSD and ISD. The data clearinghouse and data warehouse (where data from the test bed are stored) do not perform any quality checking or data bundling operations. Connected vehicle applications or data aggregators may subscribe to the real-time data clearinghouse; or query from data warehouse to obtain data on the basis of time and/or location.

Given the opportunity afforded by work being done in the Southeast Michigan Test Bed, the SEMI ODE that is presented here is not only aimed at supporting the general development of connected vehicle technology but more specifically, providing an initial implementation of quality checking, aggregating and provisioning of both real-time and archived data from the test bed (and other sources) to various client applications.

1.2. System Overview

The SEMI ODE is intended to complement a connected vehicle infrastructure by functioning as a smart data router by brokering processed data from various data sources, including connected vehicles, to a variety of data users. Data users include transportation software applications, such as those that may be used by a Transportation Management Center (TMC).

As a data provisioning service, the SEMI ODE provisions data from disparate data sources to software applications that have placed data subscription requests to the ODE. These subscribing applications may include CV applications as well as non-CV applications. CV applications may include applications that:

- Warn drivers that are approaching temporary work zones at unsafe speeds and/or trajectory.
- Warn public safety personnel and other officials working in the zone through an audible warning system about errand vehicles.
- Dynamically adjust and coordinate maximum appropriate vehicle speeds in response to downstream congestion, incidents, and weather or road conditions in order to maximize traffic throughput and reduce crashes.

- To provide a vehicle operator with sufficient warning of an impending queue backup in order to brake safely, change lanes, or modify the route such that secondary collisions can be minimized or even eliminated.
- Perform calculations to determine the vehicle's optimal speed to pass the next traffic signal on a green light or to decelerate to a stop in the most eco-friendly manner.

Non-CV applications may include applications that:

- Monitor signal operations to see if a traffic signal is operating properly
- Monitor road weather conditions to alert drivers of weather-related road hazards.

While provisioning data from data sources for data users, the SEMI ODE also performs necessary security / credential checks and, as needed, data valuation, aggregation, integration, sanitization and propagation functions. These functions are core functions to the SEMI ODE and are detailed in later sections. However, these may be summarized, in no particular order, as the following:

- data valuation is the process of making a judgment about the quality or value of the data
- data integration is the process of combining different data from multiple sources to provide more complete information
- data sanitization is the modification of data as originally received in order to reduce or eliminate the possibility that the data can be used to compromise the privacy of the individual(s) that might be linked to the data
- data aggregation is the creation of composite or summary information from more granular data

1.3. Document Overview

This System Design Document defines the strategies necessary to accomplish the design activities associated with Connected Vehicle ODE.

The remaining System Design Document sections are organized as follows:

- **Section 2. System Architecture:** Describes the system, the conceptual and architectural design, interfaces, and the various environments that are used throughout the development lifecycle.
- **Section 3. Components:** Identifies and describes each major component of the ODE.
- **Section 4. Requirements Traceability:** This section includes two traceability matrices that trace from Design to Requirements and vice-versa.
- **Section 5. Database:** Describes database-wide design decisions about the databases' behavioral design and other decisions affecting the further design of the databases broken down into its parts that are described in detail.

- **Section 6. Interface Specifications:** This section lists the API specifications used by the SEMI ODE to communicate to the Clearinghouse / Data warehouse, for return messages as well as for the outbound messages.
- **Section 7. Connected Vehicle Reference Implementation Architecture (CVRIA) Physical Object Definitions**
- **Section 8 System Architecture Application Object Definitions.**

1.4. Reference documents

Documents used as the basis for creation of the System Design Document include:

- PWS for Southeast Michigan Testbed Advanced Data Capture Field Testing
- Southeast Michigan Testbed Advanced Data Capture Field Testing Task 3 Whitepaper v1.1
- Southeast Michigan Testbed Advanced Data Capture Field Testing 4 Concept of Operations v1.0
- Southeast Michigan Testbed Advanced Data Capture Field Testing Task 4 Field Test Plan v1.0
- Southeast Michigan Testbed Advanced Data Capture Field Testing Task 5 Technical Report v1.0
- Southeast Michigan Testbed Advanced Data Capture Field Testing Task 6 Technical Report v1.0

This page intentionally left blank

CHAPTER 2. SYSTEM ARCHITECTURE

The SEMI ODE is a data forwarding service that is configurable by its users via a web application component. It operates between one or more CV (or other ITS) related data sources and any category/type of client application that requires one of these types of data to function. Figure 1 below depicts the SEMI ODE within the larger system of the Connected Vehicle Reference Implementation Architecture (CVRIA) and its potential data sources and consumers. Each element in this diagram depicts a computational component that can exist as a single purpose physical device or as a virtualized device existing within the shared computational capabilities of a larger virtualized computing environment.

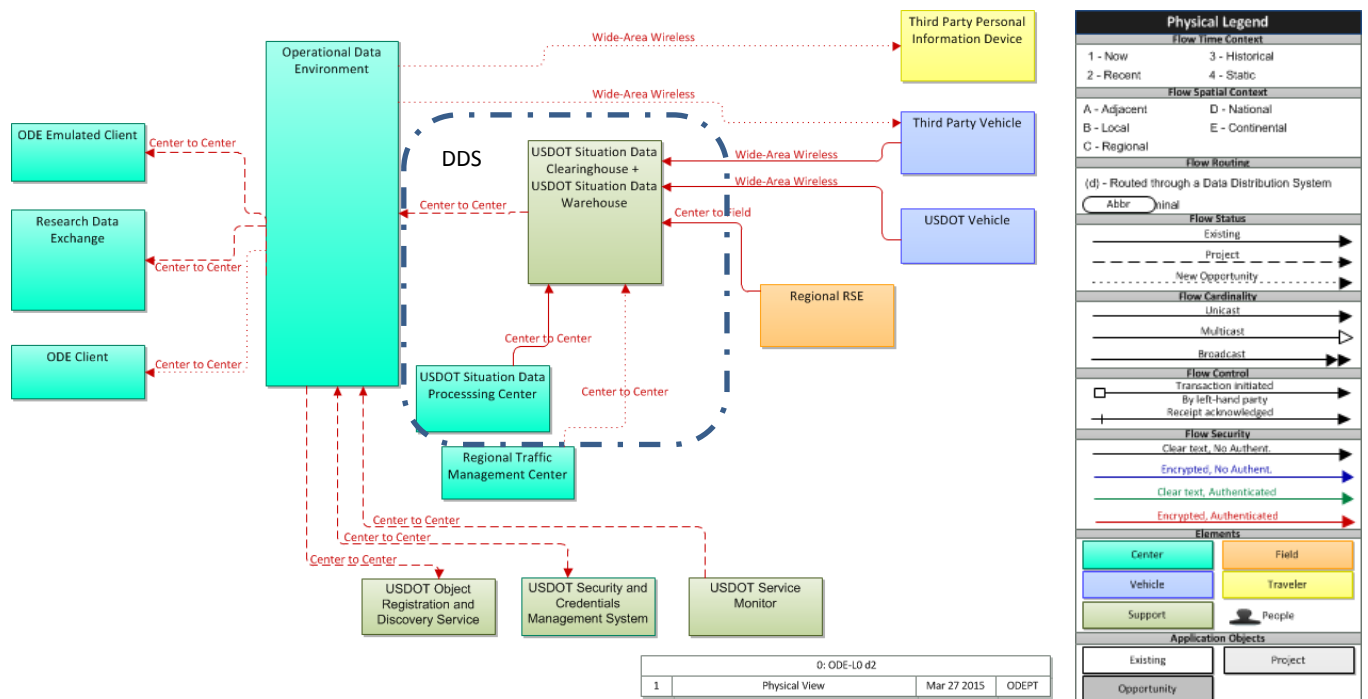


Figure 1 - System Architecture Physical View – Level 0

Figure 1 presents a number of key components of the system, each of which are described in subsequent sections. The core capability of the SEMI ODE to ingest data is enabled by the ODE's ability to actively manage subscriptions to the SDC, the SDW and the USDOT Situation Data Processing Center, collectively referred to herein as the USDOT Data Distribution System (DDS) based on Connected Vehicle Reference Implementation Architecture (CVRIA) nomenclature. (Appendix A contains the detailed descriptions of the physical objects shown above in Figure 1.)

As mentioned above, the US DOT DDS consists of three repositories. These are:

1. US DOT Situation Data Clearinghouse (SDC)

The "USDOT Situation Data Clearinghouse" collects, processes, and distributes connected vehicle data of short term utility; that is, data that are useful 'now', and typically over a small area. It consumes data from data producers, such as connected vehicles, and provides that data to interested data consumers using a publish-and-subscribe mechanism. It does not store data longer than is necessary to satisfy the publication mechanism.

2. US DOT Situation Data Warehouse (SDW)

The "USDOT Situation Data Warehouse" is a Data Distribution System that collects, processes, and distributes connected vehicle data, connecting data producers with data consumers and facilitating data exchange in the Connected Vehicle Environment. It focuses on data that is relevant for a period beyond the immediate, and distributes that data to interested parties using a query mechanism. The SDW data retention window is designed to allow data consumers to query for relevant data whenever the supporting use case does not justify a constant stream of data as is provided by the SDC. It may discard data when it is no longer relevant.

3. US DOT Situation Data Processing Center (SDPC)

The "USDOT Situation Data Processing Center" is a Data Distribution System that collects, processes, and distributes connected vehicle data, connecting data producers with data consumers and facilitating data exchange in the Connected Vehicle Environment. It is a subscriber to the SDC and retains data for a longer period of time than does the SDW. The retention period for SDPC data is 6 months. The SEMI ODE interfaces with the SDPC via the same interface mechanism as the SDC and SDW but with a different data source identifier.

In addition to these CV data sources, the SEMI ODE is extensible in the sense that it is designed to ingest data from other sources as these are identified in the future. These other data sources may include repositories that contain data such as weather, special event calendar(s), and transportation management strategies from third parties.

The SEMI ODE supports the following classes of applications:

- **ODE Emulated Clients:** These clients are developed to emulate the operation of potential full scale applications for the purpose of validating the concept and function of the ODE. These applications may impose performance and quality requirements on the SEMI ODE to enable the application to operate as expected.
- **Research Data Exchange (RDE):** The Research Data Exchange is developed as a data sharing system that is focused on both archived and real-time data from multiple transportation related sources (including vehicle probes) and multiple modes. The primary purpose of the Connected Data Systems (CDS) Research Data Exchange is to

provide a variety of data-related services that support the development, testing, and demonstration of multi-modal transportation mobility applications being pursued under the USDOT ITS Dynamic Mobility Applications (DMA) Program and other connected vehicle research activities.

The SEMI ODE will serve as a source of data for a new data environment in the RDE. The PWS states the RDE will be enhanced by its own development team in order to capture SEMI ODE provided data. The RDE will interface with the SEMI ODE and archive data provided by the SEMI ODE for use by the transportation research community.

- **ODE Client:** Analogous to the SEMI ODE Emulated Client, an SEMI ODE client is a real-life application that interfaces with the SEMI ODE and whose operational capability depends on the data provided by the ODE.
- **Third Party Personal Information Device:** Apps installed on third party personal information devices such as smart-phones could interface with SEMI ODE via wireless internet and receive data for the purpose of providing mobility, environmental and other ITS applications.
- **Third Party Vehicles:** Similar to Third Party Personal Information Devices, third party vehicles On-Board Equipment could interface with SEMI ODE via wireless internet and receive data for the purpose of providing mobility, environmental and other ITS applications.

2.1. System Design Considerations

The SEMI ODE is designed as a service-based software component that operates in the background without a visual user interface. It functions within the realm of Machine-to-Machine (M2M) processing through a set of defined programming interfaces which are directly invoked by remote applications requiring its data provisioning services via published API's. When contrasted to a traditional On-Line Transaction Processing (OLTP) system, such as an order entry system or Human Resources system, M2M processing is typically characterized by significantly higher data volumes. This higher volume can be attributed to the fact that M2M environments process data created by sensors or other automated methods rather than by traditional human input. In the case of the ODE, millions of Connected Vehicles will eventually be generating data every day, with industry estimates suggesting that each vehicle may produce an average of 1.3 gigabytes (GB) of data every month¹.

Given these anticipated data volumes, the SEMI ODE will require significant scalability in order to meet its anticipated processing demands. The term *scalability* is used here to describe the ability of a system to accommodate increased computational demands without the need to

¹ (http://www.cisco.com/web/about/ac79/docs/mfg/Connected-Vehicles_Exec_Summary.pdf).

require significant adjustment or customization to existing software. Given this operational definition, the first key decision in the design of the SEMI ODE has been to determine the most appropriate hardware architecture on which to deploy the SEMI ODE to ensure maximum scalability. This is reviewed by looking at the two approaches currently capable of achieving scale: vertical and horizontal scaling.

Vertical scaling is the traditional approach to scalability which is implemented within a “single box” solution. It entails the upgrade of hardware, primarily RAM, CPU, Disk and/or software to achieve improved performance. Implicit in this approach is operational downtime to achieve the upgrade as well as the fact that this approach has limited benefits. The socket on the motherboard in which the CPU connects, for example, is designed to accept CPU’s based on a general class, or family, of CPU’s. It becomes impossible, therefore, to upgrade to newer even more powerful CPU’s that employ a different socket configuration. Another practical aspect is that as each component is upgraded, the other “legacy” components connecting to the upgraded component typically becomes the weakest link in the processing chain, thereby limiting the potential gains of any single upgrade.

Horizontal scaling, also known as scaling out, is the more modern approach to achieving scalability. It leverages inexpensive commodity servers to create a computing cluster which is able to “divide and conquer” the computational tasks at hand through cooperative processing. Unlike vertical scaling, horizontal scaling is practically unlimited as additional servers can be easily and inexpensively added to the cluster while achieving near linear gains in performance/throughput. The cluster architecture also permits horizontally scaled solutions to achieve High Availability (HA) and fault tolerance.

Horizontal scaling requires the use of Big Data software architectures to leverage the computing capacity of the server cluster. These software architectures typically integrate a number of Free and Open Source Software (FOSS) products to produce an integrated application solution. A key component common to most, if not all, Big Data architectures is the Hadoop Distributed File System (HDFS). HDFS is used to efficiently distribute and manage large volumes of data across the computational nodes so that the data resident on each node can be processed “locally”, i.e., without excessive network Input/Output (I/O). Another benefit of HDFS is that it also achieves fault tolerance through built-in replication of data across nodes, usually replicating data across a minimum of three nodes. This permits any node to fail without impacting the cluster’s overall availability and its capacity to continue processing. A resource manager is another key complementary technology to HDFS. A resource manager is responsible for coordinating work done at the nodes, i.e., process the data residing on each node as necessary. Resource managers typically integrate a job scheduler/tracker that distributes the code to be executed at each node. Once submitted for execution to the nodes, the manager monitors execution and upon conclusion of all work, consolidates the intermediate results from the individual nodes at a central location to deliver a final result. The first, and most well know of these resource managers, is MapReduce. Since then, other resource such as Yarn, Mesos and Spark have been developed to address some of MapReduce limitations.

Elastic scaling is an enhanced implementation of horizontal scaling. This type of scaling provides an automated method to dynamically provision additional worker nodes for the computing cluster as utilization within the cluster approaches preset utilization thresholds or, conversely, to reduce the number of nodes in the cluster as demand falls back below the threshold. Elastic scalability is available in relatively advanced and large scale compute clusters/data centers where various consumers will at any point in time be scaled in or out to provide better budgetary and hardware resource utilization. Elastic scalability is particularly interesting in the context of the SEMI ODE since the ebb and flow of vehicular traffic during the course of the day is directly reflected in the volume of data traffic within the CV data network. Auto scaling could therefore allow the processing capacity of the SEMI ODE to expand and contract as needed throughout the day to match processing needs.

Lastly, **load balancing** provides a “traffic management” capability, which can monitor the resource utilization across several different providers in a cluster in order to route new computing requests to the lowest utilized members of the cluster, which is a group of machines or nodes that are configured to cooperatively process incoming traffic. Load balancers can also serve to implement fail-over, which occurs whenever specific compute resources become unavailable and computing flow is re-directed to other similar computing resources.

2.2. Architectural Design

Figure 2 illustrates the SEMI ODE system architecture at level 1. The Level 1 diagram identifies “application objects” that are involved with the operation of the system as it pertains to the SEMI ODE and the SEMI ODE itself. Appendix B APPENDIX B: CVRIA System Architecture Application Object Definitions describes/ defines the application objects.

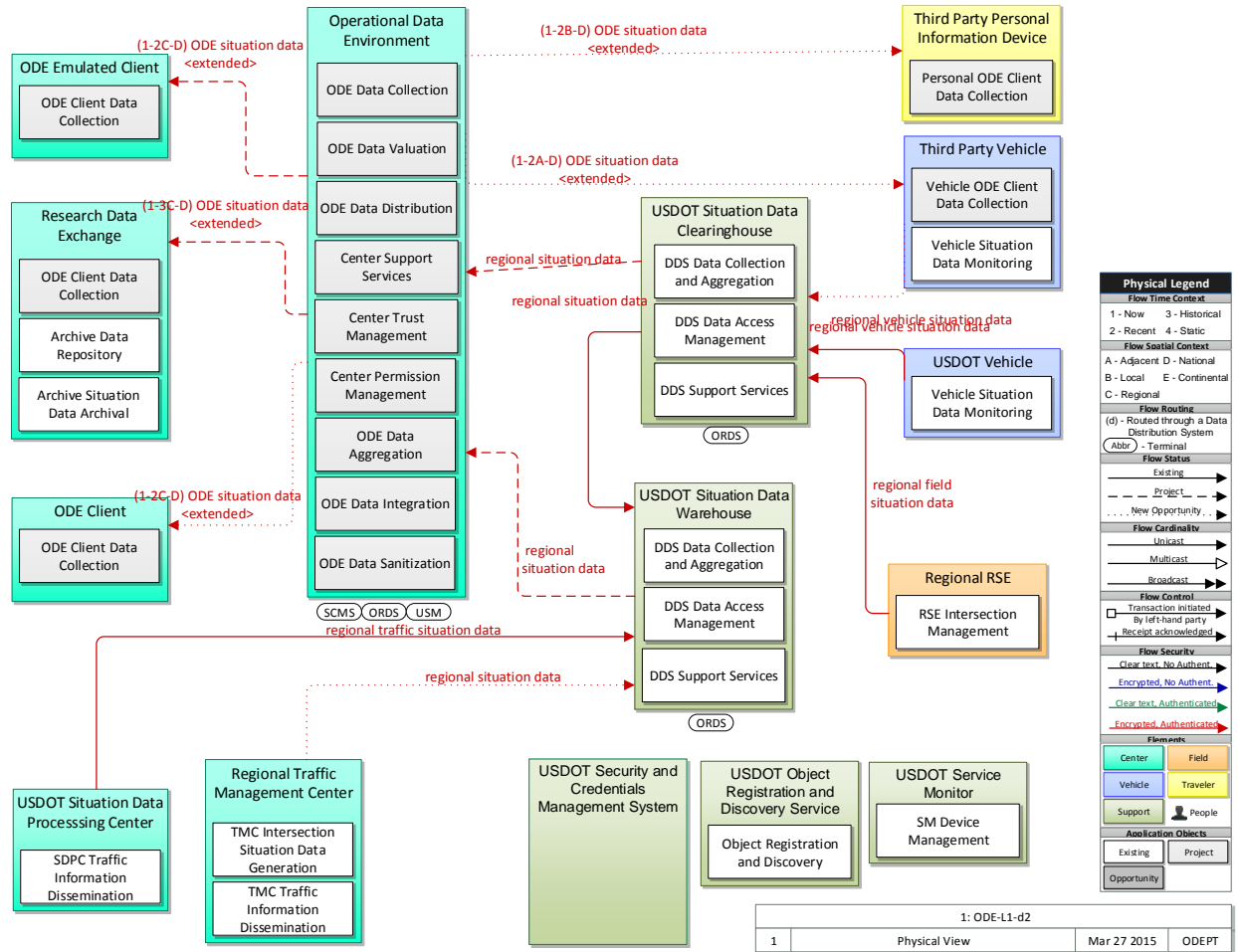


Figure 2 - System Architecture Physical View – Level 1

As mentioned, the ODE's core source of data is currently the US DOT SDC, SDW and SDPC. However, the SEMI ODE is designed with the capability to process data from other data sources. Once data are received from any source, the ODE's key capability of performing transformational processing on a given stream of data will allow each stream to be continuously subject to data quality checking using any appropriate number of data checks (which includes valuations as well as validation), sanitization, aggregation and integration processing of the same stream. Client applications will place requests to the SEMI ODE via APIs in order to receive data that support their operation. The requests may be for near real time or historical data.

The SEMI ODE application architecture can be envisioned as having four main tiers as shown in Figure 3:

1. the Public Facing Application
2. the Distributed Data Processing Engine
3. Security Controls
4. Operations and Management

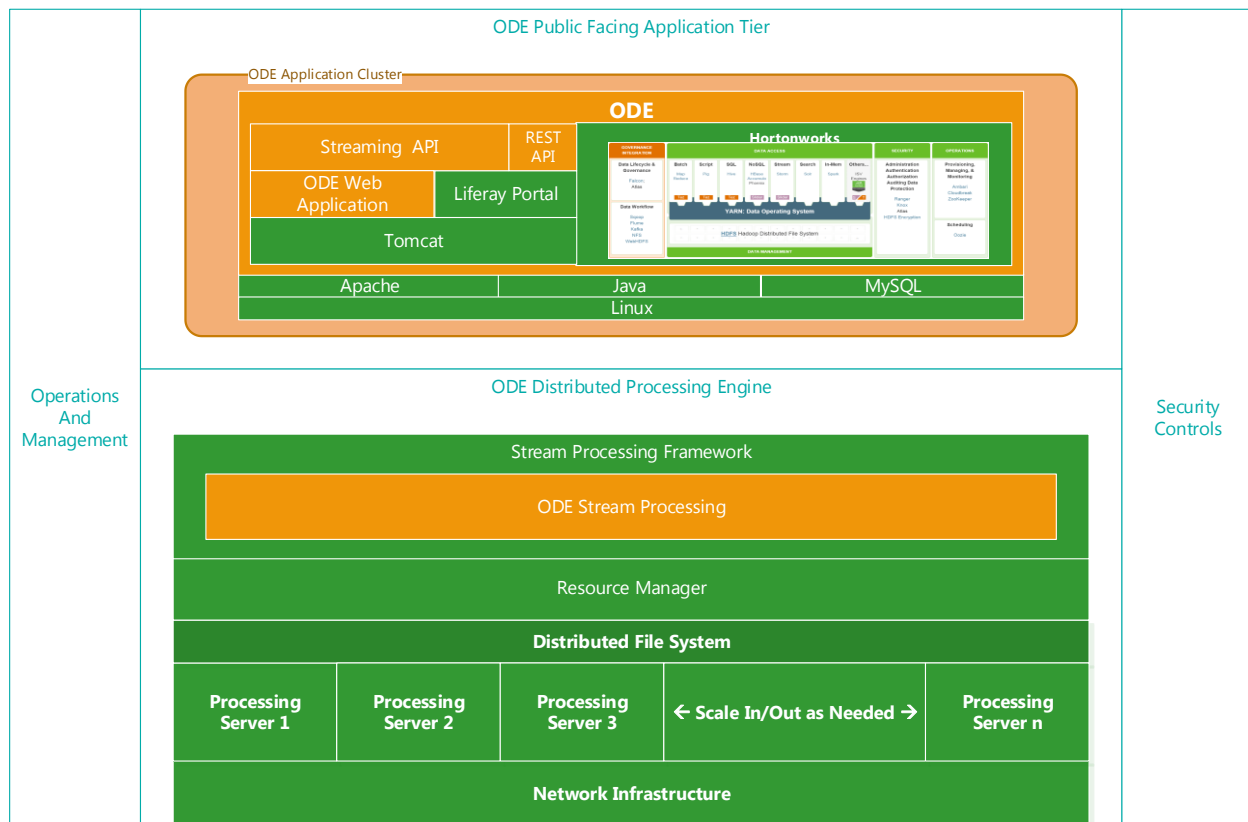


Figure 3 - SEMI ODE Technology Stack

The SEMI ODE public facing application tier shown at the top is further broken down into logical service tiers to enable application services to be loosely coupled with one another. Loosely coupled services will allow the public facing SEMI ODE application to horizontally and vertically scale application components as needed based on demand. Loosely coupling the application architecture into multiple tiers also enables the deployment of a system that is fault tolerant to application and service failures.

Security Controls for the SEMI ODE system will apply at all layers of the SEMI ODE system. The security controls will consist of a combination of network, application and access controls working in concert to ensure the confidentiality, availability and integrity of the SEMI ODE systems and data. More information on security controls is available in the System Security section.

The Operations and Monitoring subsystems will provide the means to monitor the health of the various SEMI ODE software and application components through the use of centralized reporting, logging and monitoring tools. The Operations and Monitoring subsystems will also enable the setup and tear down of computing resources to support auto-scaling based on the utilization of the various SEMI ODE application servers and services through the use of specialized tools and custom business logic.

2.3. Interfaces

The SEMI ODE interacts with the DDS and other data sources through the available network interfaces provided by these sources. The following section highlights the key components and features that support the ODE's interaction with the DDS.

2.3.1. DDS Interfaces

The DDS exposes three interfaces for client applications

1. Java Messaging Service (JMS) interface - for the client to receive subscribed data
2. Universal Datagram Protocol (UDP) Interface - for the client to send data requests and receive query results
3. WebSocket interface - for the client to establish data subscription, receive subscribed data, send query requests and receive query results

The real-time data is obtained via a long running *subscription*. Historical data are provisioned with a *query* request to the SDW or SDPC. Both subscription requests and query requests include geospatial parameters. Query requests also include temporal parameters such as start date and time and stop date and time. Subscription requests deliver latest data available as soon as they become available. Query requests deliver data that was previously captured and stored in the data warehouse (SDW) or the SDPC in the time period specified within the request.

The JMS and UDP interfaces work hand-in-hand in order to establish the data flow between the DDS and its client. UDP interface is used to establish trust and perform dialogs between the DDS and a client. JMS is used for data subscription to the SDC.

The SEMI ODE utilizes the WebSocket interface exclusively for all interactions with the DDS. The following sections briefly describe some of the characteristics of the JMS and UDP interfaces for the sole purpose of providing background information.

JMS Interface

The JMS Interface can be used for receiving real-time streaming subscription data from the SDC. To subscribe to and receive data, a potential consumer of data must do the following:

- Step 1: Create a subscription via Data Subscription Service Dialog for Vehicle and/or Intersection Situation Data (EVSD, ISD)
- Step 2: Create a JMS listener to receive messages

UDP Interface

The UDP interface can be used for dialogs between a DDS client and the DDS. There are significant challenges associated with the UDP interface, including:

- Implementing CV UDP based dialogs has a steep learning curve and requires significant programming skills.
- Difficult to implement clients residing behind a corporate firewall
- A firewall on the DDS side has to open a new port for each partner's IP address

Dialog messages sent and received via UDP as well as the data query results are in binary, Distinguished Encoding Rules (DER) encoded, ASN.1 format².

WebSocket Interface

The DDS provides a WebSocket interface as an alternative distribution method better suited for higher-volume center-to-center distributions due to its lower overhead aspects of the protocol. It is also more suitable for internet distribution due to its standard adoptability among internet based applications.

- The WebSocket server is co-located within the DDS and acts as an "internal super client" using the JMS and UDP interfaces.
- The server exposes query and subscription functionality to the external clients via WebSockets and Secure Sockets Layer (SSL) which supports X.509 certificates.
- Standard based: IETF RFC 6455 The WebSocket Protocol
- TCP based so data delivery is guaranteed
- SSL ports are usually open in corporate firewalls
- SSL traffic is rarely blocked by corporate firewalls

Dialog messages sent and received via WebSocket interface are in Java Simple Object Notation (JSON) text format. Data subscription request messages and warehouse query request messages are also in JSON text format.

Subscription data streams are always ASN.1 BER encoded format at the time of this writing but there are plans to switch to the UPER (Unaligned PER encoding) in the near future. The subscription request may specify the format of the response as:

- Hex format
- Base64 format

Warehouse query results are also ASN.1 encoded. The query request may specify the format of the response as:

² The SEMI ODE utilizes OSS Nokalva ASN.1 Compiler and Runtime software framework that was acquired specifically for use by ODE and other US DOT ITS Connected Vehicle Data programs. Its primary function is to decode inbound ODE messages that arrive in ASN.1 format.

- Hex format
- Base64 format
- JSON format

The SEMI ODE requests data in Base64 format due to the smaller size of Base64 messages.

The following criteria were used in the decision to use WebSocket interface instead of the UDP/JMS interface.

Table 1 – DDS Interface Selection Decision Matrix

| Criteria | WebSocket | UDP/JMS |
|-------------------------------------|-----------|---------|
| Easier to use and adopt | Y | N |
| Suitable for Web Clients | Y | N |
| Certificate-based encryption | Y | Y |
| Standards based | Y | Y |
| Reliable Delivery | Y | N |

WebSocket Interface Security Model

Only authorized users have access to the WebSocket server. The screen image below is from the server's user registration page

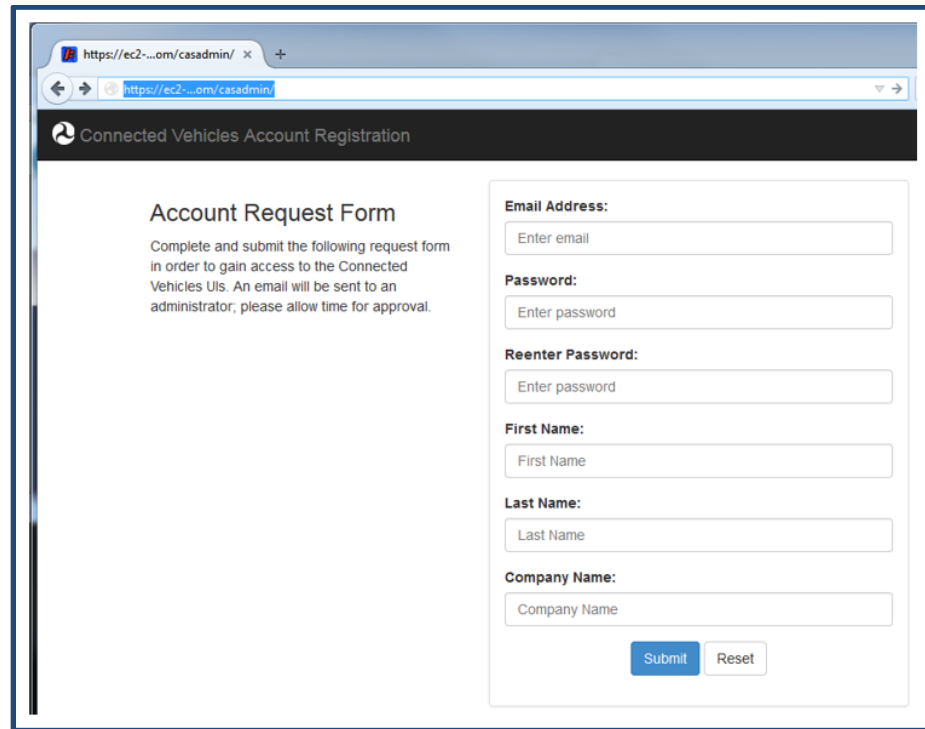
A screenshot of a web browser displaying the 'Connected Vehicles Account Registration' page. The browser's address bar shows 'https://ec2-...om/casadmin/'. The page has a dark header with the text 'Connected Vehicles Account Registration'. Below the header, the main content area is titled 'Account Request Form'. To the left of the form, there is a paragraph: 'Complete and submit the following request form in order to gain access to the Connected Vehicles Uls. An email will be sent to an administrator, please allow time for approval.' The form itself is on the right and contains several input fields: 'Email Address:' with a placeholder 'Enter email', 'Password:' with a placeholder 'Enter password', 'Reenter Password:' with a placeholder 'Enter password', 'First Name:' with a placeholder 'First Name', 'Last Name:' with a placeholder 'Last Name', and 'Company Name:' with a placeholder 'Company Name'. At the bottom of the form are two buttons: 'Submit' (blue) and 'Reset' (white with a blue border).

Figure 4 - DDS Account Request Form

SSL is used to protect data in transit. Data are not wrapped into signed or encrypted 1609.2 envelopes but the DDS may add support for 1609.2 certificate authentication in the future.

2.3.2. ODE Interfaces

The SEMI ODE Client applications will be able to submit one or more concurrent subscription and/or query requests to the ODE. The SEMI ODE will consolidate the requests based on the request parameters in order to determine what subscription and/or query requests it must submit to DDS and other data sources. This task is performed by the “Request Manager” component described in the upcoming sections of the document.

Initial access to these interfaces will be restricted to the SEMI ODE development team, but once sufficient testing is complete, any individual who successfully registers at the SEMI ODE Portal web site will be able to access the SEMI ODE interfaces. At that time, API documentation will be provided to new users via the Portal. The word “individual” is intended to identify a person who will develop an application leveraging SEMI ODE data stream(s). As a service provider, the SEMI ODE does not currently implement a granular API access model that would reflect the notion of a hierarchy of system stakeholders, which, by extension, would infer a governance model. If the desired SEMI ODE deployment, for example, is targeted to exclusively help state and local entities in the management of their transportation assets, the API’s can be restricted to this group. Currently, the design allows anyone registering with the SEMI ODE to be able to

leverage SEMI ODE provisioned data within their application. Initially, this group will be restricted to immediate project personnel.

The SEMI ODE provides two types of API's to its subscribers. These are a REST API and a streaming API. The reason for offering two API's is best understood by describing their differences.

The RESTful API is provided for administrative functions only.

The streaming API provided for SEMI ODE data delivery and begins to stream data to the application that invokes it immediately upon receipt of the subscription request and it continues to do so indefinitely. This is consistent with the SE-MI clearinghouse mode of operation, which forwards real-time EVSD and/or ISD data. The streaming API operates on a "good until cancelled" basis or until there is no data from the upstream source. In other words, the streaming API connection remains connected indefinitely whether there is any data to be sent or not.

ODE RESTful (REST) API

The SEMI ODE RESTful API employs *synchronous* calls to SEMI ODE services for use by clients and is used primarily for sending administrative commands to the ODE. The only data item provided by the REST API is the EVSD data.

Table 2 illustrates a high-level specification of the REST API. The API details will be provided in a separate document.

Table 2 – SEMI ODE REST API

| Description | API Type (REST Streaming) | Resource | Request Parameters | Request Body |
|-----------------------------------|-----------------------------------|--------------|-----------------------|-------------------------------------|
| Log in to ODE³ | REST | /auth/login | UN/PW | Token |
| Log out of ODE⁴ | REST | /auth/logout | Token | |
| Discover Data⁵ | REST | /datatypes | Token | [EVSD,isd,T SD,map,spat ,agg] |
| Get Users⁵ | REST | /users | | [{User}...] |
| Create User⁵ | REST | /users | {User} | |

³ Response Body will contain a security token. User Name and Password will be in the Authorization Header, so the request body will be empty.

⁴ Log out API will revoke a non-expired security token.

⁵ This capability will be provided in a future release of SEMI ODE

| Description | API Type (REST Streaming) | Resource | Request Parameters | Request Body |
|----------------------------------------|--------------------------------|------------------------|--------------------|-----------------------|
| Read User⁵ | REST | /users/{userId} | | {User} |
| Update User⁵ | REST | /users/{userId} | {User} | |
| Delete User⁵ | REST | /users/{userId} | | |
| Get Subscriptions⁵ | REST | /subscriptions | | [subId1, subId2, ...] |
| Create Subscription⁵ | REST | /subscriptions | {Subscription} | |
| Read Subscription⁵ | REST | /subscriptions/{subId} | | {Subscription} |
| Update Subscription⁵ | REST | /subscriptions/{subId} | {Subscription} | |
| Delete Subscription⁵ | REST | /subscriptions/{subId} | | |

ODE Streaming API

The SEMI ODE Streaming API employs *asynchronous* calls to SEMI ODE services which are forwarded to the SE-MI clearinghouse and other real-time data sources. Clearinghouse information is received indefinitely by the client application or until the application deliberately disconnects from the SEMI ODE or connection between SEMI ODE and the application endpoint is disrupted abnormally. Because of this undefined termination, the streaming API will typically forward a large volume of data to the client application under typical circumstances.

As mentioned earlier, the DDS and by extension ODE, provide real-time data as well as historical data. The real-time data is provisioned as a *subscription* while the historical data is provisioned as a *query* request to the SDW or SDPC. Due the potential volume of data, both real-time and historical data are provided using the Streaming API.

Real-time data from the SDC are propagated by the DDS as soon as the data is available and will continue the flow of that as long as the connection is maintained, regardless of the availability of data. If there is no data available, the client will stay connected to the SEMI ODE but will not receive any data until data is available.

Historical data from either the SDW or the SDPC are by definition not “real-time”. The SDW retains data for a depositor-selected period of time as specified by the timeToLive parameter of the data deposit message (an enumerated value indicating 1 minute, 30 minutes, 1 day, 1 week, 1 month, or 1 year). This optional parameter is available only for ISD and TSD messages. If not

specified and for EVSD messages, the data will be retained in the SDW for a default duration of 30 minutes. The SDPC on the other hand retains data for a much longer period of time⁶. Applications requiring historical data must specify the source of data in the data query request. Since queries submitted to the SEMI ODE are historical by nature, a start and end date and time should be provided, otherwise all data available in the SDW or SDPC database will be returned. Nevertheless, the time series data returned by SDW and SDPC are finite when compared to clearinghouse subscriptions, which will stream data indefinitely to the client application (or until the underlying WebSocket connection linking the SEMI ODE and endpoint is broken for whatever reason).

Table 3 documents the available options for the SEMI ODE Streaming API.

Table 3 – ODE Streaming API

| Description | API Type (REST Streaming) | Resource | Request Parameters | Request Body |
|------------------------------------|--------------------------------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| Request Vehicle Data Stream | Streaming | sub qry/veh/?token={token} | token nwlat nwlon selat selon [startDate startDateOperator endDate endDateOperator orderByField orderByOrder | { Vehicle Data } |

⁶As of this writing, the SDPC is configured to retain data for 6-months

| | | | | |
|-----------------------------------------|-----------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| | | | skip limit] | |
| Request Intersection Data Stream | Streaming | sub qry/int/?token={token} | token nwlat nwlon selat selon [startDate startDateOperator endDate endDateOperator orderByField orderByOrder skip limit] | { Intersection Data } |
| Request SPaT Data Stream | Streaming | sub qry/spat/?token={token} | token nwlat nwlon selat selon [startDate startDateOperator endDate endDateOperator | { SPaT Data } |

| | | | | |
|--------------------------------|-----------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| | | | orderByField orderByOrder skip limit] | |
| Request Map Data Stream | Streaming | sub qry/map/?token={token} | token nwlat nwlon selat selon [startDate startDateOperator endDate endDateOperator orderByField orderByOrder skip limit] | {Map Data} |
| Request Advisory Data | Streaming | qry/adv/?token={token} | Token or UN/PW nwlat nwlon selat selon startDate startDateOperator | {Advisory Data} |

| | | | | |
|-------------------------------|-----------|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| | | | endDate endDateOperator orderByField orderByOrder skip limit | |
| Request Aggregate Data | Streaming | sub qry/agg/?token={token} | token or UN/PW nwlat nwlon selat selon [startDate startDateOperator endDate endDateOperator orderByField orderByOrder skip limit] | {Aggregates Data} |

Sub|qry part of the Resource URI defines the “request type”, Subscription or Query. If the request type is query, the following parameters are required in the request:

- startDate
- startDateOperator
- endDate
- endDateOperator

- orderByField
- orderByOrder
- skip
- limit

Please refer to the SEMI ODE API Specifications document for details about the request parameters.

2.4. ODE Database Design

In a traditional application environment, database technology is deployed to facilitate the storage of information so that it may be more easily, securely and reliably accessed by its users at different points in time. This includes the ability to perform the basic create, read, update and delete (CRUD) functions as these are associated with application users via user roles within the context of the application. Since the ODE's primary purpose is to forward real-time data, such as EVSD data, from production to consumption endpoints, there is no requirement to store data within the SEMI ODE and consequently no need for a database in its traditional role, especially since the CVRIA already identifies the Research Data Exchange (RDE) as the focal point for long term archiving of data.

In light of the above distinction, the SEMI ODE design will initially leverage only the Identity and Access Management (IAM) functionality of a database to provide basic user registration and authentication. This authentication mechanism will be triggered each time an end user application invokes one of the Restful API's that are published by the SEMI ODE (as detailed in Table 1 of the previous section). This also simplifies issues related to privacy when storing data, thereby simplifying SEMI ODE design, functionality and vulnerability to privacy concerns.

This IAM-only functionality implies that the SEMI ODE will not capture and store application subscription information in a database and that subscribing applications will therefore be responsible for maintaining all necessary parameters to invoke the relevant API(s) resulting in the (re)establishment of data flows to the application. This feature may be added to SEMI ODE at a later time if deemed valuable.

The only other type of data maintained by the SEMI ODE is a collection of active subscription profiles, including certain metadata that are associated with each subscription for managing the flow of data and processing performed on the data. These active subscriptions are to be maintained in-memory and there would not be any need for persisting that data on disk for the purpose of restoring it after a reboot. We refer to this data store as Active Subscription In-Memory Cache (or ASIM Cache for short). We envision that a distributed caching framework such as REDIS or MemCached would serve this purpose well when needed. Currently however, ASIM Cache is maintained in memory within the application heap space.

2.5. System Security

Security is a key component of any system, but since one side of the SEMI ODE interface reaches into the CV network, vulnerabilities are a significant concern. Therefore, the SEMI ODE MUST clear through SCMS authentication and certification dialog before being allowed to subscribe to any data requested from the DDS.

A layered security approach is the traditional design approach to mitigate security risk.

2.5.1. Network Level Security

The SEMI ODE will employ multiple network controls to ensure the privacy and security of the SEMI ODE systems and the data therein. The SEMI ODE computer network will utilize various network subnets that will logically isolate various SEMI ODE components. Network access control lists (NACL) will be configured to control the flow of network data between computer systems that reside on different subnets. Firewalls will be configured and deployed to further define how servers and services will communicate with each other in addition to the NACLs. System Administrator access to SEMI ODE private resources will be restricted to pre-approved users and computers through a combination of network firewalls and NACLs.

Public SEMI ODE services are protected against eavesdropping by the use of SSL technology (X.509 Certification). Additional security is provided by the fact that SEMI ODE components communicating inwards to the CV network are isolated on a separate subnet and this subnet is only accessible by a single defined interface to the public facing subnet, the latter hosting components such as the IAM and the subscription engine.

2.5.2. Database Level Security

Database level security will be layered to ensure data availability, integrity and confidentiality. At the network level, the database server will reside on a private subnet network, which will have NACLs applied to it to allow or deny traffic⁷. The database servers will have their own firewall to further restrict how external services, systems and users access the servers. The third layer will be leveraging internal database security mechanisms to control user access to data and the type of operations the user can perform.

2.5.3. Application Level Security

ODE applications will execute within the Operating System (OS) with the least required level of privileges. This approach is a key best practice in reducing security risks from intrusions since it

⁷ Not applicable during the initial release of the ODE as the ODE web application resides on a single server that contains the IDAM Database. No separate database server is scheduled to be deployed at this time.

restricts the possible damage that can be caused by any undetected attack vector that may be present in the code. The practice requires that a piece of code that executes within the OS be associated with an OS level agent which has the minimum level of OS privileges needed to properly execute. The usage of minimal privileges will ensure the application server integrity and data integrity from application faults or from unauthorized use. Secure class loading and verification mechanisms provided by Java programming language, which is the language used to develop the ODE, ensures that only legitimate Java code is executed. Even though SEMI ODE is written in Java programming language, the SEMI ODE application interface is a language agnostic Web Interface which can be used by applications written in any programming language.

The SEMI ODE client applications must register with SEMI ODE directly with the provided Identity and Access Management (IAM) software in order to gain access to the SEMI ODE API. A web-based administrative console will be provided for users to register and receive credentials needed to access the system.

2.6. ODE Process Flow

The process flow managed by the SEMI ODE is depicted in Figure 5 below. The first step in the flow is initiated by a client application when it makes a data request to the ODE. Upon receipt of the subscription at the ODE, the Request Manager (RM) is responsible for parsing the request details and communicating with three other components within the ODE:

1. The RM derives metadata from the subscription request in order to pass this on to the Collector. Currently this metadata provides the context for the subscription request so that the Collector can subsequently place the data on the correct output queue.
2. The RM also communicates with the Connector. The Connector will create and submit an upstream data request to an external data source.
3. The RM communicates with the Distributor. The Distributor will be given the WebSocket session object generated when the client application initiated its request to the SEMI ODE and the outbound topic associated with the WebSocket session.

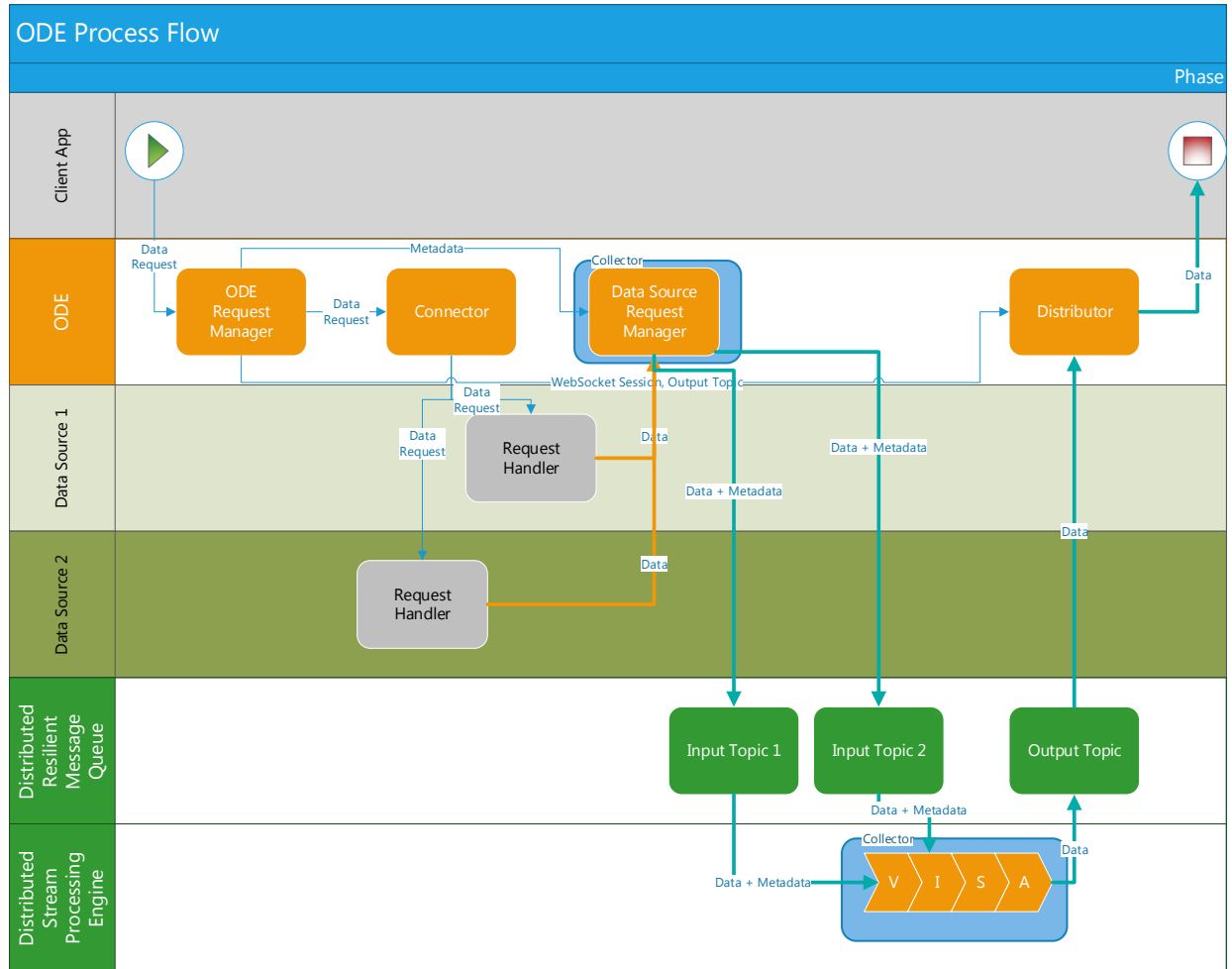


Figure 5 - SEMI ODE Process Flow Diagram

At the time of this writing, the Collector receives the subscription stream from the external data source, merges the subscription metadata with the return data flow, and places the combination of data and metadata onto the input topic dedicated for consumption by the Distributed Stream Processing Engine (DSPE). In a future enhancement, the metadata will be made available to the Collector and DSPE via shared in-memory cache in order to reduce internal network bandwidth utilization.

For each data record received, the DSPE will be able to identify the data processing requirements as defined by the metadata and provide the appropriate combination of predefined Validation, Integration, Sanitization and Aggregation for that record. In a future enhancement, the metadata may explicitly specify the modular functions that the DSPE is being asked to perform on each piece of data.

As the messages are processed and exit the DPSE, the data is placed on an outbound queue to await distribution. As a final step, the Distributor associated with the each outbound queue will remove the record from the queue and route the message to the associated client application.

This page intentionally left blank

CHAPTER 3. COMPONENTS

This section describes the major system components of the SEMI ODE as depicted in Figure 6. Figure 6 - SEMI ODE Application Components below.

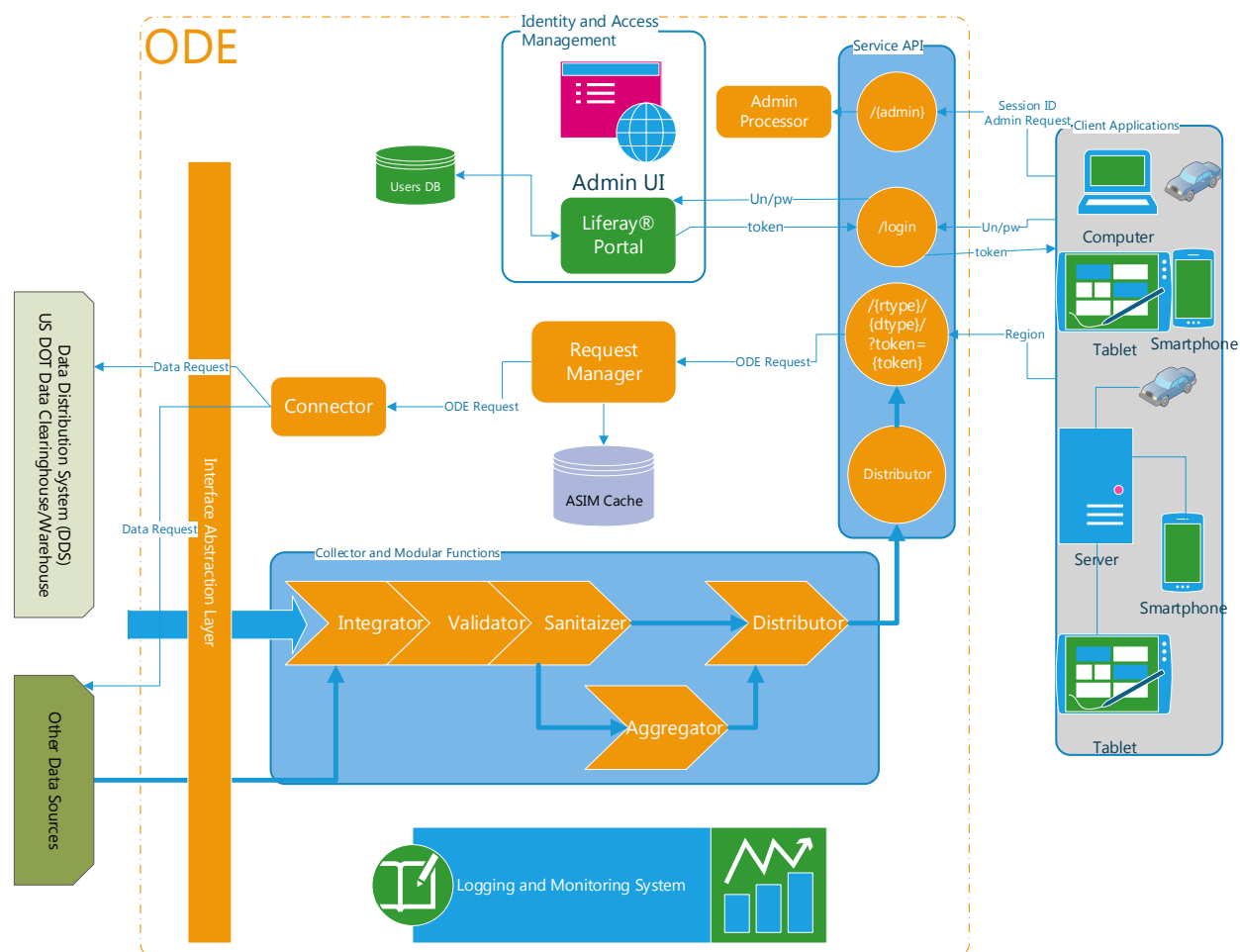


Figure 6 - SEMI ODE Application Components

The components shown in Figure 6 can be grouped in four major functional areas:

Request Processing Functions: Section 3.1 describes the components that implement these functions, which include the Service API, Identity and Access Management (IAM), Request Manager, and the CVRIA Application Object

- Center Support Services
- 1. Connector

Data Processing Functions: Section 0 describes the components that implement these functions, which include the Collector, the CVRIA Application Object

- ODE Data Collection

Integrator, the CVRIA Application Object

- ODE Data Integration

Validator (Data Valuation), the CVRIA Application Object

- ODE Data Valuation

Sanitizer, the CVRIA Application Object

- ODE Data Sanitization
2. , and the Aggregator.
 3. Data Consumption: Two types of applications have been developed to interface with the SEMI ODE:
 - a. Test applications that were developed for the purpose of testing and validating ODE functions. These applications DO NOT implement any particular ITS or Connected Vehicle algorithm other than to subscribe and query the SEMI ODE and verify that the data received is what the application expected to receive and at the rate that was expected.
 - b. Emulated CV application that were retrofitted with the SEMI ODE interface capability to perform actual mobility functions using real-time subscriptions to the SEMI ODE. Please refer to the “ODE System Test Plan” document for details about the emulated CV applications.
 4. Logging and Monitoring: This section describes the tools and techniques used to monitor and maintain the operations of ODE.

The following sections describe each SEMI ODE component within the above noted 4 functional areas in further detail.

3.2. Request Processing Components

Request Processing components of SEMI ODE are involved in the processing of the subscription and/or query requests placed with ODE. Figure 7 below illustrates the sequence of events and activities when a data request is received by the ODE.

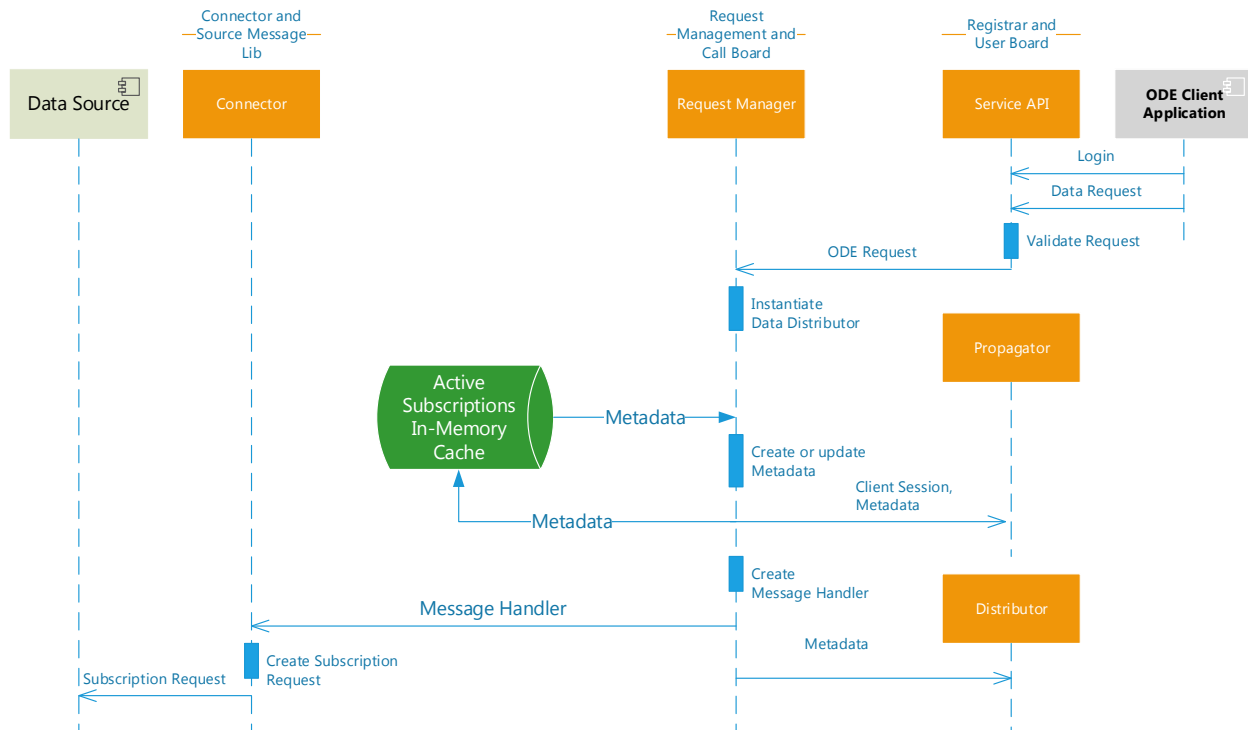


Figure 7 - Request Processing Sequence of Operations

The request process flow is initiated by the client application authenticating itself using the Service API *login* function. The *login* function is a REST API which returns a security token back to the caller. The caller will use the security token in the subsequent API calls to either the REST API or Streaming API.

Once it obtains this security token, the SEMI ODE client application will embed the token in the data service request (a WebSocket message) that it places to the ODE. The Service API will validate the security token along with other request parameters and pass it on to the Request Manager if all validation is successful.

The Request Manager then instantiates a Distributor thread/object dedicated to the requesting client and queries the Active Subscriptions In-Memory (ASIM) Cache that is established for maintaining the collection of all SEMI ODE requests to determine if the current request can be derived from one of the existing subscriptions. For example, if an existing subscription provides data for a larger geographic area than the current request, then the current request is said to be “derivable” from that subscription by providing a filter that will let only the data belonging to the current request to pass through to the client. If the data can be derived from an existing

subscription, the RM will pass the client's Session object and the Metadata to its Distributor. The Metadata contains the message queue that will contain the data, therefore the Distributor will be able to retrieve the data from the specified queue and propagate it to the destination client.

If there is no existing subscription that contains all of the data for the current request, the Request Manager will create a new subscription by consolidating one or more subscriptions into one subscription. This is done in order to minimize the load imposed by the ODE's client applications on the upstream data sources. This function is an important responsibility of a data aggregator because it relieves the data sources from having to support thousands or perhaps millions of clients directly. If the system will need to be scaled in the future, first the SEMI ODE which is designed with utmost scalability features will be scaled before having to scale the data sources. See section 3.2.3 Request Manager for details of the algorithm used by the RM to consolidate data requests.

The Request Manager creates subscription metadata by parsing the parameters provided in the subscription request and associating the request with a specific queue or message topic (Note: the terms queue and message topic are often used interchangeably). The assigned queue will be used to momentarily store subscribed data as it arrives at the SEMI ODE and before it is processed. The metadata may also include instructions about the processing to be performed on the data records.

The Request Manager will also create a Message Handler (MH) for receiving the data. The MH object is delivered to the Connector and the Connector will utilize the message handler for receiving and decoding the message. The message handler is a part of the Collector function and is specialized for each data source and data type.

3.2.1. Service API

Purpose

The purpose of the Service API is to provide access to the ODE's capabilities and streaming data via external web interfaces.

Function

The SEMI ODE supports two types of Request Processing API's:

1. REST API: Performs a request and returns the results
2. Streaming API: Submits a subscription or query to the DDS and maintains a connection to the application that initiated the request.

The following actions are performed for both APIs (unless otherwise noted in a specific action details):

- Receives inbound requests from external applications.
- Validates the requester's authentication key as passed in by the request.
- Validates the request parameters and analyzes the request for the purpose of authorization.

- If the requester has permission to perform the function of the request, launches processes to perform the request.
- If the request type is a REST API, the function would be of administrative type, in which case, it will be performed and results returned.
- If the request type is a Streaming API, the request is forwarded to the Request Manager for processing while at the same time, the request response is returned to the requester with a status code (see Interface Specifications for the details of the status response).

Dependencies

This component depends on the Identity and Access Management (IAM) for authentication and authorization functions. Request Processing Components and CVRIA Application Object

- Center Support Services
- Data Processing Components for downstream data services.

CVRIA Application Object

- Center Support Services

3.2.2. Identity and Access Management (IAM)

Purpose

The purpose of the Identity and Access Management (IAM) is to maintain a database of users, their roles, and the groups to which they belong.

Function

This component will be implemented by leveraging the functionality provided by the Liferay® portal Community Edition software framework. Liferay provides:

- User authentication
- Role-based and permission-based authorization
- User Groups
- Administrative UI
- Framework for developing web-based user interface for future enhancements to the application.

Dependencies

The functionality provided by Liferay depends on a Relational Database Management System (RDBMS) for maintaining the state of the portal. Currently, MySQL database is being used to provide the database for the Liferay portal.

CVRIA Application Object

- Center Trust Management
- Center Permission Management

3.2.3. Request Manager

Purpose

The purpose of the Request Manager is to minimize the number of subscription requests to the DDS and other data sources by consolidating incoming requests and assisting the Distributor and Propagator with propagation of data to the appropriate recipients.

Function

This component will evaluate each subscription request in relation to existing subscription requests in order to determine whether this latest request can leverage existing requests. The goal of this evaluation is to produce efficiency gains by leveraging existing pipelines.

As of this writing, the request manager identifies a request as “existing” based on the hash-code of the request. This technique yields *identical* requests as “existing” requests. If a request is even slightly different from another request, it is considered a *new* request and the SEMI ODE will not attempt to merge or consolidate the two requests. In future enhancements of the ODE, a more sophisticated algorithm shall be used to identify requests that can be extracted from non-identical, yet encompassing subscriptions and providing a filter in order to eliminate non-overlapping records.

Figure 8 below depicts a non-exhaustive set of possible subscription scenarios that could happen in any order but shown here in one particular sequence to illustrate the functionality of the Request Manager.

On the left, there is a sequence of events and actions and on the top right corner there is representation of the geographic areas mentioned in the activity diagram on the left.

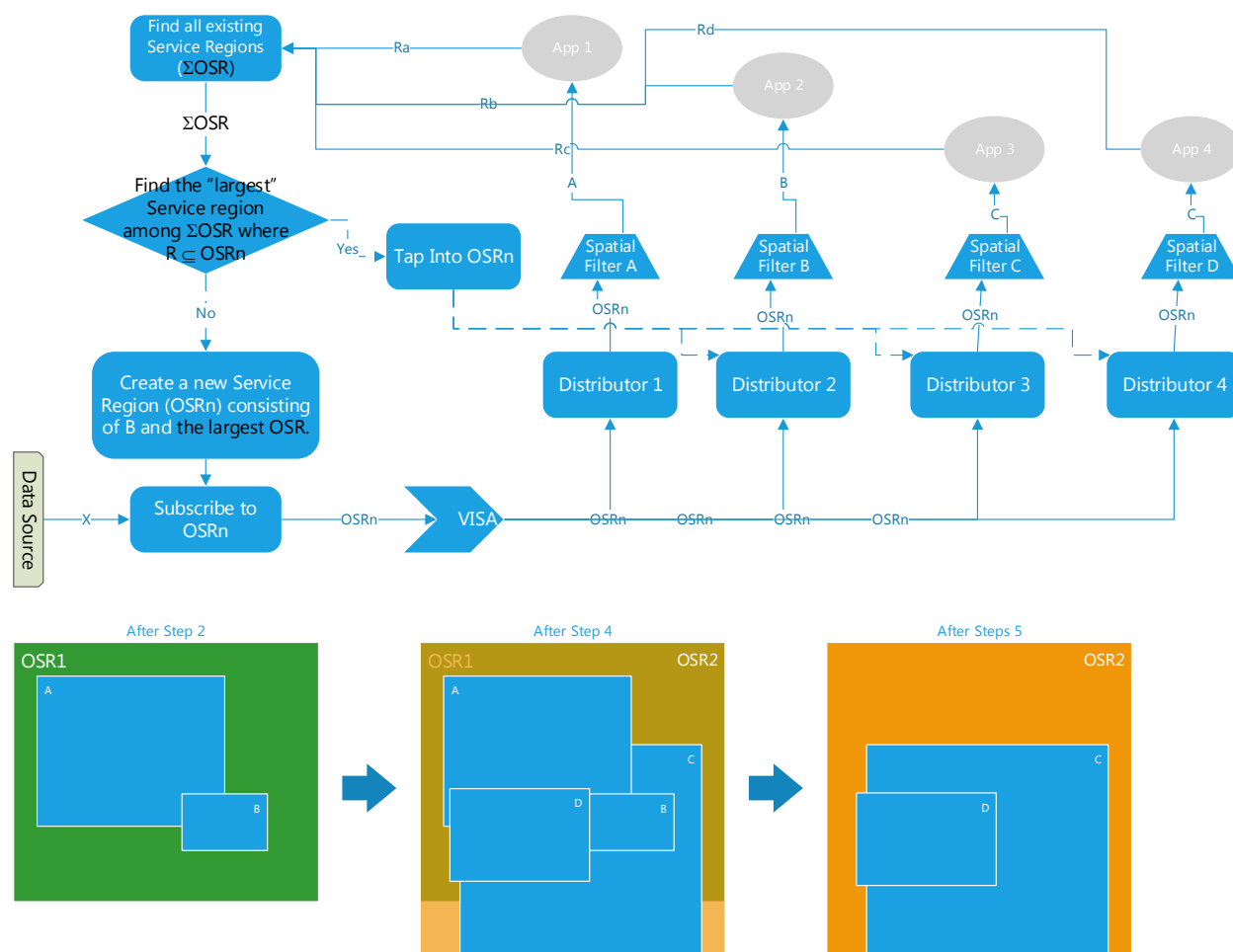


Figure 8 - Request Manager Subscription Consolidation Algorithm

The scenarios depicted in Figure 8 follow the sequence outlined below:

1. Application 1 is the first application connecting to the SEMI ODE (after a “system⁸” restart). App 1 requests for data from geographic Region A. Since App A is the first app to request data from the ODE, no other subscription exists so the Request Manager will initiate a subscription to the default SEMI ODE Service Region (OSR1) and dispatch Distributor 1 to deliver the data to App 1. This results in the Distributor deploying a spatial filter which only allows records that originate from Region A to be sent to App 1.

2. Application 2 subsequently requests data from Region B. The Request Manager will determine if Region B falls completely within any of the existing subscriptions. As Region B happens to fall within OSR1, the Request Manager will configure and launch Distributor 2 to tap into OSR1 subscription. Distributor 2 deploys a spatial filter that would remove any records that do not fall within Region B.
3. Application 3 follows with a subscription request for Region C which is partially or wholly outside all existing OSRs. The Request Manager will create a new subscription to a new Service Region (OSR2) which encompasses OSR1 and Region C and subscribes to the expanded OSR2 region and launches Distributor 3 to service OSR2. Distributor 3 deploys a spatial filter that would strip out any records that do not belong to Region C. In order to avoid any disruption to data propagated to App1 and App2, Distributors 1 and 2 will continue to receive data from OSR1 even though OSR2 is a larger service region and encompasses OSR1.
4. Application 4 requests data for Region D which falls wholly within OSR2 (or even OSR1 for that matter). SEMI ODE will deploy a Region D specific spatial filter to serve App 4 and will tap into the larger OSR2 subscription.
5. Application 1 and 2 discontinue subscriptions to data. SEMI ODE will remove subscription to OSR1.

In a future enhancement, Machine Learning can be leveraged to implement a more effective algorithm based on heuristics.

Request Manager will create a unique Distributor for applications when they initiate a data request. As part of the Distributor's creation, subscription specific metadata residing within ASIM Cache will be passed to that relevant distributor which will allow the Distributor to create the appropriate geo-spatial filter for the data.

Dependencies

- Cached collection of active subscriptions that can be used to determine whether a new subscription results in a duplicated request

CVRIA Application Object

- Center Support Services

3.2.4. Connector

Purpose

The purpose of the CVRIA Application Object

- Center Support Services

Connector is to place subscription and query requests to the supported data sources in order to obtain data.

Function

Submits subscription or query requests to DDS and other data sources built and delivered by the Request Manager.

Dependencies

- Request Manager
- Collector
- Software Library or Framework in support of the supported data sources

CVRIA Application Object

- Center Support Services

3.3. Data Processing Components

3.3.1. Overview

The SEMI ODE is focused on the processing of data stream(s), which arrive primarily from the data clearinghouse but could also be received from ancillary data sources. As discussed in Section 0 -

This page intentionally left blank

System Architecture, processing streams for low latency applications create particular challenges. Traditional (i.e., non-stream) data processing models typically start with data that has been stored to disk as a first step in order to then be accessed by the application that is processing the data. This allows the application to explicitly control when and at what rate the disk I/O should occur. This “roundtrip” of data to storage on disk and back into RAM (Random Access Memory) for processing introduces relatively large amounts of latency since this operation is very slow even in a clustered computing environment where data and processing tends to be confined within single servers to reduce the amount of data that travels between servers. If the stream can be processed without the “roundtrip” by being placed directly in RAM, latency can be reduced. Reduced latency, in turn, increases the scalability of the architecture since higher volumes of data streams can be processed using the same hardware configuration. As the processing limits of the given hardware architecture are approached, the risk of losing data is increased since the streamed data are not stored to disk. This is where the benefits of a cloud computing environment come to play since virtual machines (VM’s) in a compute cluster can have their RAM upgraded to scale “vertically” and/or additional VM’s can be added to the cluster to scale horizontally. This permits the amount of collective RAM in the cluster to be expanded in an apparently limitless manner within the cluster. The same scalability applies to CPU capacity, which can be sized to properly match the increased amount of RAM.

Given the above approach to scaling, any incoming stream to the SEMI ODE can be processed by any given sequence of transformations to form a stream “pipeline” based on the needs of SEMI ODE data consumers. In this way, the SEMI ODE is not a single black box, but a series of loosely coupled modules that can be created and/or redesigned over time as the number of available streams increases and/or as data consumers have new requirements of the available data. The notional architecture presented in the SEMI ODE White Paper identifies five general categories of components capable of transforming data once it enters a pipeline. These include the collector, aggregator, integrator, sanitizer and valuator. We will define some instances of each of the five component categories below, but it is important to note that these are defined to illustrate the capability of the technology to process streaming data and they are not the definitive version of any component. Data sanitization, for example, can be achieved in many ways based on the data found in a particular stream, the privacy policies associated with data in the stream, and the type of data consumer. So over time, there will be several/many sanitization transformation components developed, and a particular subscription will be configured to leverage the appropriate combination of transformation categories/modules, including sanitization, if appropriate. Each pipeline can leverage any of these sanitization transformation(s) at any point(s) within the pipeline.

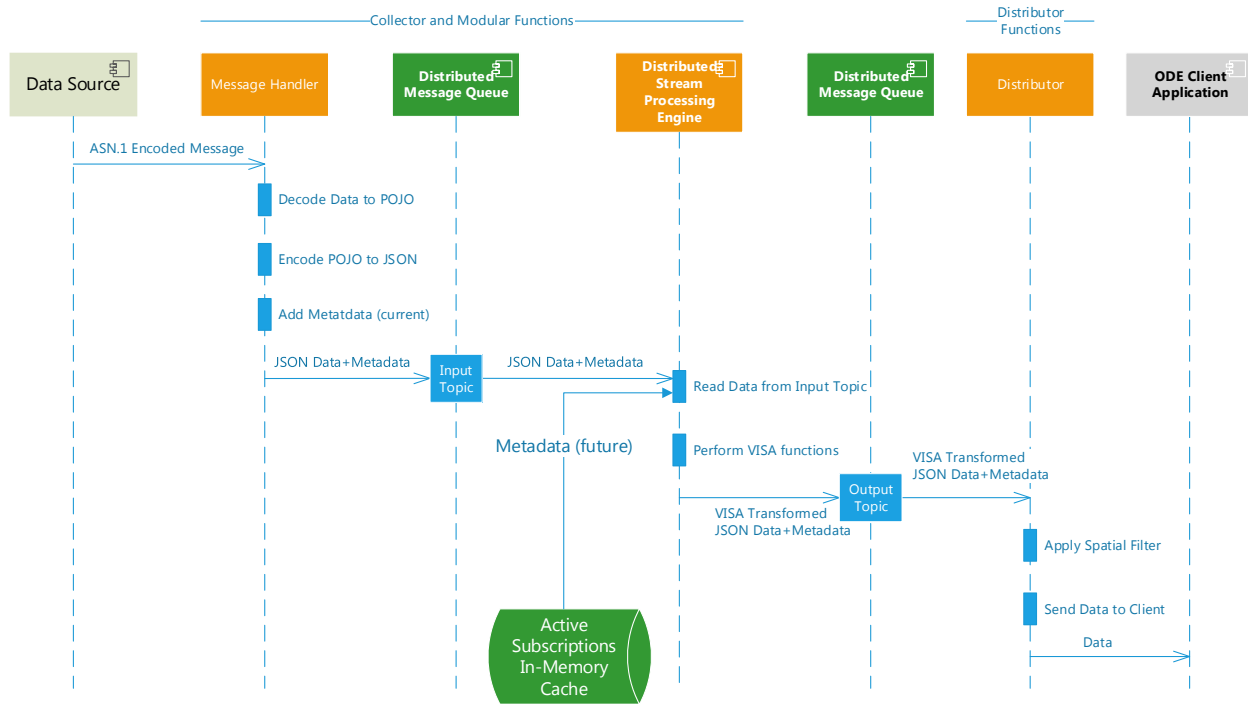


Figure 9 - Data Processing Sequence of Operations

Figure 9 illustrates the sequence of events and the actors involved in the processing of data streams between the data sources and the SEMI ODE client applications. The following sections describe the manipulations performed by the SEMI ODE in various processing categories.

Data Decoding

The data from the US DOT data clearinghouse is received by the SEMI ODE encoded in ASN.1 format. The SEMI ODE will be subscribing to the Base64 encoded version of the ASN.1 messages since this is more bandwidth efficient than the hexadecimal alternative.

Data Conversions

ASN.1 messages are decoded into binary format (from Base64) and then decoded into a Plain Old Java Objects (POJO) by the Message Handler. The decoded message may be a bundle of one or more messages of the same type, as is the case with EVSD messages. In such cases, the message is unbundled into individual records. Each record within a message is then converted to a format more suitable for the client applications to receive and process. For example, heading comes in as an integer between 0 and 28800 representing increments of 0.125 degrees, as defined by j2735 schema. The SEMI ODE will convert the raw heading data to a more straightforward and readily usable decimal number in a range of 0 to 359.9875 degrees.

Once the data is converted to the desired format, the SEMI ODE will encode the POJO formatted data into a JSON (JavaScript Object Notation) format. JSON formatting allows a serialized

version of the data to be shuffled around the network to worker nodes for distributed processing. It can also be easily parsed, is in a human readable format and enjoys a rich ecosystem of tools and frameworks that make manipulation and processing of the easier and more efficient. The human readable aspect of JSON formatted data, as well as the level of widespread support for it, make the standard ideal during the current Research and Development phase of the ODE. However, as JSON is not bandwidth and memory friendly, once the R&D phase of the SEMI ODE is concluded, the representation of data *within* the SEMI ODE should be managed using Java objects and other serialization technologies when crossing subsystem boundaries, such as going through message queues. Ultimately, it is anticipated that JSON formatting would be limited to the SEMI ODE interface with the SEMI ODE client applications in order to better serve client application interoperability.

Metadata

As of now, a small amount of essential metadata is associated with each data record to help manage the processing and routing of data. As metadata grows, it is envisioned that it will be stored and retrieved from the Active Subscriptions In-Memory (ASIM) cache. The ASIM Cache will provide a central, ideally distributed, repository of metadata storage for the ODE.

Message Queues

The SEMI ODE utilizes the open-source Kafka messaging technology for its *internal* data transfers. Kafka provides for a distributed and resilient message transfer technology which will enhance the ODE's reliability with regard to handling messages.

ODE Stream Processing Engine (OSPE)

When the message handler concludes its decoding, conversion, enrichment and serialization of the data, it stores the data into the message queue on which the SEMI ODE Stream Processing Engine (OSPE) is listening. The OSPE retrieves the data from the input queue/topic and performs the necessary data transformation functions that is the core processing of the SEMI ODE prior to distribution of the data to subscribers. These transformational functions will Validate and Sanitize the data (i.e. strip any Personally Identifiable Information or PII from the data), Integrate two or more data sources to complement, complete or correct the data, and perform Aggregation algorithms to derive analytical data from the data stream and present them to client applications.

OSPE deposits the data that it has processed back on to the distributed messaging framework but onto the specific outbound queue as defined by the metadata accompanying the data.

Data Distribution

At the tail end of the data processing flow resides the data distributor. The distributor will be listening on a queue that will provide the data to it. Once data arrives, the distributor will run the data through a spatial filter in order to remove any records that are not spatially relevant to the specific subscription. The data record may be unwanted because the application did not ask for it

or because the data arrived with corrupted spatial parameters. In either case, this filter provides for spatial validation and specialization of the data.

As a last step in the Data Processing Sequence of Operations, the validated and client-specialized data is then routed back to the client application from the Distributor.

The remainder of this section is dedicated to a review of all of the data processing components.

3.3.2. Collector

Purpose

The purpose of the Collector is to collect data from various supported data sources and forward the data streams to data processors.

Function

The collector listens to the incoming data streams and invokes the message handler registered by the Request Manager in order to perform the following functions:

- Decodes the data from ASN.1 format to Java objects based on ASN.1 schema definition files provided by the US DOT DDS system.
- Decodes the data from other data sources based on the published schema definition of the data.
- Passes data to downstream processing functions.

Dependencies

- Request Manager
- ASN.1 Compiled Java API and Runtime
- Other decoding libraries as required by the data source

CVRIA Application Object

- ODE Data Collection

3.3.3. Integrator

Purpose

The purpose of the integrator is to join data in one data source with data from another data source. The data sources joined can be characterized as static and finite, as would be the case for the merging of data from a relatively static road segment information, or dynamic and long-lived, as would be the case for the merging of data from a road weather data source. The integration of two streams, or the hybrid of integrating a stream and a static/ finite data source, requires the identification of elements within each data collection that will serve as the synchronization point

for the integration. For example, when joining weather and vehicle data, the common key may be the vehicle position or the road segment on which the vehicle travels. The data sources (vehicle and weather in this case) are not likely to provide the SEMI ODE with identical keys, however. It would be upon the SEMI ODE to determine a common key for the two sources based on other information provided by the two sources.

In the transportation domain, time and space are key synchronization elements in the integration of two data collections. During the integration of two streams with time and space, the values of the time and space elements in each stream have to be contextually relevant to each other in order for the logic of the consuming application to work properly. The simplest illustration of this concept would be to merge a weather feed with vehicle data. Let's assume that both vehicle and road weather data provide a latitude/longitude and timestamp data. The SEMI ODE must map the lat/long to a common key such as snapping to a road segment for both data sources before joining the two sources on that key.

Function

For the purposes of demonstrating data integration using the ODE's technology platform, we will integrate a weather data feed with EVSD subscription feed. As of this writing, the source of weather data will be a file residing on the SEMI ODE Distributed File System (DFS).

Dependencies

None, as of this writing. In a future enhancement of ODE, data from a published weather feed with adequate latency and spatially relevant to the SE-MI testbed in Farmington Hills should be used instead of the current file-based integration.

CVRIA Application Object

- ODE Data Integration

3.3.4. Validator (Data Valuation)

Purpose

The purpose of the validator is to ensure that the data being processed and sent downstream to SEMI ODE consumers meets minimum quality standards. Minimum quality can be achieved by a two-step process. The first is the application of logic capable of detecting a data anomaly and the second is deciding what processing should occur after detection of the anomaly.

Detection of invalid data can be accomplished in 4 distinct steps for any of the aforementioned CV messages.

Step #1 – Does the data arrive in an uncorrupted state?

If a message cannot be decoded using software adhering to the ASN.1 standard, validation cannot proceed since the data transmission may have corrupted the data or there may have

been an error in the original encoding of the data onboard the vehicle (or other source). The latter might be the case, for example, if the software used for encoding and decoding are not interoperable such as if a different schema was used to encode the data, will not be decodable by the ODE. When this validation fails, error logging will capture the date, time and the contents of the corrupted message.

Step #2 - Can the data elements of any given data record be validated in isolation of other data records in the data stream?

A majority of data elements in the ASN.1 definition of the EVSD are such narrowly defined in the schema definition that cannot possibly be invalid if the ASN.1 runtime has been able to decode the message. For example, an EVSD data element that identifies if ABS has been engaged or not can be either true or false. It cannot ever be in between because a single bit in the computer memory can only be a 0 or a 1. Therefore, if ODE receives a valid ASN.1 message and can decode the message, the ABS status will always reflect a valid status. That leaves only those data elements that have a wide range of values that additional validation is warranted. Detecting erroneous values of a single instance of EVSD data element can be a problematic proposition. Some data types, such as a heading, have a clearly identified upper and lower boundaries that facilitates the detection of data anomalies. Other data, such as speed, are more problematic in terms of applying error detection rules. Speed readings in excess of 200 MPH, for example, can be considered a clear case of invalid speed depending on the situation, but it would be not as clear when the speed is reported at 110 MPH..

Step #3 – Can the data elements of a data record be validated based on the analysis of a range of records in the spatial and temporal vicinity of the record under valuation?

The SEMI ODE will be able to do contextual analysis of the data records and use the results to validate individual data records. For example, vehicle speeds that fall outside the range of plus or minus X standard deviations away from the mean of other vehicles traveling on the same road segment and in the same direction can be flagged as invalid. The value of X is to be defined but will be a configurable parameter. Contextual validation of data records is not within the scope of the current release of SEMI ODE software.

Step #4 - Does the received information match the data request?

Once successfully decoded, the final validation step is to ensure that the data source is providing what was actually requested by the application. The key aspect of this validation is the spatial context of the subscription, and, in the case of a query from the data warehouse, that the time range is also as requested. As such, the query parameters can be used as filters to exclude any data that does not fall within the request criteria. Therefore, filtering the data based on spatial and temporal context is **not** considered an error. Instead, spatial and temporal filters will remove any data records that do not belong to the received data request.

Once an anomaly is detected (Steps 2 and 3 only), subsequent processing of the anomaly is the next phase of the validation. In all cases, we will log detected anomalies to an appropriate log file. The SEMI ODE will not attempt to correct data element values. Instead, the value is

propagated to the client application but flagged as invalid with the criteria that failed the validation.

Function

Given the above discussion, the initial approach to anomaly detection will be to demonstrate that the technical solution is capable of detecting data anomalies by the analysis of data streams in real time.

All data sources for which a subscription is placed to generate a stream, the validation component of the respective pipeline will select specific data elements to perform a range check. In the EVSD, for example, we will validate the direction and speed. Each of these will be compared against a customizable table of values stored in a database. For example, the vehicle heading thresholds for validation purposes will be set to 0 - 359.9875 degrees (values 0-28800 in the raw ASN.1 message representing multiples of 0.0125 degrees as defined by DSRC ASN.1 schema definition). The metadata sent downstream may contain a sequence such as "violations":[{"fieldName":"heading","validMin":0.0,"validMax":359.9875}] to indicate that this particular instance exceeded the maximum threshold of 359.9875 degrees. If all data element values fall in their valid ranges, the data will lack the *violations* data element and will be streamed without any special indication that any value was invalid. Completion of the Field Test Deliverable may help more clearly identify other/additional validation routines as a function of emulated applications.

As of this writing, the list of fields and range of valid values used for validation are presented in table below.

Table 4 - Vehicle Data Validation Ranges

| fieldName | minValue | maxValue | Unit |
|------------|----------|----------|------------------|
| accelLong | -20 | 20 | m/s ² |
| accelLat | -20 | 20 | m/s ² |
| accelVert | -3.4 | 1.54 | G |
| accelYaw | -327.67 | 327.67 | deg/sec |
| heading | 0 | 359.9875 | deg |
| speed | 0 | 163.8 | m/s |
| sizeLength | 0 | 16383 | cm |
| sizeWidth | 0 | 1023 | cm |
| latitude | -90.0 | 90.0 | deg |
| longitude | -180.0 | 180.0 | deg |
| elevation | -409.5 | 6143.9 | M |
| year | 1970 | 9999 | |
| month | 1 | 12 | |

| | | | |
|----------------|-------|--------|---------|
| day | 1 | 31 | |
| hour | 0 | 23 | |
| minute | 0 | 59 | |
| second | 0 | 62.0 | |
| weatherAirPres | 580.0 | 1090.0 | mbar |
| weatherAirTemp | -40.0 | 151.0 | Celsius |

Dependencies

- Sufficient test data volumes that might be required to enable the detection validation

CVRIA Application Object

- ODE Data Valuation

3.3.5. Sanitizer

Purpose

The purpose of the sanitizing component is to maintain the privacy of vehicle operators by anonymizing the information in the SEMI ODE Vehicle Data. Loss of anonymity can occur if the identification of the vehicle can be known by VIN or other unique identifier which can then be linked to the owner record, assuming such secondary data sources are available. Once vehicle ownership is established in relation to the Vehicle Data, the assumption that the owner and the operator are one and the same can be used to compromise anonymity.

In an effort to align the current SDD with current thinking regarding sanitization and overall privacy policy related to CV data, the “Connected Vehicle Data Privacy Assessment - Geotrack De-Identification Algorithm” (2/8/15), which was prepared by Oak Ridge Labs has been reviewed. The excerpt below, which appears on page 15 (Section 2.2.1) of the document, is worth quoting to begin a discussion on sanitization as described in the document and its applicability to the ODE:

“All the trip files that are presented to the de-identification algorithm are defined by a specification, or schema. The schema specifies the field meaning, order, form or type, and the number of fields; good schemas define how an empty field should be interpreted. The file level specification may indicate how trips are aggregated, e.g., *each trip is ordered by timestamp and trips are sequentially intact (non-interleaved)*.

“Inconsistencies in file-level and field-level schemas have been accommodated in the following ways:

- Several of the large trip files provided in the 60-day sample of the Safety Pilot were not trip ordered or time-ordered. In other words, a single complete trip is not listed sequentially in the large file; sometimes trips are interleaved. Furthermore, some trip geo points were not in time sequential order. To accommodate this inconsistency, large trip files must be sorted prior to

consumption by the de-identification algorithm. We use general tools to perform this sort (see Appendix C); the algorithm does not sort trips prior to processing.”

The above excerpt implies that the privacy study analysis presupposes a finite and contiguous collection of rows within a larger data set, where an individual trip can be explicitly identified based on a trip identifier and a properly sequenced collection of data points related to a vehicle’s trajectory/trip. Once created, the file containing the data from a discrete trip can then be cleansed to improve privacy using the sanitization approaches identified in the Oak Ridge study.

In this particular case, with the SEMI ODE receiving real-time EVSD from the SDC, we have neither a data element that uniquely and consistently identifies vehicle’s owner or the driver, nor any notion of where the vehicle is destined to travel.

1. The EVSD data from the SDC contains two identifiers but neither are unique per vehicle for an extended period of time. The two IDs are the “groupId” and “tempId. The “groupId” is defined for a fleet of vehicles belonging to an organization. Privacy-By-Design principles deployed in the US DOT Data Clearinghouse and Data Warehouse dictate that the tempId be changed frequently over the course of the vehicle travel. **Therefore, there is no standalone single data element or collection of elements within a single EVSD record that can be used to uniquely identify the owner or the driver of the vehicle.**
2. The SEMI ODE distributes data to its subscribers on a FIFO (First in First Out) basis as it is received by upstream sources such as the SE-MI clearinghouse and warehouse. In order to sanitize and eliminate records that could help establish a trajectory, the SEMI ODE must delay transmitting the records, establish a trajectory and if the trajectory violates individual privacy, the records can be eliminated from the subscription. **The algorithm to perform such contextual analysis is not within the scope of the current SEMI ODE implementation.**

Therefore, the Oak Ridge algorithms are not applicable to the real-time streaming data currently available to the SEMI ODE. Instead, a subset of the Oak Ridge algorithms has been chosen to demonstrate a very basic sanitization capability of the SEMI ODE.

Function

As there is currently no defined privacy policy to address the unique aspects of streamed data as it passes through the ODE, a modified subset of the approach defined in the Oak Ridge study will be implemented to demonstrate the ability of the SEMI ODE to sanitize data. That approach, called Excluded Area Detection, is identified as one of four Critical Interval Detectors in Section 3.4 of the study. The concept allows the user of the Oak Ridge developed software to preselect spatial boxes that, once defined, will allow the software to purge all trip path data points within that box. Since the Oak Ridge implementation applies to an identified trip path, which is not the case for data streamed by the ODE, we will purge all data within the exclusion areas for all

vehicles appearing inside the area, if and only if the vehicle speed is within a pre-configured speed range for that area AND if the vehicle does NOT belong to any fleet of vehicles, as defined by their groupId, pre-configured for the exclusion area.

Dependencies

- Availability of predefined Exclusion Areas and other parameters associated with those areas such as Group ID's and a Speed range within which vehicle records are excluded.

Preconditions

The Collector component must be operational and able to produce the initial processing of the data stream(s).

CVRIA Application Object

- ODE Data Sanitization

3.3.6. Aggregator

Purpose

To calculate aggregated data as specified by the subscription derived from the data stream. Task 3 White Paper defines Data Aggregation as “the automated creation of composite or summary information from more granular data. In the case of the ODE, the intent is to perform aggregation in near-real time on various elements of connected vehicle data, most notably on speed and position of vehicles on a given segment of roadway or in a geographical area”.

Function

The Aggregator calculates aggregated data for any requested spatial area as defined by the Region request parameter or within a temporal window(s). A Region is defined by the latitude-longitude of the NW corner plus the latitude-longitude of the SE corner. The Aggregator uses the list of Regions stored in the ASIM Cache for which aggregated data is required. The initial implementation of the SEMI ODE Aggregator calculates the aggregates for specified/requested road segments as defined by the Aggregate data request. A road segment is defined by the application/requester using an identifier, latitude/longitude of the starting position (node) and latitude/longitude of the ending point. The Aggregate data request will contain a collection of road segments in a “polyline”. Here's a sample Aggregate Data request including a polyline definition:

```
{  
  
  "dataSource": "SDC",  
  
  "nwLat": "42.538046177132884",
```

```
"nwLon": "-83.48201842041014",
"seLat": "42.30538799068131",
"seLon": "-82.83863157958984",
"polyline": {
  "segments": [
    {
      "id": "LarnedShelby-LarnedGriswold",
      "startPoint": {
        "latitude": "42.328468",
        "longitude": "-83.04747"
      },
      "endPoint": {
        "latitude": "42.329067",
        "longitude": "-83.046086"
      }
    },
    {
      "id": "LarnedGriswold-LarnedRandolph",
      "prevSegment": "LarnedShelby-LarnedGriswold",
      "endPoint": {
        "latitude": "42.33053",
        "longitude": "-83.042701"
      }
    },
    {
      "id": "TelegraphBingham-TelegraphW12Mile",
      "startPoint": {
```

```
        "latitude": "42.51555",
        "longitude": "-83.28530"
    },
    "endPoint": {
        "latitude": "42.50113",
        "longitude": "-83.28470"
    }
},
{
    "id": "TelegraphW12Mile-TelegraphCivicCtr",
    "prevSegment": "TelegraphBingham-TelegraphW12Mile",
    "endPoint": {
        "latitude": "42.47965",
        "longitude": "-83.28444"
    }
}
]
}
}
```

Dependencies

- Collector

CVRIA Application Object

- ODE Data Aggregation

3.3.7. Distributor

Purpose

Distribution is the last step in the SEMI ODE's servicing of a subscription. Its goal is to ensure that endpoints having active subscriptions to the SEMI ODE data are sent their respective data feed(s) in a timely and complete manner.

- Determines which data consumption endpoints have placed requests for the data.
- Writes the data directly to those applications' open connections to ODE.

Dependencies

- Request Manager

CVRIA Application Object

- ODE Data Distribution

3.4. Data Consumption

Current data volumes from the SE-MI clearinghouse are constrained by the fact that the current fleet of Connected Vehicles is limited to 6 test vehicles. This creates the potential situation where the ODE's upstream interface to the clearinghouse is unable to generate sufficient EVSD data volumes to demonstrate the scalability of the SEMI ODE interface.

In keeping with the theme of demonstrating the technical capability of the SEMI ODE in the current flux of R&D, we will adopt three distinct approaches to demonstrate the capabilities of data ingest and data propagation. The first is a suite of functional tests in which we will verify the range of possible scenarios by manually creating data and simulating low volumes of ingested data. This is important in order to have known and controlled inputs to determine that the SEMI ODE is functioning as designed.

The next approach to the testing of SEMI ODE Data Consumption is to have the SEMI ODE ingest higher flow rates of data. This is designed to show the scalability of the SEMI ODE and determine at what data rates the initially deployed configuration will cease to operate.

Lastly, the SEMI ODE will be tested with four selected applications that will serve to emulate Field test conditions. (The details of this test are described in the SEMI ODE Field Test Plan document). The goal of the Field Test is to simulate an end-to-end test starting with test vehicles executing an orchestrated driving loop in the Detroit and Novi test beds and terminating in the consumption of vehicle, intersection and/or advisory data to four emulated applications.

3.5. Logging and Monitoring

System health and operational insights will be learned through the deployment of centralized monitoring and log analysis tools. These tools will be deployed to collect, monitor, and if necessary, alert the operations team of any issues that SEMI ODE services may be experiencing. The SEMI ODE will utilize a monitoring tool to perform active and passive monitoring, log rotation, its modular plugin framework, as well as sending alert notifications to designated recipients. To complement this monitoring capability, a data visualization tool will be used to gather and present monitoring data. Log analysis tools are still to be evaluated at this time based on ease of use, maturity and the ability to integrate with other SEMI ODE technologies.

Monitoring tools were deployed onto ODE systems to monitor and collect system and application based metrics. Host based monitoring is used to monitor server resources (CPU, disk, RAM, etc.) as well as monitoring the health of various services (running, offline, memory usage, etc.). Application based metrics are collected by leveraging 3rd party or custom built monitoring modules that query the applications for useful data. These data points are recorded over time and sent to a centralized server, which provides an overall view of the system health.

A log analysis tool will be employed in order to collect, index and query event logs to further gain system health and operational insights. Through the deployment of a centralized logging service, SEMI ODE applications will be constructed so that they can transmit event information, whether it be an exception information or regular operating information, to a centralized logging analysis tool which will store the event information in a search index capable data store. For applications and system services that write information to log files, software agents will be configured and deployed as needed to parse the log files and transmit that information to the log analysis tool.

System health and operational insights can be gained through the use of alerts, visualizations and ad-hoc querying of the system metrics and log files. Monitoring alerts can be configured based on defined thresholds for certain metrics. The ability to leverage multiple metrics for alerting will allow the SEMI ODE to create context specific alerts and react accordingly. Data visualizations include the various SEMI ODE system metrics to help identify usage trends and patterns in addition to visualization of the health of the various SEMI ODE servers and applications. Pre-defined and ad-hoc data visualization can be created as well. Lastly log data can be searched through a log search indexing tool which can help correlate log events from the various systems to aid in identifying system issues. Log data, if structured properly, will also be visualized and correlated with other metrics.

3.6. Space Estimates

There is currently no requirement to permanently store data that is streamed to the SEMI ODE since this data can already be found elsewhere within the CVRIA in locations such as the Research Data Exchange (RDE) and the SE-MI SDPC. SEMI ODE does not store, archive or cache data beyond its processing requirements. The subscriber will not be provided a throttling mechanism to limit the rate of data flow.

Depending on how SEMI ODE processing evolves in the future however, it may be helpful to store certain data that is uniquely generated by SEMI ODE transformations. There may also be the short-term need to store some additional data to disk to facilitate the development and test efforts, but it would not have a specific purpose vis-à-vis any deliverable currently in scope. Storage will also be used to store logs of various events during the operation of the ODE, but, again, these can be purged as needed or transferred to an offline (cold) storage system which is the lowest cost storage option.

Therefore, the only space relevant to the operation of the SEMI ODE is memory space. Since all the data handled by the SEMI ODE is transient data, the SEMI ODE is only required to provide memory space for the duration of time the data lives within the ODE. The maximum amount of time the SEMI ODE keeps data in its distributed memory is determined by the duration of the Aggregation sliding window. Assuming SEMI ODE receiving and processing data from 10 vehicles at 10 messages per second for 60 seconds, SEMI ODE will need to store 6000 messages in memory. Assuming 2 kilobytes (KB) of data per message, SEMI ODE must store 12 MB of data in memory.

3.7. Impact to Existing Components

At this point in time, the SEMI ODE does not impact any existing components, as it is a new system. s

This page intentionally left blank

CHAPTER 4. REQUIREMENTS TRACEABILITY

Table 5 maps requirements to design components and vice-versa. The Req # column refers to the corresponding paragraph of the SEMI ODE Task 4 - Concept of Operations document.

Table 5 - Requirements Traceability Matrix

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| 5.2.2.1 | The SEMI ODE shall allow a system administrator (super user) to set the parameters for data validation (e.g., ranges of validity for ingested data) | | | | | | | | X | | | |
| 5.2.2.2 | The ODE shall allow a client application to register for data access. | | X | | | | | | | | | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|---------|----------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| 5.2.2.3 | The SEMI ODE shall require all client applications to authenticate using a user name and password combination | X | X | | | | | | | | | |
| 5.2.2.4 | The SEMI ODE may require all client applications to connect over HTTPS | X | | | | | | | | | | |
| 5.2.2.5 | 5.2.2.5 The ODE shall authenticate with all its data sources using whichever authentication the data sources require | | | | | | | | X | | | |
| 5.2.2.6 | The SEMI ODE shall interface with the Situation Data Clearinghouse (SDC) in the SE-MI testbed | X | | X | X | | | | | | | |
| 5.2.2.7 | The SEMI ODE shall interface with the | X | | X | X | | | | | | | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|-----------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| | Situation Data Warehouse (SDW) in the SE-MI testbed | | | | | | | | | | | |
| 5.2.2.8 | The SEMI ODE shall ingest streaming Vehicle Situation Data (EVSD) from the SDC | X | | X | X | X | | | | | | |
| 5.2.2.9 | The SEMI ODE shall query the SDW or SDPC for archived Vehicle Situation Data (EVSD) | X | | X | X | | | | | | | |
| 5.2.2.10 | The SEMI ODE shall ingest streaming Intersection Situation Data (ISD) from the SDC | X | | X | X | X | | | | | | |
| 5.2.2.11 | The SEMI ODE shall query the SDW or SDPC for archived Intersection Situation Data (ISD) | X | | X | X | | | | | | | |
| 5.2.2.12 | The SEMI ODE shall ingest streaming Traveler | X | | X | X | X | | | | | | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|----------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| | Advisory Situation Data (TSD) from the SDW | | | | | | | | | | | |
| 5.2.2.13 | The SEMI ODE shall query the SDW for archived Traveler Advisory Situation Data (TSD) | X | | X | X | | | | | | | |
| 5.2.2.14 | The SEMI ODE shall allow specification of road network data. | | | | | | | | | X | | |
| 5.2.2.15 | The SEMI ODE shall ingest simple weather data (e.g., atmospheric temperature, barometric pressure) | X | | X | X | X | | | | | | |
| 5.2.2.16 | The SEMI ODE shall allow applications to deposit EVSD and ISD data to SDC and TSD data to the SDW | X | X | | | | | | | | | |
| 5.2.2.17 | The SEMI ODE shall publish data to consumer | | | | | | X | | | | | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| | applications in the form of JSON payloads | | | | | | | | | | | |
| 5.2.2.18 | The SEMI ODE shall allow consumer applications to specify the geographic area for which they wish to receive data | X | | X | | | | | | | | |
| 5.2.2.19 | The SEMI ODE shall allow consumer applications to provide the following parameters when requesting archived data: data type, geographic region, time interval | X | | X | | | | | | | | |
| 5.2.2.20 | The SEMI ODE shall remove any ingested data that was generated in a geographic location outside a predefined area | | | | | | X | | | | | |
| 5.2.2.21 | The SEMI ODE shall remove any ingested | | | | | | X | | | | | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| | data that was generated outside a predefined temporal window | | | | | | | | | | | |
| 5.2.2.22 | The SEMI ODE shall mark as invalid any ingested data the value of which falls outside a predefined range | | | | | | | | X | | | |
| 5.2.2.23 | The SEMI ODE shall distribute all invalid data to the requesting applications | | | | | | X | | X | | | |
| 5.2.2.24 | The SEMI ODE shall mark invalid data with validation parameters used to validate the data which indicates why the data was deemed invalid | | | | | | | | X | | | |
| 5.2.2.25 | The SEMI ODE shall mark invalid data with a specific and human-readable description of why the data was | | | | | | | | X | | | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|-------------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| | deemed invalid (e.g., "violations":[{"fieldName": "accelVert", "validMin": -3.4, "validMax": 1.54}]) | | | | | | | | | | | |
| 5.2.2.26 | The SEMI ODE shall integrate EVSD data with temperature data from at least one meteorological sources | | | | | | | | | X | | |
| 5.2.2.27 | The SEMI ODE shall perform the integration for all subscriptions | | | | | | | | | X | | |
| 5.2.2.28 | The SEMI ODE shall integrate data that was marked as invalid during validation | | | | | | | | | X | | |
| 5.2.2.29 | The SEMI ODE shall remove all data that falls within predefined exclusion areas, unless covered by requirement 5.2.2.30 | | | | | | | | | | X | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| 5.2.2.30 | The SEMI ODE shall not remove data for vehicles with predefined vehicle group ID, because these vehicles are considered exempt from sanitization (there is no expectation of privacy for these vehicles) | | | | | | | | | | X | |
| 5.2.2.31 | The SEMI ODE shall allow consumer applications to request aggregate data | X | | X | X | | | | | | | |
| 5.2.2.32 | The SEMI ODE shall calculate aggregate data within a configurable pre-defined time window (aggregation window). | | | | | | | | | | | X |
| 5.2.2.33 | The SEMI ODE shall disseminate the aggregate data to the requesting applications at | | | | | | | | | | | X |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| | the end of the aggregation window. | | | | | | | | | | | |
| 5.2.2.34 | The SEMI ODE shall calculate the count, average, minimum and maximum vehicle speeds for all supplied road segments, within a configurable sliding time window. | | | | | | | | | | | X |
| 5.2.2.35 | The SEMI ODE shall not use data that was marked as invalid during validation as input for aggregation | | | | | | | | X | | | X |
| 5.2.2.36 | The SEMI ODE shall perform validation for all data sources where applicable | | | | | | | | X | | | |
| 5.2.2.37 | The SEMI ODE shall perform sanitization for | | | | | | | | | | X | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| | all data sources where applicable | | | | | | | | | | | |
| 5.2.2.38 | The SEMI ODE shall perform aggregation as required by the SEMI ODE configuration | | | | | | | | | | | X |
| 5.2.2.39 | The SEMI ODE shall monitor the volume of data ingested over time. | | | | | X | | X | | | | |
| 5.2.2.40 | The ODE may monitor the volume of data distributed to each consumer application over time. | | | | | | X | X | | | | |
| 5.2.2.41 | The SEMI ODE shall measure and monitor the amount of delay introduced by its processing (i.e., the time elapsed between ingestion and dissemination). | | | | | X | X | X | | | | |

| Req. # | Requirement | Service API | IAM | Request Manager | Connector | Collector | Distributor | Logging and Monitoring | Validator | Integrator | Sanitizer | Aggregator |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----|-----------------|-----------|-----------|-------------|------------------------|-----------|------------|-----------|------------|
| 5.2.2.42 | The SEMI ODE shall capture all errors related to its operation, including but not limited to data ingestion errors, data processing errors, and data dissemination errors. | X | X | X | X | X | X | X | X | X | X | X |
| | | | | | | | | | | | | |

This page intentionally left blank

CHAPTER 5. DATABASE

5.1. Overview

As discussed earlier, the SEMI ODE does not persist the data that is streamed through it, except as a function of low level processes that are employed by the various components to ensure fault tolerance.

The SEMI ODE currently employs a predefined database in the Liferay® server to capture essential user profile information, including authentication and access privileges. This data is currently managed in a schema deployed inside of a MySQL database, which is the default Liferay RDBMS. In order to allow applications to save predefined subscription and query profiles in greater volumes in the future, a secondary database can be designed to store and maintain that information.

Active Subscriptions In-Memory Cache is another form of database that will be needed at some point in order to maintain a scalable and efficient non-persistent metadata of active subscriptions. REDIS is one such open-source framework that can be utilized to deploy this capability when needed.

5.2. Entity Relationship Diagram

The SEMI ODE currently has no utility for a relational database but it can be envisioned to need one in the future if client application require the management of pre-defined subscriptions and queries (see 5.1 above).

5.3. Data Dictionary

The Active Subscriptions In-Memory Cache is the only type of data dictionary defined and envisioned at this point. This data dictionary will be a collection of key-value pair where the key is a subscription ID and the value is a data structure consisting of subscription request, message queue names and other metadata associated with each active subscription.

This page intentionally left blank

CHAPTER 6. INTERFACE SPECIFICATIONS

6.1. Outbound Requests placed by SEMI ODE to DDS

In order to accomplish its basic role brokering information on behalf of client applications, the SEMI ODE will accept requests through either the RESTful API or the SEMI ODE Streaming API. Once received, the SEMI ODE will determine whether the request can be fulfilled by the SDC, SDW, or SDPC. The SEMI ODE will construct subscription or query requests according to the following...

6.1.1. Enhanced Vehicle Situation Data (EVSD) Subscription Request

```
SUBSCRIBE:{
  vsmType: <VsmType>,
  systemSubName: "SDC 2.2",
  dialogID: "154",
  nwLat2: <floating point latitude in degrees>,
  nwLon2: <floating point longitude in degrees>,
  seLat2: <floating point latitude in degrees>,
  seLon2: <floating point longitude in degrees>,
  resultEncoding: <"hex" | "base64">
}
```

```
VsmType ::= OCTET STRING (SIZE(1))
  -- fund (1), "00000001", VehSitRcd that only contains the
  fundamental data elements
  -- vehstat (2), "00000010", VehSitRcd that contains the
  VehicleStatus Data Frame
  -- weather (4), "00000100", VehSitRcd that contains Weather
  Data
  -- env (8), "00001000", VehSitRcd that contains Environmental
  data
  -- elveh (16) "00010000", VehSitRcd that contains Electric
  Vehicle data
```


6.1.2. Intersection Situation Data (ISD) Subscription Request

```
SUBSCRIBE:{
  vsmType: <VsmType>,
  systemSubName: "SDC 2.2",
  dialogID: "162",
  nwLat2: <floating point latitude in degrees>,
  nwLon2: <floating point longitude in degrees>,
  seLat2: <floating point latitude in degrees>,
  seLon2: <floating point longitude in degrees>,
  resultEncoding: <"hex" | "base64">
}
```

6.1.3. Traveler Situation Data (TSD) Query

```
QUERY:{
  systemSubName: <"SDW 2.2" | "SDPC 2.2">,
  dialogID: "156",
  startDate: <yyyy-MM-ddThh:mm:ss>,
  startDateOperator: <"GT" | "GTE" | "LT" | "LTE">,
  endDate: <yyyy-MM-ddThh:mm:ss>,
  endDateOperator: <"GT" | "GTE" | "LT" | "LTE">,
  nwLat: <floating point latitude in degrees>,
  nwLon: <floating point longitude in degrees>,
  seLat: <floating point latitude in degrees>,
  seLon: <floating point longitude in degrees>,
  orderByField: <field name>,
  orderByOrder: <1 (Ascending) | -1 (Descending)>,
  skip: <integer>,
  limit: <integer>,
  resultEncoding: <"hex" | "base64" | "full">
}
```

6.2. Inbound Data Streams from the DDS to the ODE

The SEMI ODE can receive 3 types of data messages from the DDS:

1. Enhanced Vehicle Situation Data (EVSD) Message
2. Intersection Situation Data (ISD) Message which consists of
 - a. SPaT Message
 - b. MAP Message
3. Traveler Situation Data (TSD) Message

Sections below describe each of these messages in detail.

6.2.1. Enhanced Vehicle Situation Data (EVSD) Messages

Vehicles equipped with onboard equipment (OBE) generate enhanced EVSD messages (similar to the Basic Safety Message (BSM)). These EVSD messages are sent to the data clearinghouse over Internet Protocol (IP); the Road Side Unit (RSU) serves as a gateway to pass along the messages to the clearinghouse. EVSDs are encoded with a schema as shown below

Table 6 – Enhanced Vehicle Situation Data

| ASN.1 TAG | Data Elements | Value | Comments |
|-----------|-------------------------|--------------------------|-------------------------------------------------------------------------------------------------------------------|
| | DialogID | 0x009A | --Enhanced Vehicle Situation Data/ Enhanced Vehicle Situation Data Deposit |
| | seqID | 0x05 | --data |
| | groupID | <OCTET STRING (SIZE(4))> | -- Generated by the vehicle. Unique ID used to identify an organization that this service\data is associated with |
| | requestID | <OCTET STRING (SIZE(4))> | --from the ServiceRequest |
| | vsmType | <OCTET STRING (SIZE(1))> | 1=fundamental, 2=vehicle status, 4=weather, 8=environmental, 16=electric vehicle |
| | VehSitRcd bundle | | |
| 30 | VehSitRcd 1 | | |
| 80 | TempID | DSRC.TemporaryID | --from BSM |

| | | | |
|----|---------------------|---------------------------|------------|
| a1 | time | DSRC.DDateTime | |
| 80 | year | DSRC.DYear | |
| 81 | month | DSRC.DMonth | |
| 82 | day | DSRC.DDay | |
| 83 | hour | DSRC.DHour | |
| 84 | minute | DSRC.DMinute | |
| 85 | second | DSRC.DSecond | |
| a2 | position | DSRC.Position3D | |
| 80 | lat | DSRC.Latitude | |
| 81 | Long | DSRC.Longitude | |
| 82 | elev | DSRC.Elevation | |
| a3 | fundamental | | |
| 80 | speed | DSRC.TransmissionAndSpeed | --from BSM |
| 81 | heading | DSRC.Heading | --from BSM |
| 82 | steeringWheelAngle | DSRC.SteeringWheelAngle | --from BSM |
| 83 | accelerationSet4Way | DSRC.AccelerationSet4Way | --from BSM |
| 84 | brakeSystemStatus | DSRC.BrakeSystemStatus | --from BSM |

| | | | |
|----|------------------------|---------------------|--------------------------------------------------------------------|
| a5 | vehicleSize | DSRC.VehicleSize | |
| 80 | Width | DSRC.Width | |
| 81 | Length | DSRC.Length | |
| 86 | event | ENUMERATED OPTIONAL | -- used to indicate Start and Stop Events; 1 = "Start", 2 = "Stop" |
| a4 | VehicleSituationStatus | OPTIONAL | |
| a5 | Weather | OPTIONAL | |
| a6 | Environmental | OPTIONAL | |
| a7 | ElectricVeh | OPTIONAL | |
| 30 | VehSitRcd 2 | | |
| 30 | VehSitRcd 3 | | |
| | : | | |
| 30 | VehSitRcd 10 | | |
| | crc | DSRC.MsgCRC | --2 Bytes |

See the bulleted list below for a few notes regarding and:

- The numbers and letters in the ASN.1 TAG column are ASN.1 tags, that indicate either: a “primitive” value if the tag starts with “8,” a “constructed” value if the tag starts with “A,” and a “composite” value (meaning one or more may be repeated) if the tag is equal to “30.”
- The number at the end of some data elements represents the iteration for that element; that is, the number n at the end of an element name means that element may repeat, and it is the “nth” iteration. Gaps in iterations are represented by ellipses.
- The rows with text appearing in the “Value” column list the J2735 Standard defined object name.
- The last element is listed as “CRC” (cyclic redundancy check) to serve as an internal validity confirmation that bits were not lost in transmission.

6.2.2. Intersection Situational Data (ISD) Messages

Road Side Equipment (RSE) devices collect and transmit Intersection Situation Data (ISD) for deposit to DDS. ISD communicates MAP and Signal Phase and Timing (SPaT) information. MAP information communicates an intersection's location (latitude and longitude), elevation, and geometric features such as approaches and lane configuration. SPaT data communicates the current state of the intersection's signal indication(s). Transmitted ISDs are received by the DDS and as such are available via subscription to the clearinghouse.

As for the contents of intersection situational data Table 5 (below) shows the data bundle layout. The ISD is designed to support both SPaT and MAP data and the presence of each within the ISD message is distinguished by its respective ASN.1 tag

Table 7 – Intersection Situation Data

| ASN. 1 TAG | Data Elements | Value | Comments |
|------------------|---------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | dialogID | 0x00A2 | --Intersection Situation Data Deposit |
| | seqID | 0x05 | --Data |
| | groupID | <OCTET STRING (SIZE(4))> | --from ServiceRequest |
| | requestID | <OCTET STRING (SIZE(4))> | --randomly generated by the initiating device. Enables generating device and warehouse to manage multiple "Data" sessions. Note that this is different from the requestID in ServiceRequest |
| | bundleNumber | INTEGER(1..32767) | -- cumulative count of bundles sent this session |

| | | | | |
|----|---------------|------------|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | timeToLive | | ENUMERATED OPTIONAL | --indicates when the PDU should be purged from the SDW |
| | | | | --1 minute (0), 30 minutes (1), 1 day (2), 1 week (3), 1 month (4), 1 year (5) |
| | serviceRegion | | | --geographic region of the intersection |
| a0 | | nwCorner | DSRC.Position3D | --Northwest corner of the intersection |
| 80 | | lat | DSRC.Latitude | |
| 81 | | long | DSRC.Longitude | |
| 82 | | evel | DSRC.Elevation | |
| a1 | | seCorner | DSRC.Position3D | --Southeast corner of the intersection |
| 80 | | lat | DSRC.Latitude | |
| 81 | | long | DSRC.Longitude | |
| 82 | | evel | DSRC.Elevation | |
| | isdRcd | | | |
| a0 | | Map | DSRC.MapData | --SAE J2735 Encoded Map message describing the subject intersection. All spatRecords are relative to this single Map |
| a1 | | spatRecord | | -- a single SAE J2735 Signal Phase and Timing (IntersectionState) message pertaining to the subject intersection and relative to the included Map with a time stamp as to when the |

| | | | | |
|----|--|---------------|------------------------|------------------------------------------------------------------------|
| | | | | IntersectionState was captured |
| a0 | | timeStamp | DSRC.DDateTime | --time at which the IntersectionState was generated |
| 80 | | year | DSRC.DYear | |
| 81 | | month | DSRC.DMonth | |
| 82 | | day | DSRC.DDay | |
| 83 | | hour | DSRC.DHour | |
| 84 | | minute | DSRC.DMinute | |
| 85 | | second | DSRC.DSecond | |
| a1 | | intersections | DSRC.IntersectionState | --SAE J2735 Intersection State containing the Phase State at timestamp |

6.2.3. Traveler Situation Data (TSD) Messages

Traveler Situation Data are obtained from the US DOT SDW and US DOT SDPC and not the US DOT SDC. To access TSD from the DDS, a query must be submitted to the SDW or SDPC. Table 6 (below) shows the data bundle layout for Traveler Situation Data.

| ASN.1 Tag | Data Elements | Value | Comments |
|-----------|---------------|--------|-----------------------------------|
| | dialogID | 0x009C | --Traveler Situation Data Deposit |
| | seqID | 0x05 | --Data |

| | | | | |
|----|---------------|----------|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | groupID | | <OCTET STRING (SIZE(4))> | --from the ServiceRequest |
| | requestID | | <OCTET STRING (SIZE(4))> | --randomly generated by the initiating device. Enables generating device and warehouse to manage multiple "Data" sessions. Note that this is different from the requestID in ServiceRequest |
| | recordID | | OPTIONAL <OCTET STRING (SIZE(4))> | --used to by the provider to overwrite exiting records\PDU's in the SDW |
| | timeToLive | | ENUMERATED OPTIONAL | --indicates when the PDU should be purged from the SDW |
| | | | | --1 minute (0), 30 minutes (1), 1 day (2), 1 week (3), 1 month (4), 1 year (5) |
| | serviceRegion | | | |
| a0 | | nwCorner | DSRC.Position3D | --Northwest corner of the data geographic region |
| 80 | | lat | DSRC.Latitude | |
| 81 | | long | DSRC.Longitude | |
| 82 | | evel | DSRC.Elevation | |
| a1 | | seCorner | DSRC.Position3D | --Southeast corner of the data geographic region |
| 80 | | lat | DSRC.Latitude | |

| | | | | |
|----|-----------------|------------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 81 | | long | DSRC.Longitude | |
| 82 | | Elev | DSRC.Elevation | |
| | advisoryDetails | | | |
| 80 | | TSDmID | <OCTET STRING (SIZE(4))> | --assignend by the DPC to manage TSDMs in the RHSDW. DPC can add additional TSDMs or overwrite existing TSDMs in the RHSDW based on the TSDmID |
| 81 | | TSDmType | ENUMERATED | --SPaT aggregate (0), MAP (1), TIM (2), EV (3), etc. |
| | | | | --Only 1 Map message is to be broadcast by an RSU at any given time. May not want to allow DPCs to provide Map Messages to the warehouse....to be discussed... |
| 82 | | distributionType | <OCTET STRING (SIZE(1))> | --indicates how the TSDM is distributed; 1=rsu, 2=ip, |
| 83 | | startTime | DSRC.DFullTime OPTIONAL | --Time at which the TSDM should start to be available to vehicles (e.g. RSU starts broadcasting the TSDM). If not provided, startTime is the system time at which the TSDM is persisted into the warehouse |

| | | | | |
|----|--|-----------------|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 84 | | stopTime | DSRC.DFullTime OPTIONAL | --Time at which the TSDM should stop being available to vehicles (e.g. RSU stops broadcasting the TSDM). If not provided, stopTime is 30 minutes after the startTime |
| 85 | | advisoryMessage | <OCTET STRING (SIZE(0..1400))> | --1 encoded Traveler Information, or similar, Message for distribution. Only the TSDM is distributed |

Table 8 – Traveler Situation Data

6.2.4. 3rd party/ancillary data providers

The SEMI ODE can in the future interface with weather data services and road network data providers such as OpenStreets or Google Earth to obtain road segment information and integrate that data into the system. These functions are not within the scope of the initial SEMI ODE implementation.

6.3. Outbound Data Streams from SEMI ODE to Client Applications

The sections below describe the schema for the outbound data messages distributed to client applications. The schema is described in a YAML-like (Yet Another Markup Language) format with indentation representing the nesting of various components of the data structure. The output of the SEMI ODE will be in JSON (Java Simple Object Notation) format so all of the data elements are represented in plain ASCII text with angle brackets used to identify the expected type to which the value can be converted. The type can be used by the client application to bind the message with a type-safe programming language, the latter being a language(s) that avoid run-time errors induced by data types that are incorrectly specified in relation the data that is actually processed by the code (at run time).

For a detailed description of the SEMI ODE API specification, please refer to the SEMI ODE API Documentation.

6.3.1. Vehicle Data

The following is a sample Vehicle Data record propagated by the ODE. Full specification of the SEMI ODE API will be provided in a separate document.

```
{
  "serialId": "10817812-036b-4d7b-867b-ae0bc62a2b3e.0",
  "receivedAt": "2015-07-22T19:21:16.413+0000",
  "groupId": "4130008F",
  "accelLong": 0.34,
  "accelVert": 0,
  "accelYaw": 8.42,
  "heading": 650.95,
  "speed": 8.12,
  "sizeLength": 500,
  "sizeWidth": 200,
  "latitude": 42.3296667,
  "longitude": -83.044539,
  "elevation": 156.9,
  "tempId": "C4290123",
  "year": 2015,
  "month": 5,
  "day": 13,
  "hour": 15,
  "minute": 52,
  "second": 45.5,
  "dateTime": "2015-06-13T19:52:45.500+0000"
  "roadSegment": "ABCD"
}
```

6.3.2. Intersection Data

```
IntersectionData:
  groupId
  serviceRegion
  nwCorner
  position3D:
    elevation: <float in meters, Precision: in decimeters
(10 cm)>
    latitude: <float in degrees range: -90..+90>
    longitude: <float in degrees range: -90..+90>
  seCorner
  position3D:
```

```
        elevation: <float in meters, Precision: in decimeters
(10 cm)>
        latitude: <float in degrees range: -90..+90>
        longitude: <float in degrees range: -90..+90>
mapData
  Name: <string>
  layerType: <>
  layerID: <>
  intersections:
    -
      name
      id
      refPoint
      refInterNum
      orientation
      laneWidth
      type
      approaches
        -
          refPoint: <Position3D>
          laneWidth
          approach
          egress
        - ...
  preemptZones
    -
      name
      pValue
      data
        laneSet:[laneNumber,...]
        zones
          -
            enclosed:[laneNumber,...]
            laneWidth
            nodeList
          - ...
    - ...
  priorityZones
    -
      name
      pValue
      data
        laneSet:[laneNumber,...]
        zones
          -
```

```
        enclosed:[laneNumber,...]
        laneWidth
        nodeList
    - ...
- ...
- ...
dataParameters
  processMethod
  processAgency
  lastCheckedDate
  geiodUsed
spatData
  timeStamp
  year
  month
  day
  hour
  minute
  second
  name
  id
  status
  lanesCnt
  states
  priority
  preempt
```

6.3.3. SPaT Data

```
IntersectionData:
  groupID
  serviceRegion
  nwCorner
    position3D:
      elevation: <float in meters, Precision: in decimeters
(10 cm)>
      latitude: <float in degrees range: -90..+90>
      longitude: <float in degrees range: -90..+90>
  seCorner
    position3D:
      elevation: <float in meters, Precision: in decimeters
(10 cm)>
      latitude: <float in degrees range: -90..+90>
      longitude: <float in degrees range: -90..+90>
```

```
spatData
  timeStamp
    year
    month
    day
    hour
    minute
    second
  name
  id
  status
  lanesCnt
  states
  priority
  preempt
```

6.3.4. Map Data

```
IntersectionData:
  groupID
  serviceRegion
  nwCorner
    position3D:
      elevation: <float in meters, Precision: in decimeters
(10 cm)>
      latitude: <float in degrees range: -90..+90>
      longitude: <float in degrees range: -90..+90>
  seCorner
    position3D:
      elevation: <float in meters, Precision: in decimeters
(10 cm)>
      latitude: <float in degrees range: -90..+90>
      longitude: <float in degrees range: -90..+90>
  mapData
    Name: <string>
    layerType: <>
    layerID: <>
    intersections:
      -
        name
        id
        refPoint
        refInterNum
        orientation
```

```
    laneWidth
    type
    approaches
      -
        refPoint: <Position3D>
        laneWidth
        approach
        egress
      - ...
    preemptZones
      -
        name
        pValue
        data
          laneSet:[laneNumber,...]
          zones
            -
              enclosed:[laneNumber,...]
              laneWidth
              nodeList
            - ...
        - ...
    priorityZones
      -
        name
        pValue
        data
          laneSet:[laneNumber,...]
          zones
            -
              enclosed:[laneNumber,...]
              laneWidth
              nodeList
            - ...
        - ...
    - ...
  dataParameters
    processMethod
    processAgency
    lastCheckedDate
    geiodUsed
```

6.3.5. Advisory Data

```
AdvisoryData:
  groupID
  serviceRegion
  nwCorner
  position3D:
    elevation: <float in meters, Precision: in decimeters
(10 cm)>
    latitude: <float in degrees range: -90..+90>
    longitude: <float in degrees range: -90..+90>
  seCorner
  position3D:
    elevation: <float in meters, Precision: in decimeters
(10 cm)>
    latitude: <float in degrees range: -90..+90>
    longitude: <float in degrees range: -90..+90>
  TSDmDetails
  TSDmID
  TSDmType: <"spatAggregate" | "map" | "tim" | "ev">
  distType: <"none" | "rsu" | "ip">
  startTime
  stopTime
  advisoryMessage
```

6.3.6. Aggregate Data

```
AggregateData
  serviceRegion
  nwCorner
  position3D:
    elevation: <float in meters, Precision: in decimeters
(10 cm)>
    latitude: <float in degrees range: -90..+90>
    longitude: <float in degrees range: -90..+90>
  seCorner
  position3D:
    elevation: <float in meters, Precision: in decimeters
(10 cm)>
    latitude: <float in degrees range: -90..+90>
    longitude: <float in degrees range: -90..+90>
  region:
    numVehicles: <integer>
    avgSpeed: <>
```



```
    minSpeed: <>
    maxSpeed: <>
    medianSpeed: <>
roadSegment:
    numVehicles: <integer>
    avgSpeed: <>
    minSpeed: <>
    maxSpeed: <>
    medianSpeed: <>
nb:
    numVehicles: <integer>
    avgSpeed: <>
    minSpeed: <>
    maxSpeed: <>
    medianSpeed: <>
neb:
    numVehicles: <integer>
    avgSpeed: <>
    minSpeed: <>
    maxSpeed: <>
    medianSpeed: <>
eb:
    numVehicles: <integer>
    avgSpeed: <>
    minSpeed: <>
    maxSpeed: <>
    medianSpeed: <>
seb:
    numVehicles: <integer>
    avgSpeed: <>
    minSpeed: <>
    maxSpeed: <>
    medianSpeed: <>
sb:
    numVehicles: <integer>
    avgSpeed: <>
    minSpeed: <>
    maxSpeed: <>
    medianSpeed: <>
swb:
    numVehicles: <integer>
    avgSpeed: <>
    minSpeed: <>
    maxSpeed: <>
    medianSpeed: <>
```

```
wb:
  numVehicles: <integer>
  avgSpeed: <>
  minSpeed: <>
  maxSpeed: <>
  medianSpeed: <>
nwb:
  numVehicles: <integer>
  avgSpeed: <>
  minSpeed: <>
  maxSpeed: <>
  medianSpeed: <>
```

6.3.7. User

This interface schema encapsulates the characteristics of an SEMI ODE User. The attributes listed correspond to the user identifier that is designed to be captured within the database.

```
User:
  email: <email>
  pw: <password>
  firstName: <string>
  middleName: <string>
  lastName: <string>
  organization: <string>
  phoneNum: <phone>
```

This page intentionally left blank

APPENDIX A: CONNECTED VEHICLE REFERENCE IMPLEMENTATION ARCHITECTURE (CVRIA) PHYSICAL OBJECT DEFINITIONS (FOR REFERENCE ONLY)⁹

Definitions in this section support the objects depicted in CVRIA diagrams depicted in Section 2.

6.4. Archived Data Center

The "Archived Data Center" collects, archives, manages, and distributes data generated from ITS sources for use in transportation administration, policy evaluation, safety, planning, performance monitoring, program assessment, operations, and research applications. The data received is formatted and tagged with attributes that define the data source, conditions under which it was collected, data transformations, and other information (i.e. meta data) necessary to interpret the data. The archive can fuse ITS generated data with data from non-ITS sources and other archives to generate information products utilizing data from multiple functional areas, modes, and jurisdictions. The archive prepares data products that can serve as inputs to federal, state, and local data reporting systems. The "Archived Data Center" may be implemented in many different ways. It may reside within an operational center and provide focused access to a particular agency's data archives. Alternatively, it may operate as a distinct center that collects data from multiple agencies and sources and provides a general data warehouse service for a region.

6.5. Center

This general physical object is used to model general functions and core capabilities that may be supported by any center included in CVRIA.

⁹ These definitions are copied verbatim from CVRIA documentation for reference only in order to provide context to the CVRIA physical architecture diagram presented in Figure 1 - System Architecture Physical View – Level 0.

6.6. Cooperative ITS Credentials Management System

The "Cooperative ITS Credentials Management System" is a high-level aggregate representation of the interconnected systems that enable trusted communications between mobile devices and other mobile devices, roadside devices, and centers and protect the data they handle from unauthorized access. This physical object will be implemented as an interconnected system of support applications that enable the secure distribution, use, and revocation of trust credentials.

6.7. Data Distribution System

The "Data Distribution System" collects, processes, and distributes connected vehicle data, connecting data producers with data consumers and facilitating data exchange in the Connected Vehicle Environment.

6.8. ITS Roadway Equipment

"ITS Roadway Equipment" represents the ITS equipment that is distributed on and along the roadway that monitors and controls traffic and monitors and manages the roadway itself. In CVRIA, this physical object represents all of the other ITS field equipment that interfaces with and supports the Connected Vehicle Roadside Equipment (RSE). This physical object includes traffic detectors, environmental sensors, traffic signals, highway advisory radios, dynamic message signs, CCTV cameras and video image processing systems, grade crossing warning systems, and ramp metering systems. Lane management systems and barrier systems that control access to transportation infrastructure such as roadways, bridges and tunnels are also included. This object also provides environmental monitoring including sensors that measure road conditions, surface weather, and vehicle emissions. Work zone systems including work zone surveillance, traffic control, driver warning, and work crew safety systems are also included.

6.9. Object Registration and Discovery Service

The "Object Registration and Discovery Service" represents one or more center-based applications that provide registration and lookup services necessary to allow objects to locate other objects operating within the Connected Vehicle Environment. These registration and discovery services are support services that enable other applications to provide transportation services.

6.10. Personal Information Device

The "Personal Information Device" provides the capability for travelers to receive formatted traveler information wherever they are. Capabilities include traveler information, trip planning, and route guidance. It provides travelers with the capability to receive route planning from the

infrastructure at home, at work, or en route using personal devices that may be linked with connected vehicle on-board equipment.

6.11. Roadside Equipment

"Roadside Equipment " (RSE) represents the Connected Vehicle roadside devices that are used to send messages to, and receive messages from, nearby vehicles using Dedicated Short Range Communications (DSRC). Communications with adjacent ITS Roadway Equipment (see the separate object) and back office centers that monitor and control the RSUs are also supported. This device operates from a fixed position and may be permanently deployed or a portable device that is located temporarily in the vicinity of a traffic incident, road construction, or a special event. It includes a processor, data storage, and communications capabilities that support secure communications with passing vehicles, other roadside equipment, and centers that provide back office support.

6.12. Service Monitor System

The "Service Monitor System" represents one or more center-based applications that provide monitoring, management and control services necessary to other applications and/or devices operating within the Connected Vehicle Environment. These support services enable other applications to provide transportation services.

6.13. Transportation Information Center

The "Transportation Information Center" collects, processes, stores, and disseminates transportation information to system operators and the traveling public. The physical object can play several different roles in an integrated ITS. In one role, the TIC provides a data collection, fusing, and repackaging function, collecting information from transportation system operators and redistributing this information to other system operators in the region and other TICs. In this information redistribution role, the TIC provides a bridge between the various transportation systems that produce the information and the other TICs and their subscribers that use the information. The second role of a TIC is focused on delivery of traveler information to subscribers and the public at large. Information provided includes basic advisories, traffic and road conditions, transit schedule information, yellow pages information, ride matching information, and parking information. The TIC is commonly implemented as a website or a web-based application service, but it represents any traveler information distribution service including systems that broadcast digital transportation data (e.g., satellite radio networks) and systems that support distribution through a connected vehicle network.

6.14. Vehicle On-Board Equipment (OBE)

The Vehicle On-Board Equipment (OBE) provides the vehicle-based processing, storage, and communications functions necessary to support connected vehicle operations. The radio(s) supporting V2V and V2I communications are a key component of the Vehicle OBE. This communication platform is augmented with processing and data storage capability that supports the connected vehicle applications.

Four different types of implementations are represented by the Vehicle OBE:

1. **Vehicle Awareness Device** – This is an aftermarket electronic device, installed in a vehicle without connection to vehicle systems, that is only capable of sending the basic safety message over short range communications. Vehicle awareness devices do not generate warnings.
2. **Aftermarket Device** – This is an aftermarket electronic device, installed in a vehicle, and capable of sending and receiving messages over a wireless communications link. The self-contained device includes GPS, runs connected vehicle applications, and includes an integrated driver interface that issues audible or visual warnings, alerts, and guidance to the driver of the vehicle.
3. **Retrofit Device** – This is an electronic device installed in vehicles by an authorized service provider, at a service facility after the vehicle has completed the manufacturing process (retrofit). This type of device provides two-way communications and is connected to a vehicle databus to integrate the device with other on-board systems. Depending on implementation, the device may include an integrated driver interface and GPS or integrate with modules on the vehicle bus that provide these services. Depending on implementation, it may support all of the connected vehicle applications identified in CVRIA.
4. **Integrated System** – This is a system of one or more electronic devices integrated into vehicles during the vehicle production. The Integrated System is connected to proprietary data busses to share information with other on-board systems. The Integrated System may include many control modules and may be configured to support all of the connected vehicle applications identified in CVRIA.

In retrofit and integrated implementations, the Vehicle OBE interfaces to other on-board systems through a vehicle bus (e.g., CAN). Represented in CVRIA as the Vehicle Platform, this interface provides access to on-board sensors, monitoring and control systems, and information systems that support connected vehicle applications. The vehicle bus may also be the source for GPS location and time and the access point for the vehicle's driver-vehicle interface. Self-contained devices include an integrated GPS and driver interface that supports direct visual, audible, or haptic interaction with the driver.

In CVRIA, the Vehicle OBE includes the functions and interfaces that support connected vehicle applications for passenger cars and trucks. Many of these applications (e.g., V2V Safety applications) apply to all vehicle types including personal automobiles, commercial vehicles, emergency vehicles, transit vehicles, and maintenance vehicles. In CVRIA, the Vehicle OBE is used to model the common interfaces and functions that apply to all of these vehicle types.

This page intentionally left blank

APPENDIX B: CVRIA SYSTEM ARCHITECTURE APPLICATION OBJECT DEFINITIONS (FOR REFERENCE ONLY)¹⁰

6.15. Operational Data Environment

The "Operational Data Environment" collects, processes, stores, and disseminates transportation information to system operators and the traveling public. The physical object can play several different roles in an integrated ITS. In one role, the SEMI ODE provides a data collection, aggregation, integration and repackaging function, collecting information from transportation various data sources and redistributing this information to other system operators in the region and TICs. In this information redistribution role, the SEMI ODE provides a bridge between the various transportation systems that produce the information and subscribers that use the information. Information provided includes Enhanced Vehicle Situation Data, SPaT, MAP and advisories, traffic and road conditions, transit schedule information, yellow pages information, ride-matching information, and parking information. The SEMI ODE is implemented as a web-based application service.

6.16. Archived Data Center

The "Archived Data Center" collects, archives, manages, and distributes data generated from ITS sources for use in transportation administration, policy evaluation, safety, planning, performance monitoring, program assessment, operations, and research applications. The data received is formatted and tagged with attributes that define the data source, conditions under which it was collected, data transformations, and other information (i.e. meta data) necessary to interpret the data. The archive can fuse ITS generated data with data from non-ITS sources and other archives to generate information products utilizing data from multiple functional areas, modes, and jurisdictions. The archive prepares data products that can serve as inputs to federal, state, and local data reporting systems. The "Archived Data Center" may be implemented in many different ways. It may reside within an operational center and provide focused access to a particular agency's data archives. Alternatively, it may operate as a distinct center that collects data from multiple agencies and sources and provides a general data warehouse service for a region.

¹⁰ These definitions are copied verbatim from CVRIA documentation for reference only in order to provide context to the CVRIA physical architecture diagram presented in Figure 1.

6.17. Archived Data User Systems

The "Archived Data User Systems" represents the systems users employ to access archived data. The general interface provided allows a broad range of users (e.g. planners, researchers, analysts, operators) and their systems (e.g. databases, models, analytical tools, user interface devices) to acquire data and analyses results from the archive.

6.18. Authorizing Center

The "Authorizing Center" provides the functionality needed to enable data exchange between and among mobile and fixed transportation users. Its primary mission is to enable safety, mobility and environmental communications-based applications for both mobile and non-mobile users. The Core System includes those enabling technologies and services that will provide the foundation for application transactions. The Core may also provide data distribution and network support services depending on the needs of the Core deployment. The Core System may be implemented as an autonomous center or as a set of supporting services that are collocated within another center.

6.19. Center

This general physical object is used to model general functions and core capabilities that may be supported by any center included in CVRIA.

6.20. Transportation Information Center

The "Transportation Information Center" collects, processes, stores, and disseminates transportation information to system operators and the traveling public. The physical object can play several different roles in an integrated ITS. In one role, the TIC provides a data collection, fusing, and repackaging function, collecting information from transportation system operators and redistributing this information to other system operators in the region and other TICs. In this information redistribution role, the TIC provides a bridge between the various transportation systems that produce the information and the other TICs and their subscribers that use the information. The second role of a TIC is focused on delivery of traveler information to subscribers and the public at large. Information provided includes basic advisories, traffic and road conditions, transit schedule information, yellow pages information, ride-matching information, and parking information. The TIC is commonly implemented as a website or a web-based application service, but it represents any traveler information distribution service including systems that broadcast digital transportation data (e.g., satellite radio networks) and systems that support distribution through a connected vehicle network.

U.S. Department of Transportation
Federal Highway Administration
1200 New Jersey Avenue, SE
Washington, DC 20590

Toll-Free “Help Line” 866-367-7487
www.its.dot.gov

FHWA-JPO-15-227



U.S. Department of Transportation