Design Document for: *Vehicle to Everything (V2X) Data Exchange Platform*

Approval Date: ***March 2024***

Table of Contents

List of Tables

List of Figures

1 Overview

This document provides the detailed design description of the Vehicle-to-Everything (V2X) Data Exchange Platform (DEP). This document will be updated throughout the project and reflect on-going design decisions. It may be considered current or forward-looking as of the time of publication.

1.1 Identification

Project Name: Vehicle-to-Everything (V2X) Data Exchange Platform (DEP)
Financial Project Identification: BEB93
Federal Aid Project Number: N/A

1.2 Purpose and Intended Audience

The purpose of this document is to provide a detailed design description of the V2X DEP with an emphasis on 3rd party interfaces. The detailed design description includes approaches and rationale concerning V2X DEP 3rd party interfaces and their supporting configuration and operation. The currently supported 3rd party interfaces are enumerated and detailed in this document in Sections 7 and 9.

The intended audience for this document includes:

- Technical management of participating agencies
- System developers
- Technical managers providing oversight of the project

If using this document as a development or code maintenance reference, such individuals should have, at a minimum, experience with git, cloud services, and general-purpose programming (preferably .NET/C#-based). They should also have traffic management and connected vehicle domain knowledge.

1.3 Document Overview

This document provides a detailed design description of the V2X DEP. This includes data ingestion processes, internal data streaming and processing pipelines, data dissemination capabilities, and internal security considerations. The focus is on external interfaces and considerations for the V2X DEP, while internal aspects are covered in the V2X DEP Baseline Architecture Document.

1.4 High-Level System Overview

V2X DEP will collect and store data generated from FDOTs Connected and Autonomous Vehicle (CAV) deployments across the state, enabling FDOT to consolidate data from numerous disparate transportation systems/projects across the state each with their own purposes, features, capabilities, constraints, and technical architectures and make it available to other FDOT offices and stakeholders that do not currently have access to utilize for strategic planning, equitable access, improved safety and mobility, and other functions to achieve objectives outlined in the CAV Business Plan.

The V2X DEP will collect, store, and provide analytics on a wide range of data from CAV deployments and other systems, including but not limited to SunGuide®, Data Integration and Video Aggregation System (DIVAS), SunStore, and the Florida Advanced Traveler Information System. Data from various 3rd party systems will also be consumed and used, including data from vehicle Original Equipment Manufacturers (OEM), HERE, weather systems, and 3rd party traffic data systems, among others. DEP will also make data available to vehicles by providing alerts and other data to SunGuide and other external systems (such as Regional Integrated Transportation Information System (RITIS)) that will then push the data to roadside units (RSU) or vehicles. The platform will support modular expansion to enable adding new data sources

as new data sources are made available, new analytics processes on collected data, and flexible reporting and distribution of information. The DEP does not provide real-time, low-latency alerts (such as in-vehicle forward collision warnings, intersection collision warnings, and others) to vehicles or drivers. These needs still require direct device-to-device communications between applications on field devices (OBUs and/or RSUs). Data shared or otherwise made available from the V2X DEP includes work zone data, incident information, congestion information, among other data with the expectation that information and alerts can be provided to drivers or vehicles in advance of situations. Additional information regarding the components shown in Figure 1 may be found in the V2X DEP Baseline Architecture Document.

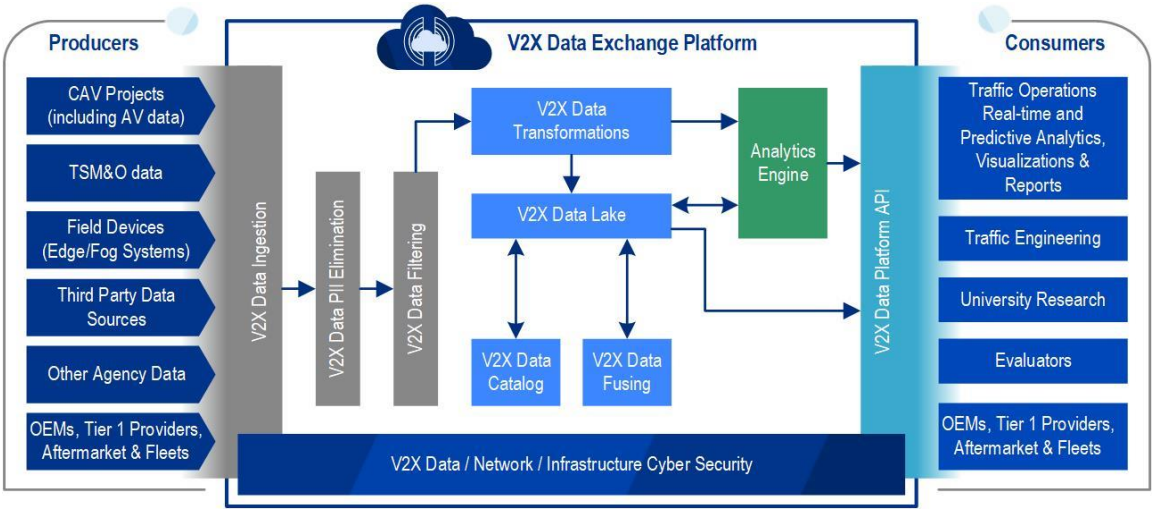


Figure 1. V2X DEP High-level Components

1.5 Stakeholders

Stakeholders for this project include the following: FDOT Districts, FDOT Central Office, Florida toll authorities, first responders, automotive OEMs, commercial fleet operators, equipment suppliers, hardware vendors, data analytics vendors, non-FDOT government agencies, other CAV automotive equipment suppliers, automotive consortia, and the traveling public. Refer to the V2X DEP Concept of Operations and Requirements Document for a comprehensive list of stakeholders.

1.6 Referenced Documentation

The following documents are referenced in or were used in elaborating the design of the system. If a referenced document conflicts with this document, this document shall take precedence. Note that document links will be added as final shared names and locations are determined.

Table 1. References

Document No.	Document Title	Link/Contact Info
1.3	V2X DEP Baseline Architecture Document	v2x-dep- baseline-architecture.docx
3.0	V2X DEP Concept of Operations and Requirements Document	FDOT V2X-DEP ConOps-Reqs v3.docx
4.0	V2X DEP Security Plan	FDOT V2X-DEP Security Plan v4.docx
SOW-012634	AWS Documentation	https://docs.aws.amazon.com/

2 Architecture Rationale

The architecture provides features for ingestion and storage of data, real time data transformation, analysis, and notifications, and query/viewing of historical and current data through a dashboard user interface or programmatic access. The architecture primarily uses a serverless, streaming and data lake approach.

In designing the core data streaming architecture, we posed then answered several questions as a means of analyzing important factors motivating the design. The following are key architecture design features with corresponding rationales:

- **Infrastructure-as-Code (IaC) Support** – To support V2X DEP build out, the architecture implementation must support a simple approach to provisioning an environment. The architecture implementation must support provisioning replicant environments (e.g., Staging and Production). The architecture must be described a declarative manger and versioned to track historic changes. The architecture implementation must be robust (e.g., full featured language support, tolerable to changes, bug free production version, etc.).
- **Scalability** – The architecture must work both with data volumes present on Day 1 as well as data volumes likely to be present many years into the future.
- **Cost Efficiency** – The architecture must scale to accommodate data fluctuation (current and future data needs as well as routine fluctuations in data throughput) along with hosting costs scaled equivalently. The architecture must support estimation of anticipated cost for the selected solution.
- **Development Effort** – The design must be optimized to reduce effort (time & money) required by development team to create a system leveraging streaming data as well as reduce effort required by development team to enhance or modify the system.
- **Flexibility and Extensibility** – The tools chosen to support the architecture design must be supported for years to come. If tooling is discontinued, the effort to transition to an alternative streaming solution should be minimal.

2.1 Engineering Analysis

In evaluating the above parameters, we considered several variants of Kafka (self-hosted Apache Kafka, Confluent Kafka, Amazon Managed Streaming for Apache Kafka (MSK), and Amazon MSK Serverless) and Amazon Kinesis Data Streams (KDS) (On Demand and Provisioned). While each option was able to perform at some level under each parameter, ultimately the best across the board solution for the DEP was determined to be KDS On Demand.

Each solution considered reported to meet any anticipated volume/velocity demand for project's period of performance including Proof of Concept (POC) Phase 1 and Phase 2 (4 years total) (See planned quantities in Sections 7.1 and 7.2 for planned data velocities and volumes) and potentially (excessive uncertainty for distant projections) beyond (optional renewal for up to 3 more years), with varying levels of development effort required to attain necessary scalability (See planned quantities in Sections 7.1 and 7.2 for quantification). Additionally, while it is impossible to predict the future, historic and current performance can be used as a factor for predictions, so each data streaming option considered was expected to remain in support for the period of performance of the project which was assessed by the hosting vendor's characteristics including market capitalization, time since initial release of product, and frequency of updates to the product. IaC support and cost efficiency were where the data streaming options started to diverge. The only offering with stable production ready IaC support was KDS. The other

two offerings have some rudimentary IaC support, but not to the level necessary to design a production grade system. And for cost efficiency, KDS cost is commensurate with other options. For MSK Serverless, the imposed quotas were an issue due to constraints on maximum data throughput. See V2X DEP Baseline Architecture Document, particularly the Baseline Architecture Design section, for more details.

3 Configuration Management

3.1 Infrastructure as Code

Infrastructure as Code (IaC) is defined as managing and provisioning data center infrastructure by way of machine-readable definition files rather than through interactive configuration process. The IaC methodology for provisioning a system provides a process for deploying a full system, including underlying compute infrastructure, in a repeatable manner, getting an equivalent system each time it's deployed. IaC further provides the ability to store in source control everything necessary to re-constitute the system, providing the same traceability for functional changes to infrastructure which has, for decades now, been available for traditional software development. In this way, use case-specific application code can be tracked alongside both the infrastructure required to run it and the configuration required for integration with 3rd party applications.

Although the V2X DEP is a single, statewide deployment to one environment, the cloud infrastructure supports multiple, parallel instantiations in different environments. A cloud environment is defined as a unique combination of cloud provider account and region. Deployment of V2X DEP IaC to multiple environments allows rapid testing and development leveraging a full development environment separate from a full production environment in addition to partial ad hoc environments deployed as needed. With an infrastructure description specified by IaC, it is the IaC tooling that facilitates provisioning and incremental updates to the system so that downtime for upgrades is minimized and deployment automation can be maximized.

The V2X DEP provides support for multiple long-lived simultaneous interface implementations, such as different versions of the SunGuide software. The code base is comprised of several smaller repositories, where code can be packaged and versioned appropriately (see Table 2). This allows for increased flexibility, better traceability and maintaining compatibility for the configuration of multiple software versions using a modularized approach.

For the above reasons, the V2X DEP leverages IaC design throughout. Specifically, the development team will leverage Amazon Web Services (AWS) Cloud Development Kit (CDK) which provides a high-level representation of the infrastructure in a modern software language. AWS CDK supports multiple languages, but V2X DEP will specifically use the .NET/C# variant.

Table 2. Code Repositories

Repository Name	Description
ActiveDX.Core	Kinesis streaming support
ActiveDX.Core.SchemaLibrary	Contains data schema files
ActiveDX.Fargate.Ford	Ford producer
ActiveDX.Fargate.J2735	J2735 producer
ActiveDX.Fargate.SunGuide	SunGuide producer
ActiveDX.Infrastructure	CDK Infrastructure as Code, Flink analytics, Lambda functions
ActiveDX.Lambda.Here	Here producer

ActiveDX.Lambda.Waze	Waze producer
ActiveDX.Lambda.Wzdx	Work zone data exchange feed
J2735	J2735 library allowing for multi-version support
SunGuide	SunGuide library allowing for multi-version support

IaC configuration is source controlled.

Another aspect of IaC and the AWS CDK is the concept of a stack. A stack is a singular unit of code that is deployed together. It is used to create modular cloud infrastructure that is interchangeable and more maintainable. The following table describes the name of the stack, the location in the code base where the source code is located, and a description on what the stack is used for.

Table 3. DEP Code Stacks

Stack Name	Location	Description
AnalyticsStack	ActiveDX.Infrastructure/src/ActiveDX/AnalyticsStack	This stack deploys Kinesis Data Analytics (KDA) stream processing applications. These applications read Kinesis data streams and transform, enrich, join, and sink new data for downstream use. Sinks may include storage in the S3/Parquet data lake or Redshift/PostgreSQL database and/or publication to Kinesis streams or SNS topics.
AuthorizationStack	ActiveDX.Infrastructure/src/ActiveDX/AuthorizationStack	This stack creates the admin API for key management and group permissions.
BucketStack	ActiveDX.Infrastructure/src/ActiveDX/BucketStack	This stack creates the S3 buckets used for storing data in the DEP.
ConfigNestedStack	ActiveDX.Infrastructure/src/ActiveDX/ConfigNestedStack	This stack is used to set up the AWS config of all the supported resources. Which will be used to capture all the metrics of the stacks and maintain the inventory of the AWS account. The purpose it is serving to us is vulnerability check and to check the CloudFormation (CF) stack drift tracking
CvdfStack	ActiveDX.Infrastructure/src/ActiveDX/CvdfStack	This stack creates an external API for the Connected Vehicle Data Framework (CVDF)

Stack Name	Location	Description
DataCatalogStack	ActiveDX.Infrastructure/src/ActiveDX/DataCatalogStack	This stack creates a Glue Data Catalog (GDC) database to store metadata for the S3/parquet data lake. This stack initializes GDC tables with information about data locations, schemas, and formats. Date-time partitions are added to tables by stream processing pipelines to meet data availability SLAs. Once data partitions are added to the GDC, they are searchable by Athena and Redshift query engines.
ExternalApiStack	ActiveDX.Infrastructure/src/ActiveDX/ExternalApiStack	This stack deploys a Swagger UI for externally used REST APIs as a static S3 public website.
FindingsStack	ActiveDX.Infrastructure/src/ActiveDX/FindingsStack	Deploys SNS topics and other resources required to publish findings from monitored resources.
GuardDutyNestedStack	ActiveDX.Infrastructure/src/ActiveDX/GuardDutyNestedStack	Implements GuardDuty, a smart intrusion detection system (IDS), for the architecture and send logs to S3 and SNS Topic using a Lambda. The SNS Topic is configured for email and SMS text message targets from the system's contacts.
MetricsStack	ActiveDX.Infrastructure/src/ActiveDX/MetricsStack	This stack deploys resources to collect AWS Cloudwatch metrics for tracking of DEP performance metrics.
NetworkStack	ActiveDX.Infrastructure/src/ActiveDX/NetworkStack	The NetworkStack contains configuration of the main VPC for hosting ActiveDX platform resources.
NotificationsNestedStack	ActiveDX.Infrastructure/src/ActiveDX/NotificationsNestedStack	The Notifications Nested Stack registers contacts for notifications through SNS.
ProduceHereStack	ActiveDX.Infrastructure/src/ActiveDX/ProduceHereStack	The ProduceHereStack ingests data from the HERE Traffic API data through a Lambda function triggered by a scheduled Event Rule.
ProduceMapUploadStack	ActiveDX.Infrastructure/src/ActiveDX/ProduceMapUploadStack	Creates a lambda function to ingest Map data manually uploaded to a S3 Bucket.
ProducerStack	ActiveDX.Infrastructure/src/ActiveDX/ProducerStack	The ProducerStack contains the ingestion logic. This includes J2735, SunGuide, and Ford.
ProduceWazeStack	ActiveDX.Infrastructure/src/ActiveDX/ProduceWazeStack	The ProduceWazeStack ingests data from the WAZE crash RoadwayEvent API data through a Lambda function triggered by a scheduled Event Rule.

Stack Name	Location	Description
RedshiftStack	ActiveDX.Infrastructure/src/ActiveDX/RedshiftStack	The RedshiftStack creates a distributed data warehouse in the main VPC.
RedshiftSchemaStack	ActiveDX.Infrastructure/src/ActiveDX/RedshiftSchemaStack	The RedshiftStack creates a distributed data warehouse in the main VPC. The stack creates a Redshift computing cluster to host the distributed data warehouse, comprised of relational PostgreSQL databases with external schemas and tables for access to data stored in S3 using Redshift Spectrum. This stack creates the cluster and a role used by Redshift Spectrum to access S3 data. S3 data can be added to the data warehouse via a Glue data catalog or directly from S3 as external tables. This stack also creates database users for authenticated application connections, and exposes some helper methods for granting permissions to applications roles.
RegistryStack	ActiveDX.Infrastructure/src/ActiveDX/RegistryStack	The RegistryStack creates ECR repositories (an AWS name for a particular Docker Registry image path). One repository represents an image and may contain one or more tagged versions. There are ECR repositories for the Core (used to generate Avro schema-derived classes) and the Producers (used to build and run specific 3rd party ingestion APIs for the DEP).
SchemaRegistryStack	ActiveDX.Infrastructure/src/ActiveDX/SchemaRegistryStack	Creates the streaming Schema Registry and Schema Constructs which should be adhered to by Producers which PutRecords on the streams and internal Consumers (Kinesis Data Analytics/Flink/ETL) which Source records from the streams.
SecretStack	ActiveDX.Infrastructure/src/ActiveDX/SecretStack	Generates AWS Secrets Manager secrets for credentials managed by deployment owners.
StreamStack	ActiveDX.Infrastructure/src/ActiveDX/StreamStack	This stack provides the integration components between various Stacks. For example, the ProducerStream connects the ProducerStack to the AnalysisStack.
TrafficFindingsNestedStack	ActiveDX.Infrastructure/src/ActiveDX/TrafficFindingsNestedStack	Deploys a SNS topic and Lambda function to publish findings from monitored traffic streams.
UserInterfaceStack	ActiveDX.Infrastructure/src/ActiveDX/UserInterfaceStack	An Amazon LightSail service/container-based user interface.
UserStack	ActiveDX.Infrastructure/src/ActiveDX/UserStack	A stack which provisions users and their permissions for the platform.

Stack Name	Location	Description
WorkZoneDataExchangeStack	ActiveDX.Infrastructure/src/ActiveDX/WorkZoneDataExchangeStack	The WorkzoneDataExchangeStack sets up required resources for dissemination of workzone data through the Workzone Data Exchange. Sets up the S3 bucket that hosts the feed as well as various properties of the feed.

3.2 Deployment Configuration

The CDK variant of IaC supports specification of deployment parameters through the AWS CDK Runtime Context (https://docs.aws.amazon.com/cdk/v2/guide/context.html). These parameters are found in files stored within the FDOT.V2xDep code repository and control deployment conditions such as minimum or maximum scale, deployment regions, etc. The Context contains parameters specified by the cloud provider (like deployment region) as well as custom parameters specific to the V2X DEP. At deployment, the Context may be set from a variety of methods, but the V2X DEP primarily uses a CDK Context JSON file. The V2X DEP additionally provides a CDK Context JSON Schema file which enumerates, defines, and provides examples, ranges, and defaults for the CDK Context parameters. Ahead of deployment time, a particular deployment Context can be validated.

3.3 Deployment Bootstrapping

Other than specifying deployment parameters in the CDK Context, in order to deploy to a cloud provider, there are a minimal sequence of prerequisite one-time bootstrapping steps which must be completed outside the automated deployment via the CDK:

1. Create an account with the cloud provider.
2. Create an initial AWS Identity and Access Management (IAM) User in the account to conduct CDK deployment.
3. Create an S3 Bucket to contain the CDK state.

Following these manual prerequisite steps, the CDK deployment may be executed. These bootstrapping steps are described in greater detail both by the cloud provider and in the V2X DEP repository (https://docs.aws.amazon.com/cdk/v2/guide/bootstrapping.html).

3.4 Deployment Updates

As the system development continues, it will become necessary to redeploy modified application and infrastructure code (See also Section 4.2). Ease of deployment and quality of deployment tooling is essential to redeploy frequently to support short feedback loops between project developers and stakeholders. CDK facilitates the update process through modular design. Units of deployment called Stacks can be redeployed as needed individually. Stacks contain all the resources (direct and indirect) required for a particular service to execute. If there are dependent Stacks, CDK will automate their update also. Stacks are composed of Constructs or AWS-specific resources (e.g. an AWS S3 bucket or AWS Lambda Function). If only a subset of Constructs within a Stack is modified, the CDK will detect and apply changes to only the impacted subset of Constructs.

4 Development Environment

The OS for development environment is Windows, but WSL 2 (Ubuntu 20.04 LTS) is used where UNIX-specific tooling is required. The following tools are used to develop the platform:

- Docker: The platform leverages several containerized applications, and docker is necessary to build, run, test them.
- Git: The platform version control system is git-based.
- AWS CLI: The AWS CLI is used in conjunction with CDK IaC to synthesize and deploy account resource descriptions.
- AWS CDK: The CDK is the tooling used to specify the IaC
- Garden: In order to support multiple interface version concurrently, the garden tool is used to interact with a multi-repo folder structure where each folder is independently versioned
- Visual Studio 2022: The primary .NET Integrated Development Environment (IDE) for the platform application builds.
- Visual Studio Code: The primary IDE for languages other than .NET.
- Java/Maven: This project uses Java for the Kinesis Data Analytics applications and the Ford Producer using OpenJDK.
- Upack: The universal package manager CLI tool used for package types other than nuget and maven.

5 DevOps

The Atlassian family of products are being used to support the CI/CD process in conjunction with CDK. Detailed information on the process may be found in the source code repository (<source_code_top_level_file_path>/docs/CICD Information.md). Note that the top-level file path location of the source code repository is an FDOT responsibility, hence the usage of a relative path. The sections that follow provide a general overview.

5.1 Continuous Integration (CI) / Automated Testing

The Atlassian tool BitBucket is used for versioning code history, as well as new modifications. BitBucket is integrated with Bamboo builds for each DEP module. Prior to merging new code into the main branches in BitBucket, a successful Bamboo build and all passing automated unit tests are required.

SonarQube has been integrated into the development process. This software performs automated code quality inspections, such as identification of bugs, code smells, test coverage, etc. Various parameter thresholds are configured which prevent code from being merged until they have met or exceeded all requirements. For example, unit testing requirements are defined for a minimum code coverage percentage. Prior to merging code into version control, developers must meet the default SonarQube ruleset, called the “SonarWay.”

Once all build, code quality, and testing requirements have been met the code changes are merged into the main branch.

5.2 Continuous Delivery/Deployment (CD)

On a cadence in sync with the conclusion of a sprint’s worth of development, the final revisions from DEP modules are tagged and the corresponding build artefacts are promoted for deployment. Then using the Atlassian Bamboo Deploy automation, the code is deployed into the DEP Staging account.

6 Data Communication Plan

The V2X DEP can communicate with a variety of current and future data producers and consumers. Data producers, which provide data for ingestion to the V2X DEP, including both software systems which may aggregate or relay data as well as device drivers for data directly from the primary data source. Data

consumers from the V2X DEP can include automated software systems in addition to external users including researchers, FDOT staff, vehicle fleet operators, and the public. The V2X DEP is designed to scale as needed to ingest an unlimited number of distinct data sources. The platform is also designed to scale up in near real time to meet daily, weekly, or other time period data usage demands throughput peak usage and scale when usage reduces. The entire data pipeline from ingestion to consumption leverages serverless resources with built in high availability and scalability outlined in further detail in the next sections.

6.1 Data Ingestion

The data ingestion process is designed to be flexible enough both to accommodate all known data sources today as well as extended to support future data sources.

Each distinct data source has a corresponding data driver implementing the authentication, authorization, and subscription protocols necessary to receive data. Depending on the nature of the data source, one of three general approaches to data collection will be implemented:

- **Persistent Connection** – For systems like SunGuide that require a persistent Transmission Control Protocol/Internet Protocol (TCP/IP) connection, the V2X DEP must first provision a Docker container to manage and maintain the connection from the DEP to the source system. The container will automatically establish a connection and reconnect as needed.
- **Polled Connection** – Where 3rd party systems must be polled periodically to receive updated datasets, asynchronous functions will be scheduled to perform these polls. Connections to the target dataset may use a variety of connection types, each determined based on the specific capabilities of the data source. Examples of these data sources may be existing FTP repositories that the DEP will have to periodically check for updates. REST or SOAP request and response methods will be used for this type of connection.
- **Pushed Connection** – Where external systems will initiate data transfer into the system, ingestion will be implemented on a case-by-case basis based on the source system's capabilities. These are expected to typically be over UDP, but may be FTP, SCP, or other transfer mechanisms.

For each of the above data ingestion methods, once initial ingestion is complete, resulting data is put into the data processing pipeline where it can be filtered and transformed as necessary to a common and consistent format that can be used for near real-time analysis and long-term storage. For example, Basic Safety Messages (BSMs) may be sent to the V2X DEP by a RSU in the field directly to a UDP port in the V2X DEP. In most cases, to capture the identity of the sending RSU, a forwarding process will relay BSMs and prepend the identity. Regardless of how the BSM is received by the DEP, it will be transformed into the same internal format for use by subsequent analytics processes.

6.1.1 CAV Data Ingestion

Due to the sheer volume and velocity of data coming into the platform by way of RSU special consideration is given to these datasets (e.g., SAE J2735 or Ford). Many traditional ITS data sources, such as field device status (online/offline), are updated relatively slowly (it may take 15-30 seconds or more to detect and report the status). Similarly, data such as what comes from SunGuide Transportation Sensor Subsystem (TSS), is typically aggregated at 30 second (or longer) intervals. CAV data on the other hand, loses much of its utility if it is not used in real time (or near real time which is within 5 seconds withing the DEP for the purposes of this document), with all available raw data provided to the utilities or processes that are performing analytics on it. While not expected to be as bandwidth intensive as video data from cameras, CAV data has similar needs in how it should be processed and shared between end points.

Unlike some other datasets considered, CAV data may be pre-processed on site (e.g. in the TMC, client network, at the edge, etc.), prior to relaying back to the DEP. This type of pre-processing is, by its very nature, outside of the purview of the V2X DEP and is mentioned here for transparency. The pre-processing may in some instances perform some rudimentary de-duplication of data in advance of relay, such as removing duplicate messages when the originating vehicle (OBU) is in range of multiple roadside units at once or augmenting the data by adding metadata describing the data source that might otherwise not be available once it reaches the V2X DEP. Pre-processing of data at edge locations (or anywhere prior to data being made available to the DEP), may provide benefit of offloading certain data cleaning and deduplication, but that very fact may also restrict certain use cases that require the DEP to have all available (including duplicate) data. Discussion and decisions regarding this will be brought up during ideation meetings, weighing the options of what can be accomplished and associated costs or assumptions.

On-site CAV data handling varies by district based on available infrastructure, however there are various systems and components that are deployed and operational or under development that could provide building blocks to facilitate providing this CAV data to the V2X DEP.

There are multiple options which can facilitate transporting CAV from the district to the V2X DEP as outlined below, all of which can be further adapted to accommodate specific architectural needs of the target district or operating agency. SunGuide, which is readily available and present in most districts already, has a driver receiving CAV messages and, for this reason, it is also shown in each diagram below. The figures and options presented below are simplified views of the potential data flows from field RSUs to the V2X DEP. Most, if not all, deployments have multiple networks and firewalls in play that will need to be identified and considered. Discussions will take place with each participating agency to determine the most appropriate path forward and how to adapt the solution to the specific network and deployment. Note that with each of these approaches, this is specific to how data from the field makes its way to the DEP. None of these approaches dictate how data from the DEP will be provided back out to consumers or field devices. That discussion is covered in the Data Consumers section, 6.2.

- **Approach A – Network Relay:** In the same way that RSUs in the field are currently configured to relay traffic back to the Traffic Management Center (TMC) to be processed by SunGuide or a vendor provided system, the RSUs would be configured to additionally relay similarly directly to the DEP. This configuration requires that a firewall hole be made both on the TMC side allowing traffic from each RSU to travel over the site-to-site VPN connection between the TMC and the DEP and also that the DEP firewall rules allow the connection from the RSU, shown in Figure 2.. Many RSUs support sending data like this to multiple destinations, but not all do currently. For RSUs that do not support this ability, this approach requires a minimal network attached relay device or software process which can accept data on one port and distribute that same packet to multiple destinations (i.e. SunGuide and V2X DEP).

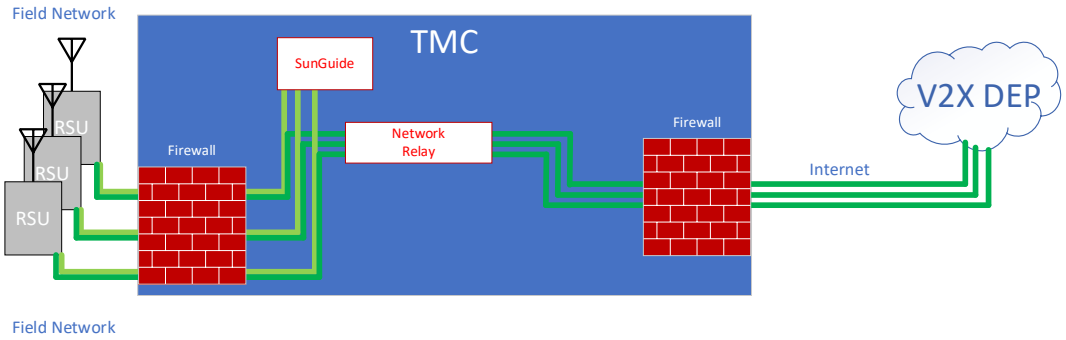


Figure 2. Field CV Data Flow Approach A

- **Approach B – Aggregation Service:** As with Approach A, this approach involves RSUs being configured to relay messages to two distinct endpoints, however the new endpoint in this approach is a new software system (identified as 'Aggregation Service') running in the TMC which receives messages from all district RSUs (or some group of RSUs), aggregates the feeds, and relays them over a single connection to the DEP, shown in Figure 3.. Depending on the hardware and software capabilities of the Aggregation Service, a single service may support all of an agency's RSUs, or they may need to be divided up into smaller, groups based on geographic location or network topology. For this approach, it is necessary for the Aggregation Service to prepend certain meta data about the original source data before sending it to the DEP. This meta data is described in Section 7.

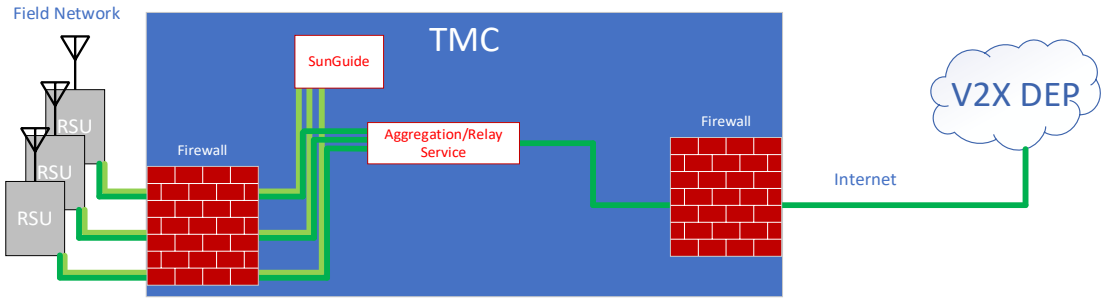


Figure 3. Field CV Data Flow Approach B

- **Approach C – a SunGuide Relay:** This approach is a derivative of Approach B, where the aggregation moves into the existing SunGuide Connected Vehicle Subsystem (CVS) driver, shown in Figure 4. An important note with this approach is that a modification to SunGuide would be necessary, requiring coordination with that FDOT project. Due to the high-volume nature of the CV data and the fact that it does not prescribe to the traditional ATMS message structures used for device management and status, a method of subscribing to the CV data directly to the CVS would be necessary since the DataBus interface does not support that style of streaming data. Therefore, this approach is not yet implemented today.

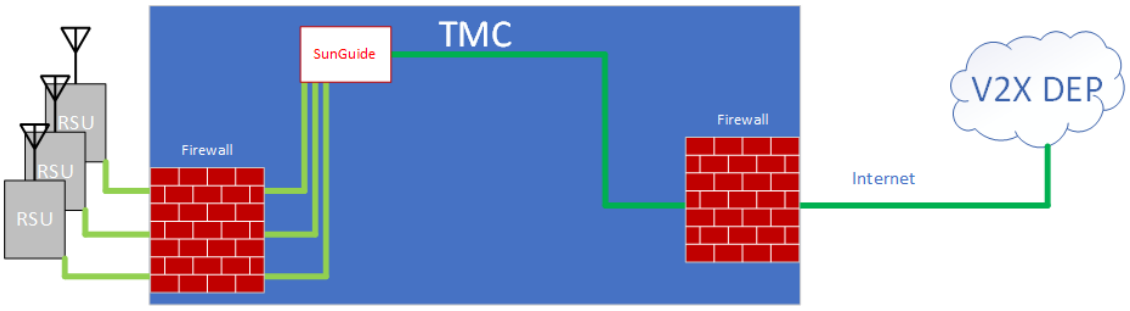


Figure 4. Field CV Data Flow Approach C

Based on prior experience and understanding of existing systems and capabilities, Approach A should present the lowest latency but may require the most complexity with respect to specific device and firewall configurations. Approach C would have increased latency due to the additional hop through SunGuide. Approach B would have variable latency based on the specific implementation of the service including any time-based aggregation window. Therefore, Approach B was chosen as part of ideation meetings with FDOT SMEs.

6.2 Data Consumers

Data will be made available through a variety of interfaces and to a variety of user types. Small subsets may be publicly available without requiring a login or other credentials. Certain categories/types of data may be available to users registering accounts. Access to the majority of the data and interfaces will be controlled by FDOT. The data consumers have access to the data in several fashions.

For internal programmatic data access, the V2X DEP acts as a server and provides Athena for ad hoc queries and Redshift for structured, performant queries. Both programmatic interfaces offer typical database-style connectivity for SQL queries provided by an OBDC or JBDC driver interface, so they appear like standard database data stores to V2X DEP consumers. Programmatic access to the V2X DEP consumer interfaces will be secured by Microsoft (MS) Azure Active Directory (AD) authentication mediated by AWS Cognito.

For external programmatic data access, CV data is provided through the Connected Data Vehicle Framework (CVDF). This interface specification can be used to create an application to access the DEP. Data is queried using filters to return relevant data. Further explanation is provided through the Open Api specification which provides request parameter requirements as well as documentation on each field. Access to use this API is managed by API keys as detailed in Section 9.2.

For an interactive Dashboard User Interface (UI), the V2X DEP has two different types of dashboards: AWS QuickSight and custom visualizations. QuickSight is used for rapid visualization of aggregate statistics and trends. Users with QuickSight author permissions granted by an AWS IAM role can create dashboards and share them for immediate viewership embedded in the DEP user interface. Additionally, some QuickSight dashboards may be predeveloped and deployed by the development team. Custom visualizations are created by the software development team to suit the specific use cases and when tooling is required that is not provided by the QuickSight reporting tool. For example, some visualizations require web-based map interfaces with multiple data layers with custom marker shapes and tool tips. This is detailed further in Section 9.1.

Some systems, such as SunGuide, may have an Application Programming Interface (API) by which they already consume alerts, events, or other external notifications. For these systems and as use cases are

identified and selected for development, their APIs can be integrated into the DEP as additional consumers.

Finally, notifications based on ingested data and analytics may be pushed to consumers via email, text message (SMS), and other custom integrations. Emerging requires for custom notifications are captured as part of the Agile process. Details can be found in the requirements table in the ConOps document.

6.3 VPN

The traffic to and from the V2X DEP is expected to be encrypted. For data producers which do not natively support encryption like SunGuide and CV data forwarded from RSUs, traffic must be secured and encrypted using Virtual Private Network (VPN) tunnels. The V2X DEP is designed to use Virtual Private Cloud (VPC) hosted resources for data ingestion, and each VPC can have only one Virtual Private Gateway (VPG). One VPG can have up to 10 VPN connections by default (configurable by contacting AWS support, limit unknown). The design plan prefers adding ingestion VPCs, if needed, rather than increasing the VPN connection limits. For example, the V2X DEP *may* initially have one VPC for FDOT district TMCs and another for non-district agencies. The delineation of VPCs will be documented as part of the platform design in IaC.

Each VPN connection can handle 1.25 GB/s, and if more bandwidth is required to ingest an agency's data, the agency and DEP needs to aggregate multiple VPN connections. This requires additional configuration for both external agencies and the DEP.

One site-to-site VPN connection is planned for each district with preliminary test connection established with select SunGuide installations (e.g. District 5 and TERL). Local agencies may have their own site-to-site VPN to the DEP, or they may be routed through their respective district network and utilize that VPN. The primary purpose of the site-to-site VPN for the current revision is securing SunGuide network traffic to/from the DEP, since it does not support encryption.

There are some scenarios where data is ingested into the DEP without using a site-to-site VPN:

- Data is only available from in-field network devices and not through an agency data center.
- An agency already provides a public REST API (traffic encrypted using SSL) or other encrypted API.
- In some cases, bandwidth or networking constraints may prevent use of a site-to-site VPN connection, for example, for high volume BSM data.

7 Data Integration Plan

The typical approach for ingestion by the V2X DEP is Docker Containers. Containers are small (relative to full virtual servers) execution environments with the flexibility to provide any of the aforementioned connection types (Pushed, Polled, or Persistent). The Docker Containers are built to consume data from the external application by implementing the 3rd party interface, convert the record format to a common V2X DEP form, and put in the V2X DEP stream for transformation, analysis, and storage. Other approaches include API Gateway to trigger Lambda for a Pushed Connection only or Lambda triggered by CloudWatch EventBridge Rules for a Polled Connection only. A CloudWatch EventBridge Rule is a schedule-based trigger (e.g. hourly, every fifteen minutes, etc. down to one minute precision as determined by the development team, but easily modified as needed). Refer to the V2X DEP Concept of Operations and Requirements Document for more information on operational scenarios. Details for system integration will be provided as part of training materials.

7.1 Curated Record Metadata

Before insertion into the V2X DEP stream, an external message is augmented with metadata fields for tracking purposes. The Metadata format and fields are common to all Record types (Raw and Curated). A Record is a concatenation of the MetaData fields and data type-specific fields. The MetaData fields describe:

- **Id** – A unique identifier for this record.
- **Partition Key** – A unique key for identifying the time-sequence to which this record belongs. For example, if this represents a single crash on the roadway, all updates for the crash would receive the same Partition Key.
- **Version** - The version of the V2X DEP used to ingest, process, and analyze the record.
- **Time** – A collection of timestamps that are set for this field. For example, when the raw data necessary to build the record was *ingested* by the DEP, or when the underlying record was *generated*.
- **Location** – A collection of Point locations to describe the record. Each point can be described in absolute (latitude, longitude, and elevation) or relative (roadway, cross street, etc.) terms. This can be a single point, a sequence of points which represent a directional stretch of roadway, or a bounding box.
- **Source** – A collection of provenance information about the record including descriptions of the data *producer*, the connection or *interface* information, *device* information (if applicable), and *references* to the *Id* of any records which were used to generate the record. For example, there are ingested records which would reference the one or more raw records from which they were built.

7.2 Raw Data Type

To ensure original source data is available, the system archives the original, unmodified raw data from the data source. Raw data is not made available to external data consumers. For details on data retention for specific record schedules, see the V2X DEP Security Documents. Note that the record schedules are implemented for raw data currently to adhere to 3rd party data agreements. The default raw data retention is specific in the CDK Context, and raw data retention per producer may also be specified.

Table 4. Raw Data Sources

Curated Data Type	Raw Format	Original Data Source
Ford	JSON string	See: Ford Data Dictionary
Vehicle	J2735 UPER hex string	SAE J2735 DSRC Message Set Dictionary
Traffic	SG TSS: XML String HERE: JSON String	SG Schemas: Schemas HERE: Documentation of HERE API
RoadwayEvent	SG EM: XML String Waze: JSON String	SG Schemas: Schemas Waze: Documentation of Waze Feed
Map	J2735 UPER hex string	SAE J2735 DSRC Message Set Dictionary

Curated Data Type	Raw Format	Original Data Source
Message	SG DMS: XML String SG CVS: XML String	SG Schemas: Schemas
SPaT	J2735 UPER hex string	SAE J2735 DSRC Message Set Dictionary
Weather	XML String	SG Schemas: Schemas

7.3 Curated Data Types

A limited set of generic curated data is used to represent the data from many data producers as listed and detailed in Table 5 – Curated Data Types. It is expected that as the system is developed, additional data types will be added.

Table 5 – Curated Data Types

Curated Data Type	Details	Data Sources
Ford*	Data pertaining to a Ford OEM vehicle status update.	Ford
Vehicle	Data pertaining to a single vehicle like bearing, speed, and various other status fields.	CAV RSU BSM
Traffic	Aggregate information about many vehicles or flow of traffic in general at a roadway location.	SunGuide TSS Links HERE Flow
RoadwayEvent	Roadway or traffic events outside of normal operation like crashes, closures, incidents, hazards, alerts, etc.	SunGuide EM Events Waze Partner Hub Alerts and Jams
Map	Information about intersection or roadway configuration	CAV RSU CV Intersection MAP Manual CAV MAP file upload
Message	Messages to the traveling public.	SunGuide DMS Signs SunGuide CVS TAMs
SPaT	Information about intersection signal phasing & timing	CV Intersection SPaT
Weather	RWIS (road weather information systems)	SunGuide RWIS

*Note, due to unique data sensitivity concerns and the fact that the Use Case ideation for Ford Curated Data Type is still ongoing, an interim implementation with a standalone Curated Data Type called “Ford” was chosen to isolate the data and provide a preview to stakeholders for a limited set of fields. Long term implementation may include merge of Ford into the Vehicle data type.

Nuances for specific data source connections are outlined in the following sections. In all cases, when there are sensitive API keys or usernames and passwords, a secrets manager is used to securely store the information. A secrets manager is a secure storage system for API keys, passwords, certificates, and other

sensitive data and provides a central place and single source of truth to manage, access, and audit secrets across. The DEP uses the AWS Secrets Manager.

7.3.1 Curated Data Schemas

The schemas are now accessible through the DEP web UI as follows:

1. Login to the Web UI.
2. Navigate to the AWS Console.
3. Navigate to AWS Glue.
4. On the left-hand side select “Data Catalog”.
    a. This should expand a menu.
    b. Select “Schemas” from the menu.
5. Click on a schema name.
6. Select the version of the schema you would like to view.
    a. A larger number corresponds to the more recent version.
7. The schema in JSON should be displayed.

7.4 CAV

Connected and Automated Vehicle data contain status information from connected vehicles such as location, speed, heading, braking status etc, which is relayed by RSUs to the V2X DEP. The typical connection to the V2X DEP from an RSU is a Pushed Connection. The DEP Forwarding tool, running in a linux Docker Container, acts as a UDP server accepting the UDP message frames from RSUs having a predefined IP. Each packet from the RSU contains a binary J2735 message frame which may be one of several types (e.g., BSM, SPaT, etc.). Additionally, these messages may be wrapped in an IEEE 1609.2 security header. Regardless, the raw data for each UDP packet received by the container is archived and transformed to a corresponding Avro message, augmented with meta data fields, and put in the stream pipeline. Meta data fields contain information from the UDP headers like source IP to uniquely identify the source RSU. Therefore, in the case of CAV data, IP maps to both the Producer Id and Device Id.

Rather than getting a direct connection/stream of data from each individual RSU to the DEP, the UDP Forwarding tool receives data from RSUs, transforms the data into the Avro message, and pushes the transformed Avro message to the DEP for D2 and D5.

CAVs are a priority data source for the V2X DEP, and one of the major challenges for CAV data is scale. The data rate of Basic Safety Messages (BSM), the typical message frame from CAVs to-date, from a single vehicle Onboard Unit (OBU) is nominally 10 Hz. As the number of RSU deployments grow and more vehicles have compatible OBUs (either direct from OEM production vehicles driven by the public or agency deployed/managed OBUs on fleet vehicles), so must the V2X DEP scale to meet these needs. To ensure scalability, the data ingestion pipeline leverages serverless resources since serverless resources are fully managed with intrinsic scalability. The V2X DEP makes use of Docker Containers hosted in a fully managed cluster called AWS Elastic Container Service (ECS) Fargate. A Docker Image coupled with a Fargate Task Definition provides a small unit of scaling and/or redundancy deployable in an automated fashion by an ECS Service. In this way, a Network Load Balancer (NLB) may be placed in front of the cluster to distribute the packets across the number of Docker Containers currently available and healthy, and the ECS Service will automatically adjust the number of docker Containers depending on load changes.

Other than BSMs, there are other CAV data types which are ingested by the DEP from RSUs at different data rates including MAP and Signal Phase and Timing (SPaT), and more still that could be ingested using

the same connection in the future including Personal Safety Message (PSM), Probe Vehicle Data (PVD), Road Safety Message (RSM), Traveler Information Message (TIM) provided outside of SunGuide (Note to capture these TIMs, Approach C – SunGuide Relay above would not be possible).

7.5 SunGuide

SunGuide (http://sunguidesoftware.com) is a modular Advanced Transportation Management System (ATMS) acting as a secondary source for several raw device data types like TSS (roadside-detector-measured speed, volume, and occupancy) and a primary source for data types like Events, DMS messages, or CVS TAMs. DataBus is the communications backbone which each SunGuide module uses to retrieve from and subscribe to data from other modules. The V2X DEP supports connections to multiple SunGuide instances each with their own version (8.1 HF2, 8.1 HF3, 8.1 HF4, 8.2 HF1, etc.) using existing SunGuide ICD specification.

Each SunGuide-specific Fargate Container from the V2X DEP acts as a client to SunGuide and makes a connection to a particular DataBus instance from an FDOT district or other SunGuide instance (e.g. CFX, FTE, etc.) through VPN. A V2X DEP Docker Container connects, authenticates, subscribes, and retrieves XML data from SunGuide DataBus for the following subsystems:

- Event Management (EM):
  - Incident (all Event features represented in the DEP outside of the specifically identified Crash and Lane Closure features), Crash, Lane Closure, etc.
  - Note: Event features which are PII or sensitive (see the Security Plan Document) are excluded
- Dynamic Message Sign (DMS):
  - DMS Configuration and Status
- Connected Vehicle Subsystem (CVS):
  - RSU Configuration, Status (note CVS RSU data is not currently ingested as a curated data source in the DEP, but rather used for outbound real time notifications from the DEP) and TAMs
- Road Weather Information Subsystem (RWIS):
  - Air Temperature, Humidity, Visibility, Wind Speed, Wind Direction, Precipitation Type, Precipitation Rate, Cloud Cover, Water Depth (via SunGuide)
- Transportation Sensor Subsystem (TSS):
  - Speed, Volume, and Occupancy

In the future, additional connections are planned for the Travel Time Subsystem (TVT). In some cases, information from SunGuide could qualify as PII (such as events containing license plates). In this case the information is hashed using SHA-256. The SunGuide Software Administration Application (SAA) username and password and DataBus host and port are coordinated with the SunGuide stakeholder in advance for use by the V2X DEP Fargate Container.

7.6 Waze

Waze Data Feeds provide crowd-sourced information about road closures, accidents, and working vehicles such as garbage trucks. FDOT's Waze Partner Hub instance (https://www.waze.com/en/signin?redirect=/partnerhub/) is ingested by the V2X DEP and treated similarly to SunGuide Events for representation purposes (they share the same curated data type). CloudWatch EventBridge Rules are scheduled to trigger the retrieval of the feed on a two-minute interval (the update period from Waze's documentation). The CloudWatch EventBridge Rule triggers a serverless snippet of code called Lambda which will use a Polled Connection to get the JSON Incidents from the Waze

API over HTTPS. The feed is then be transformed and put on the streaming pipeline. One Waze API key is used across all FDOT V2X DEP Lambdas to authenticate with the Waze API. Like all sensitive keys, the API key is stored and managed in Secrets Manager.

7.7 HERE

HERE provides traffic flow data containing speed (both raw and a speed-limit-capped value) and congestion ("jam factor") information. The HERE Traffic API (https://developer.here.com/documentation/traffic/dev_guide/topics_v6.1/flow.html) data is consumed using FDOT's pre-existing license and treated similarly to SunGuide Events for processing purposes. Similar to Waze ingestion, HERE will also be ingested through CloudWatch EventBridge Rule-triggered Lambdas using a Polled Connection. HERE Traffic data is consumed with its confidence intervals for later inclusion in analysis.

8 Data Storage Plan

Data storage takes two primary forms: data in motion and data at rest. In either case, the data governance aspects of this data are described in the V2X DEP Security Plan. To support real time ingestion, transformation, fusion, analysis, and notifications, V2X DEP uses a streaming pipeline for data in motion. The data at rest is stored in three AWS products: Redshift data warehouse, S3 data lake, and S3 for raw data. To support queries and analysis of current and historical data, V2X DEP leverages cloud-based tools which adapt the data stores to standardized interfaces (Redshift, Redshift Spectrum, and Athena). The usage estimation below is based on the highest volume data sources including 500 traffic sensors with 3 updates per minute across FDOT districts, Florida Turnpike Enterprise, and other SunGuide installations; and 3.6 billion Ford sensor messages corresponding to 180 GB per month at 50 bytes per message. Refer to Section 11 for additional details.

8.1 Data in motion

A streaming architecture allows ingestion of data from many different sources and makes that data available for consumption by a real time analytics and Extract, Transform, Load (ETL) application, and other data sinks. A streaming architecture can be thought of as high-capacity pipes which scale as needed. Streaming architectures facilitate partitioning logical divisions and ordering of the data in motion and ease the burden of scaling to handle new loads without performance penalties. These pipes allow many writers (internally called Producers) and readers (internally called Consumers) of data.

In AWS, V2X DEP leverages the Kinesis tooling as the streaming backbone. For data transit, V2X DEP uses KDS at the point of ingestion and Kinesis Data Firehose (KDF) for upload to the data stores like S3. For intermediate extract, transform, and load (ETL) operations, Kinesis Data Analytics (KDA) reads data from the ingestion stream. The average usage of Kinesis for data in motion is estimated below (Table 6). Note that the values in the table are initial estimates, but several pieces of the architectures are designed to automatically scale up or down in response to changes in data throughput. Refer to the "Data Pipeline and Internal Processing Design" section of the V2X DEP Baseline Architecture Document for additional information including a diagram.

Table 6. Average usage per month for data in motion

Service	Metric	Units Monthly
VPC Endpoint for Kinesis	Endpoint per Availability Zone (AZ) per Hour	1440
	GB processed	2560

Service	Metric	Units Monthly
Kinesis Data Streams	GB/month	2560
	Stream/hour	4320
Kinesis Data Firehose	GB/month	2560
	1000 objects to S3	44
	GB to S3	2560
Kinesis Data Analytics	Kinesis Processing Units (KPU) per hour	7
SNS	Email and Lambda/TCP	3000

Serverless streaming architectures, like all serverless products, are meant to vary cost continuously with the scale of the application and Kinesis provided by AWS is no exception. To minimize cost of the serverless streaming architecture, V2X DEP uses Avro, the binary record format, for records put on the stream. Avro is an industry standard binary format which allows a minimal serialized size, good serialization and deserialization performance, and a robust implementation for a variety of language including C#, C/C++, Java, Python, and others. Refer to the "Data Pipeline and Internal Processing Design" section of the V2X DEP Baseline Architecture Document for additional information including a diagram.

Kinesis services were chosen for the reasons listed below:

- It integrates with other AWS services which are used in the project, particularly: 1) CloudWatch for activity logs; 2) CloudTrail and Config for configuration change tracking.
- It is highly available with native replication across 3 Availability Zones (AZ).
- It supports the low latency requirements which are in line with the project SLAs (see V2X DEP Concept of Operations and Requirements Document).

8.2 Data at Rest

At various stages in the streaming pipeline, the data is stored in a serverless data store. To ensure original source data is available, the system archives the original, unmodified raw data from the data source. The transformed, aggregated, or fused data formats (collectively called curated data) are also stored for later query. In AWS, Simple Storage Service (S3) provides a low-cost, relatively infinite, serverless object storage medium. S3 stores and organizes objects within buckets, which can be individually configured with multiple AWS built-in encryption options. The V2X DEP is encrypting all data at rest using AWS encryption services which uses 256-bit Advanced Encryption Standard (AES-256) GCM.

A variety of interfaces can adapt S3 to other interface formats. For example, both KDA and KDF can write and partition streaming records directly to S3. KDA is used to write data in bulk Parquet format, whereas KDF is used to write multi-record Avro files to S3. Parquet supports the query-able data lake and bulk Avro files are meant as an archive for raw ingested data. KDA is used to perform transformation on data and write to other datastores such as a dedicated database. The average storage and usage expectations for the V2X DEP serverless data stores are listed below (Table 7).

Table 7. Average usage per month for data at rest

Service	Metric	Units Monthly
S3	GB month S3 Tier	30720
	GB month IA Tier	30720

Service	Metric	Units Monthly
VPC Endpoint for RedShift	Endpoint per AZ per Hour	1440
	per GB processed	2560
Redshift	ra3.4xlarge per hour	1
	Spectrum queries per hour per GB	288000
Athena	Athena queries per hour per GB	10
Cognito	Monthly Active Users: First 50,000 free	<50,000

The data at rest storage in S3 is optimized to reduce hosting costs using a columnar (less sparse, more compact) storage format called Parquet. With increasing numbers of records per file, this columnar data organization scales down storage size requirements and increases query performance by reducing the amount of data that must be scanned. These features of Parquet reduce the cost to store the vast amounts of data the V2X DEP captures. The Glue Data Catalog and Athena Query Engine are optimized for efficient use of the aforementioned Parquet formatted data.

Different classes of data at rest are subject to different Governance and Lifecycles that are outlined in the Security Plan. Through AWS S3 configurable lifecycle policies (defined within the CDK Context JSON file [See Section 3.2]), the platform will manage transitions from hot to cold storage medium and to deletion for raw and curated data. Other data types outside platform curated and raw data lifecycle policies are managed per application (e.g. application logs are managed in CloudWatch). Please see Security Plan for additional details.

9 Data Dissemination

The V2XDEP provides the following interfaces that allow users to retrieve data.

9.1 Users and Groups

The platform leverages Cognito federated identities for authentication through either Cognito User Pool for “external” users without an FDOT email address or Microsoft Azure Active Directory (AD) for “internal” users with an FDOT email address.

In response to the successful completion of the FDOT AARF process, the Developers provision and manage “external” users in AWS Console for Cognito using User Pool Users and Groups. The FDOT AARF process is used as a way to document the approval(s) required for providing users access to the DEP. Developers maintain users through Cognito and perform activities such as revoking access, resetting passwords and various other tasks.

FDOT manages Azure AD users and groups directly outside the platform. The names of both Cognito User Pool Groups and FDOT’s Azure AD groups are the same and match those in the Security Plan Section 3.1. These groups also provide a way to give specific permissions to APIs to specific users. API keys (as used in Section 9.5) are given permissions based on the group the key is associated with. API keys were designed to not automatically expire, but Developers are able to expire specific keys through the Authorization Admin API. More details related to authentication as well as the security groups used in the DEP may be found in the V2X DEP Security Plan.

9.2 User Interface

There is a platform User Interface (UI) which uses react framework and typescript. The UI is a single page, thin-client, web application designed to be flexible to host or route to many other platform UI elements (See later section related to Athena Query Editor, QuickSight Dashboards, and CloudWatch Dashboards).

9.2.1 Login

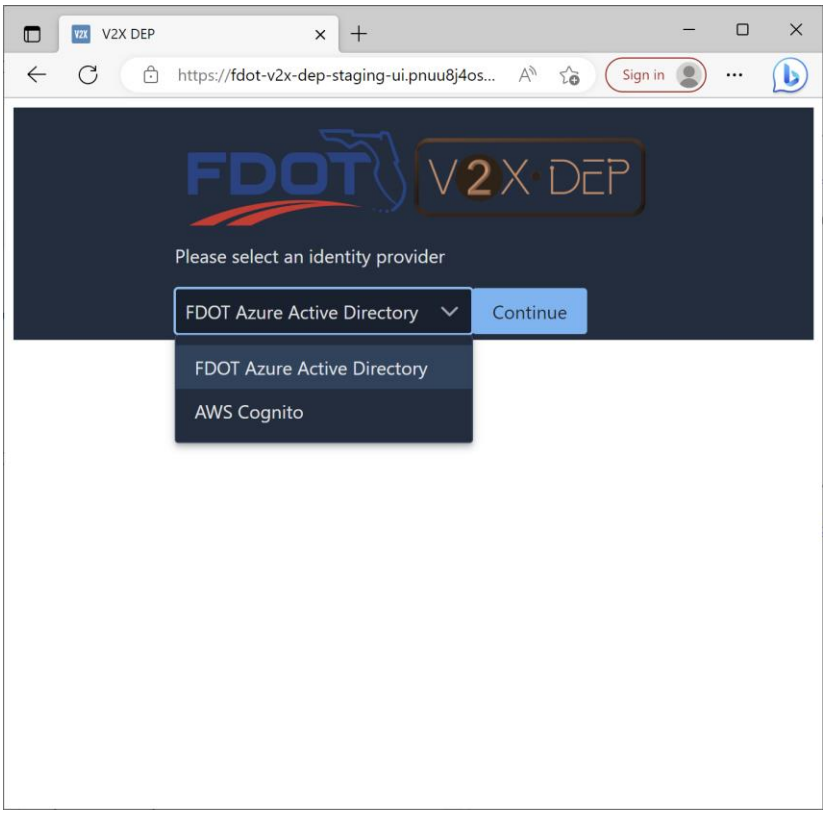


Figure 5: Login Page

9.2.2 Main page

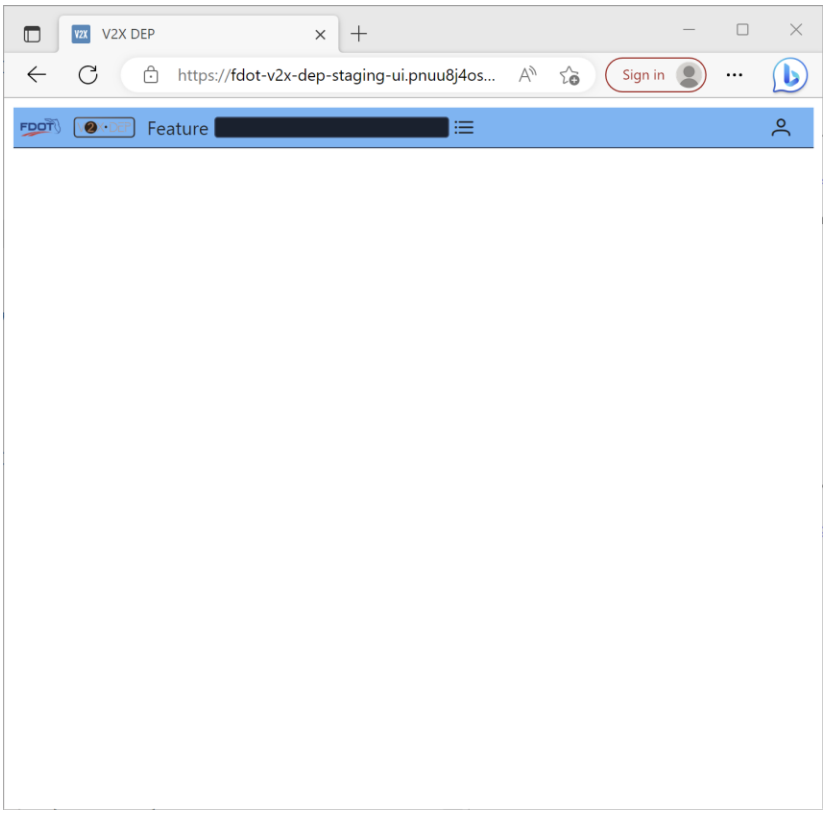


Figure 6: Main Page

9.3 Work Zone Data Exchange (WZDx)

The WZDx is a USDOT compliant (https://datahub.transportation.gov/Roadways-and-Bridges/Work-Zone-Data-Exchange-WZDx-Feed-Registry/69qe-yiui) feed that is built using the Washington State Department of Transportation (WSDOT) wzdx.models (https://github.com/WSDOT/wzdx.models). The feed is the endpoint of a data pipeline starting with SunGuide and other data producers. When these producers create or indicate new roadway events, the events are ingested into the data lake. Every 15 minutes (configurable), a lambda function queries the data lake to see what roadway events are active, recently opened, or closed. With that information, the lambda function then updates the public work zone data exchange (WZDx) feed with the relevant active events by replacing the pre-existing feed with a new one. The feed, proper resides in an S3 bucket (workzone-feed-{env.Account} where the account specifies the specific environment) and is accessible to the public at a URL that is TBD.

9.4 Connected Vehicle Data Framework (CVDF)

The CVDF API is a REST API using API Gateway (GW) backed by an AWS Lambda Function which leverages the DEP Redshift Data Warehouse. This API is authenticated by an API key created using the process described in Section 9.1. Users that have been granted permission to access data from this API will need to reference the swagger documentation for correct usage of providing the API key within their request.

The API currently supports retrieval of active Traveler Information Message (TIM) data as well as MAP data. Ingestion for TIMs to supply the DEP Redshift Data Warehouse occurs through the SunGuide CVS as the ultimate source of data. Ingestion for MAP data is done through the usual ingestion processes with the source of data being deployed RSUs that have been configured appropriately to send the DEP data.

See Section 6.1.1 for configuration of RSUs. See Section 7.3 for further discussion of the CAV data types. Otherwise, MAP data is ingested through a bulk upload process using zip archives of MAP data provided by districts.

The API for both data supports geospatial querying, as well as pagination to retrieve multiple pages of data. Pagination usage details, including the maximum number of results contained within each page, as well as knowing when the last page of data has been reached, can be found within the swagger documentation webpage. At a high level, pagination works by querying the back-end database for a defined maximum number of records, which allows the API to provide the response results more quickly and using less memory, which improves the overall user experience. Offset pagination is the technique currently being used for pagination, which is simple to understand and implement, but does not guarantee consistent results between page requests, as new data is being ingested in the back-end database.

9.4.1 Swagger UI Documentation

The V2XDEP provides a Swagger documentation webpage which is a standard tool used to provide users with various details about an API. It is an interactive webpage, meaning users can test out API calls directly from a web browser. For reference, a screenshot of the CVDF API Swagger UI webpage is shown within Figure 7.

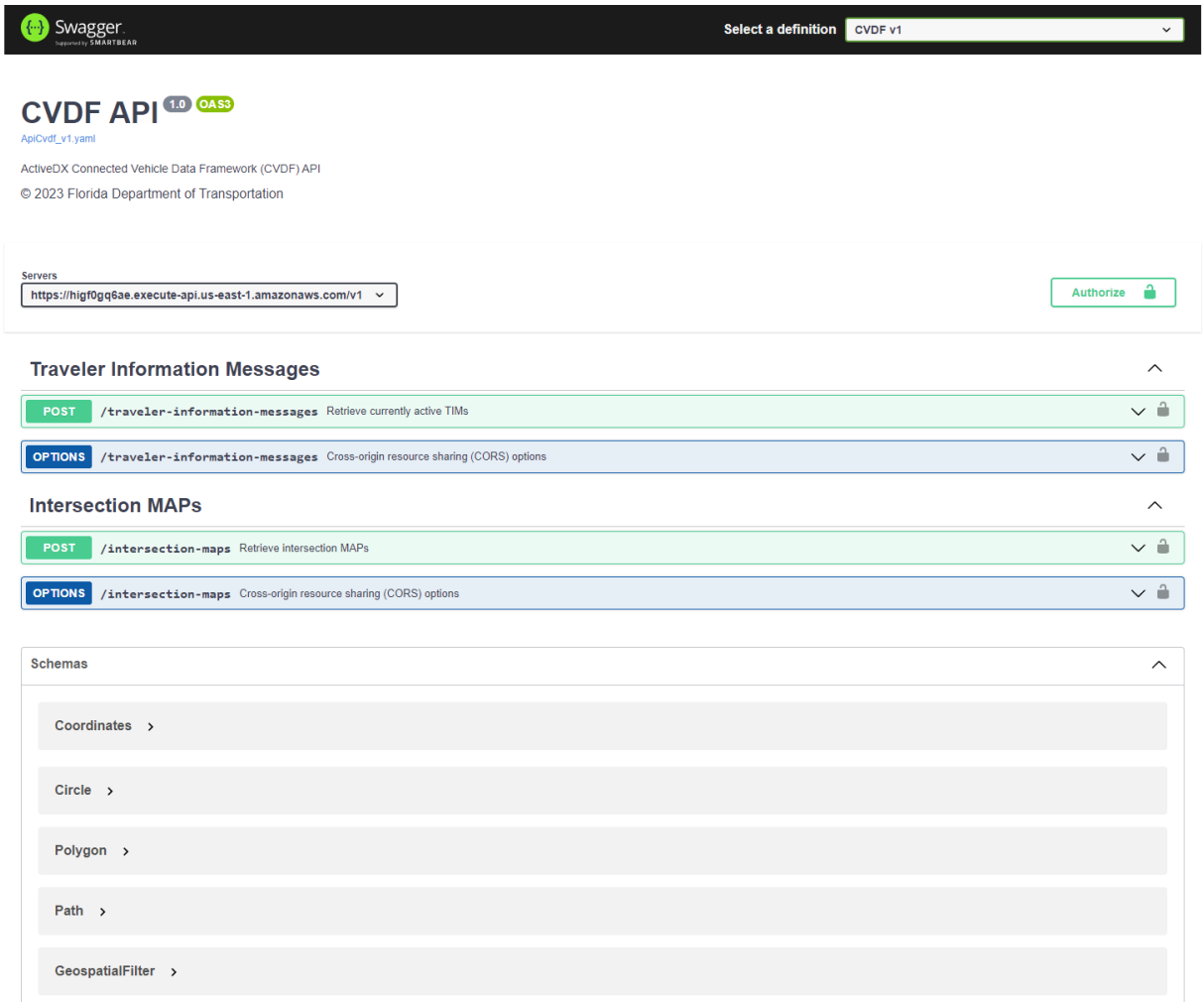


Figure 7: CVDF Swagger UI Documentation

9.4.2 CVDF TIM Implementation Components

The CVDF TIM implementation utilizes a handful of AWS components, which are illustrated within Figure 8.

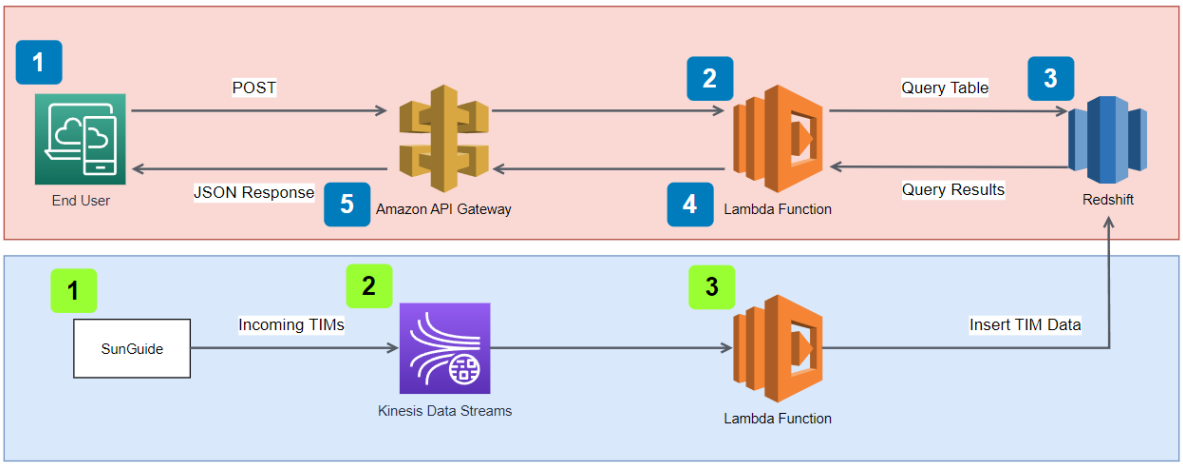


Figure 8: CVDF TIM Implementation Components

9.4.3 CVDF TIM Implementation Sequence

Detailed sequence/process steps for both the API Gateway and Data Ingestion processes are detailed within Figure 9.

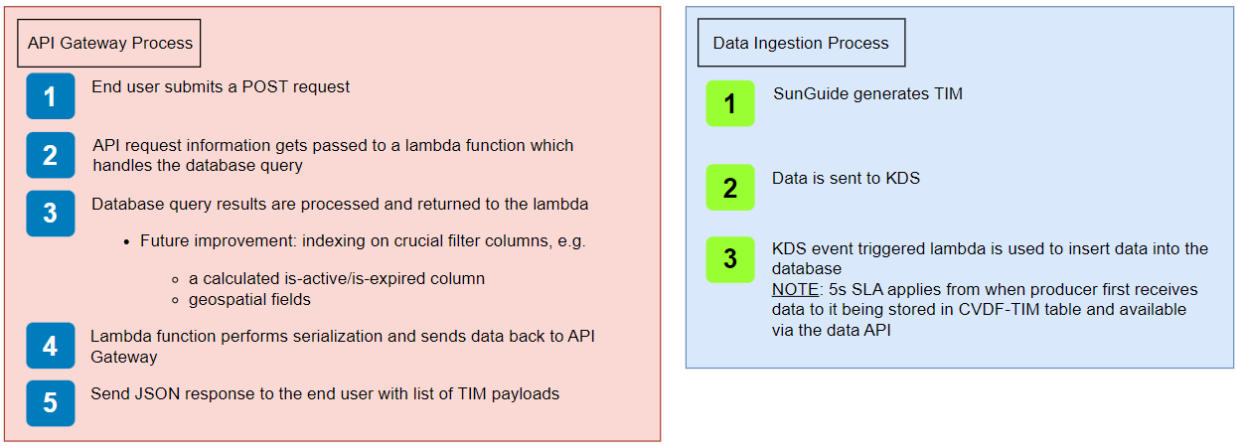


Figure 9: CVDF TIM Implementation Sequence

9.4.4 CVDF MAP Implementation Components

The CVDF TIM implementation utilizes a handful of AWS components, which are illustrated within Figure 10.

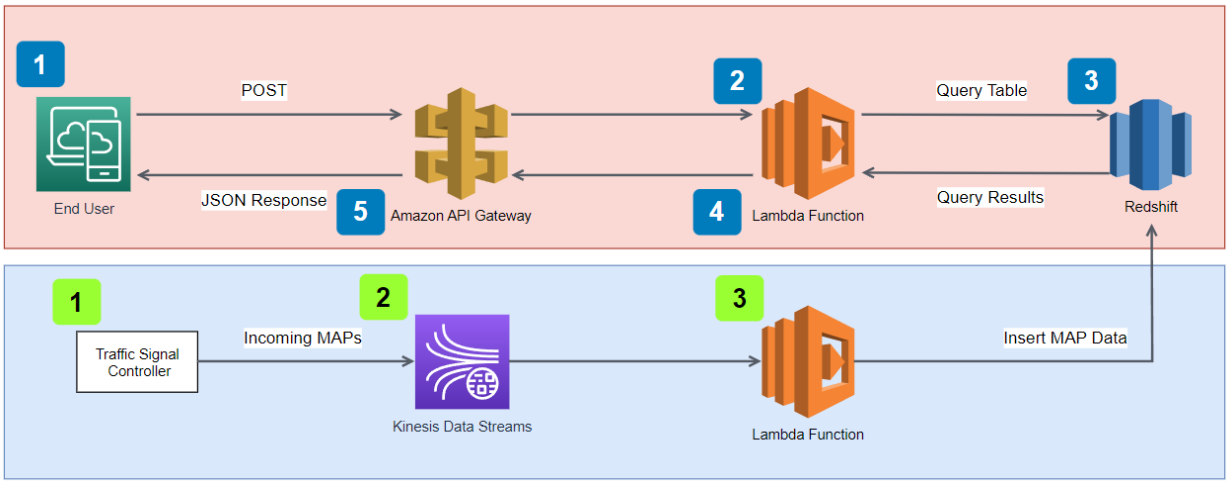


Figure 10: CVDF MAP Implementation Components

9.4.5 CVDF MAP Implementation Sequence

Detailed sequence/process steps for both the API Gateway and Data Ingestion processes are detailed within Figure 11.

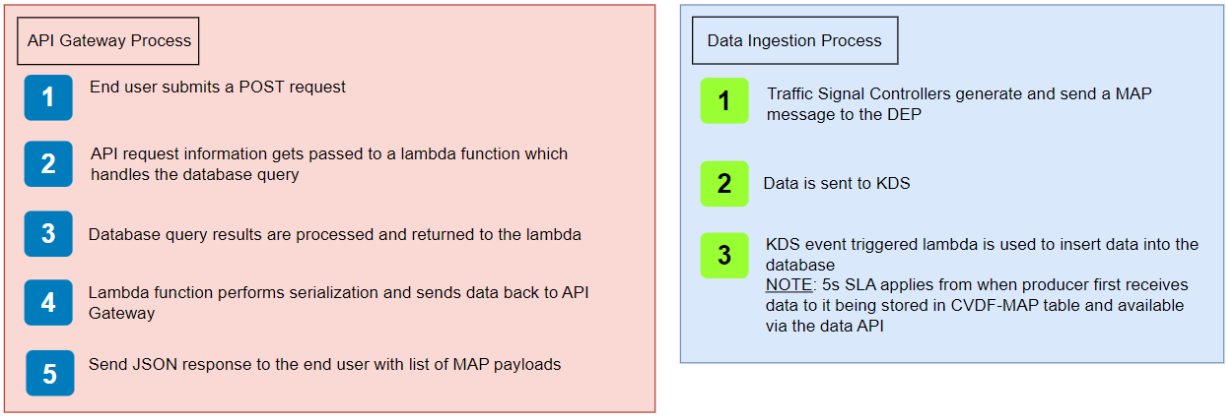


Figure 11: CVDF MAP Implementation Sequence

9.5 Athena & Redshift Queries

Data in the DEP may be queried using AWS Athena or AWS Redshift using standard SQL. For each query that is run in Athena, the results of the query, along with its associated metadata information, are automatically stored within a user-specified Amazon S3 bucket, which can be downloaded.

9.5.1 Athena Query Editor

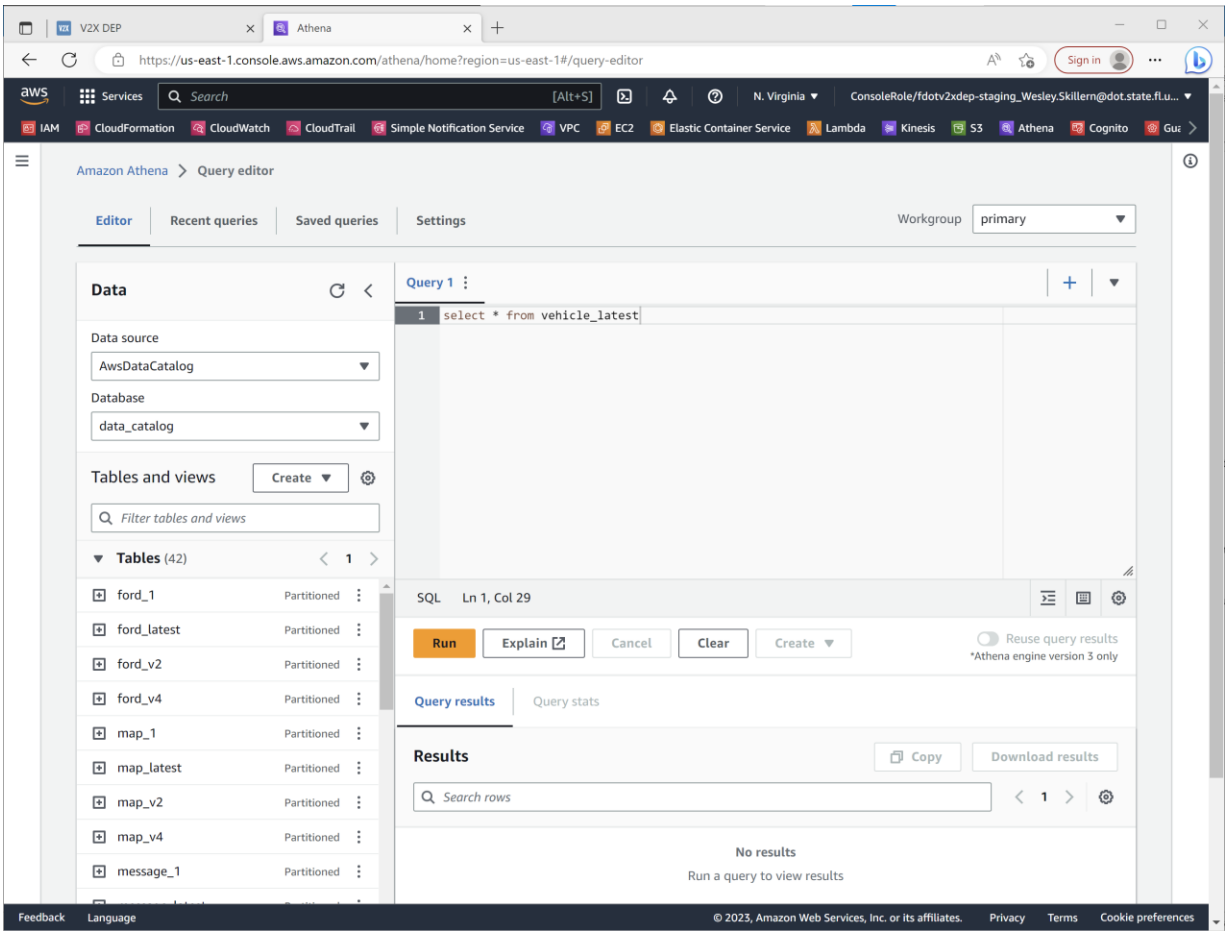


Figure 12: Example Athena Query

9.6 CloudWatch Dashboards

9.6.1 Health Status Dashboard

The Health Status Dashboard is meant for users of the platform to quickly get a feel for data rates (by curated data type), latencies (by curated data type), and availability of public endpoints. They are also used to validate the DEP's contract SLA for data ingestion

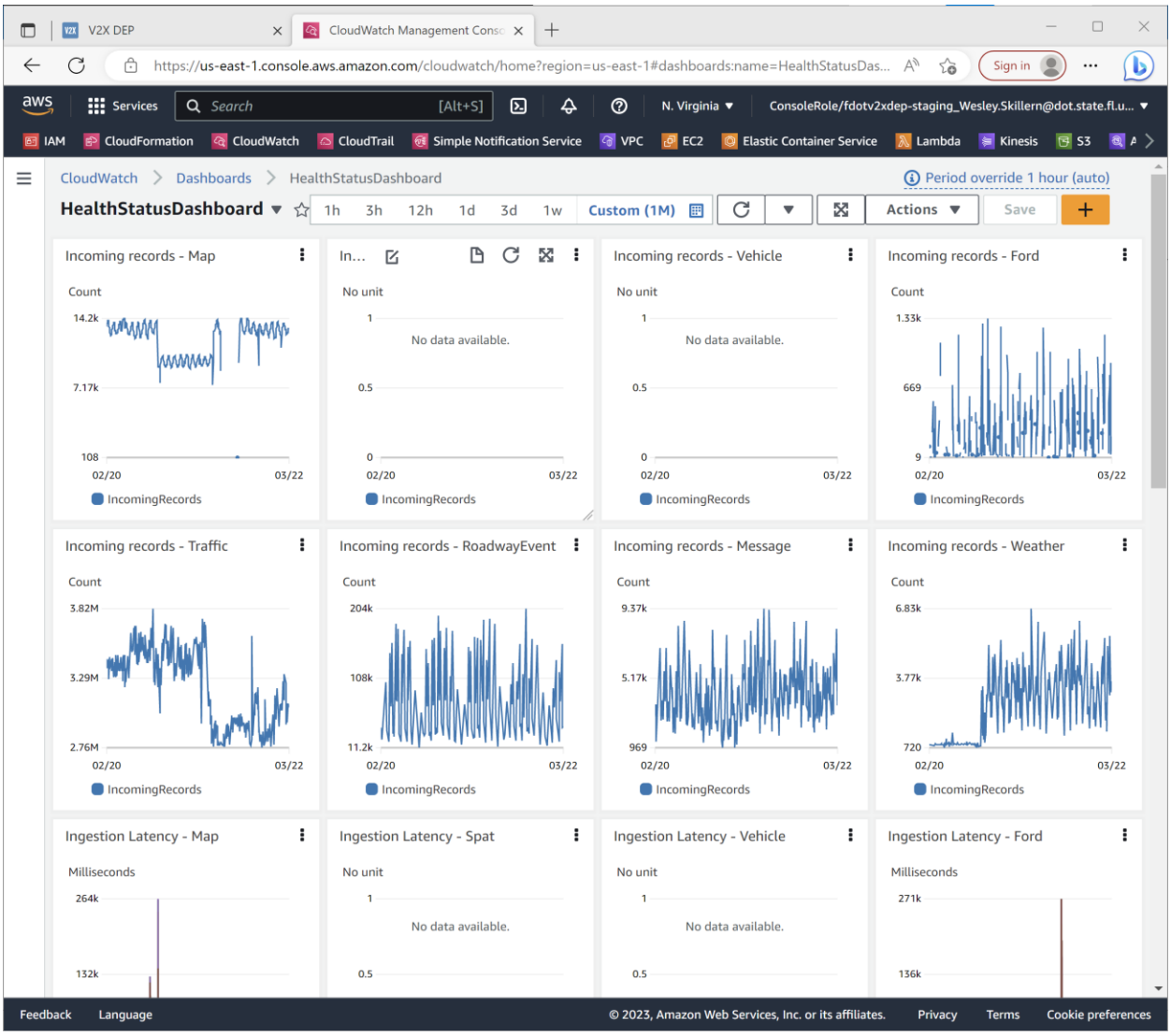
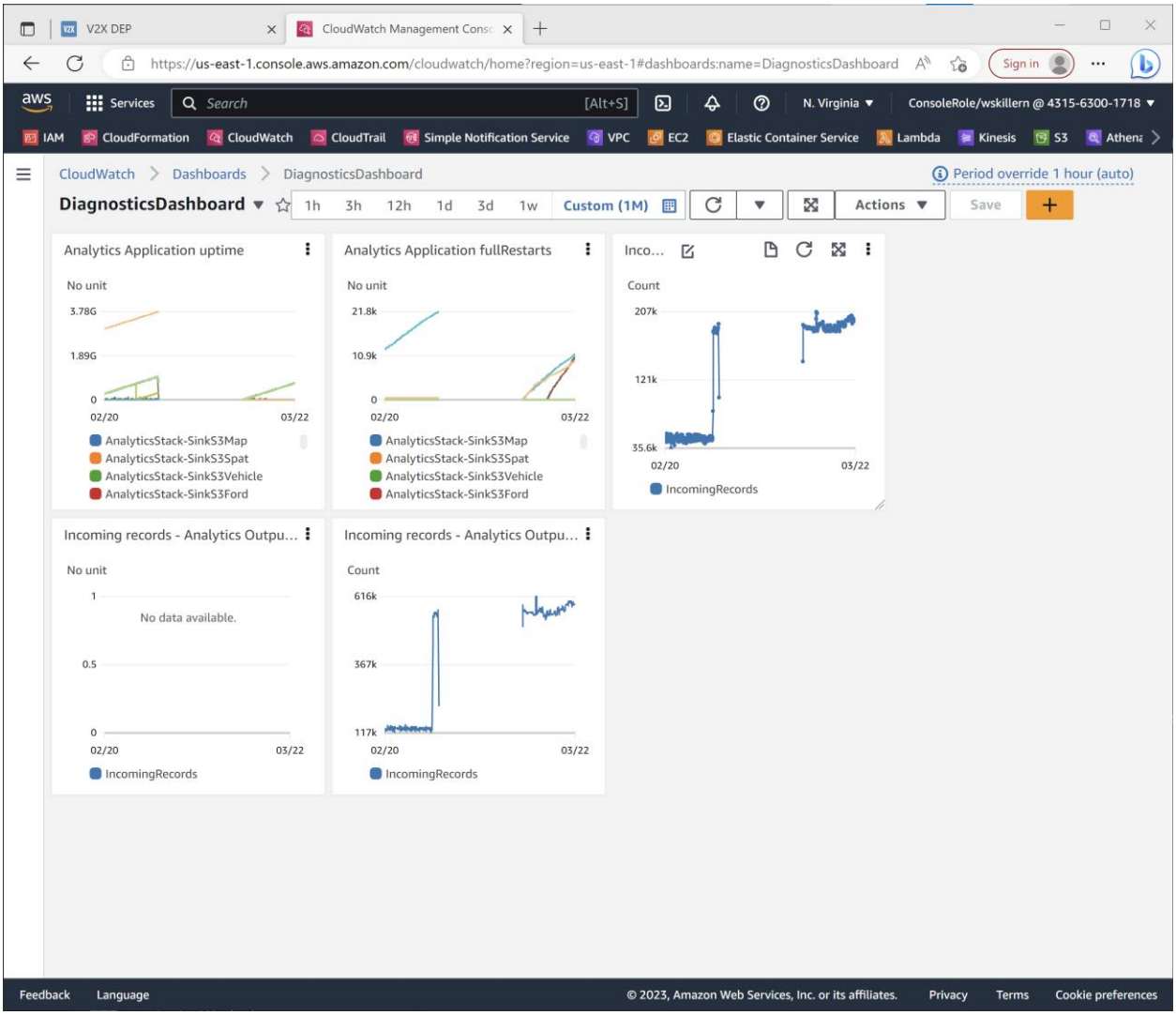


Figure 13: Sample Health Status Dashboards

9.6.2 Diagnostics Dashboard

The Diagnostics Dashboard is meant for power users or system administrators to understand and diagnose internal operation of the platform including stream processing application status by data type.



9.7 QuickSight Dashboards

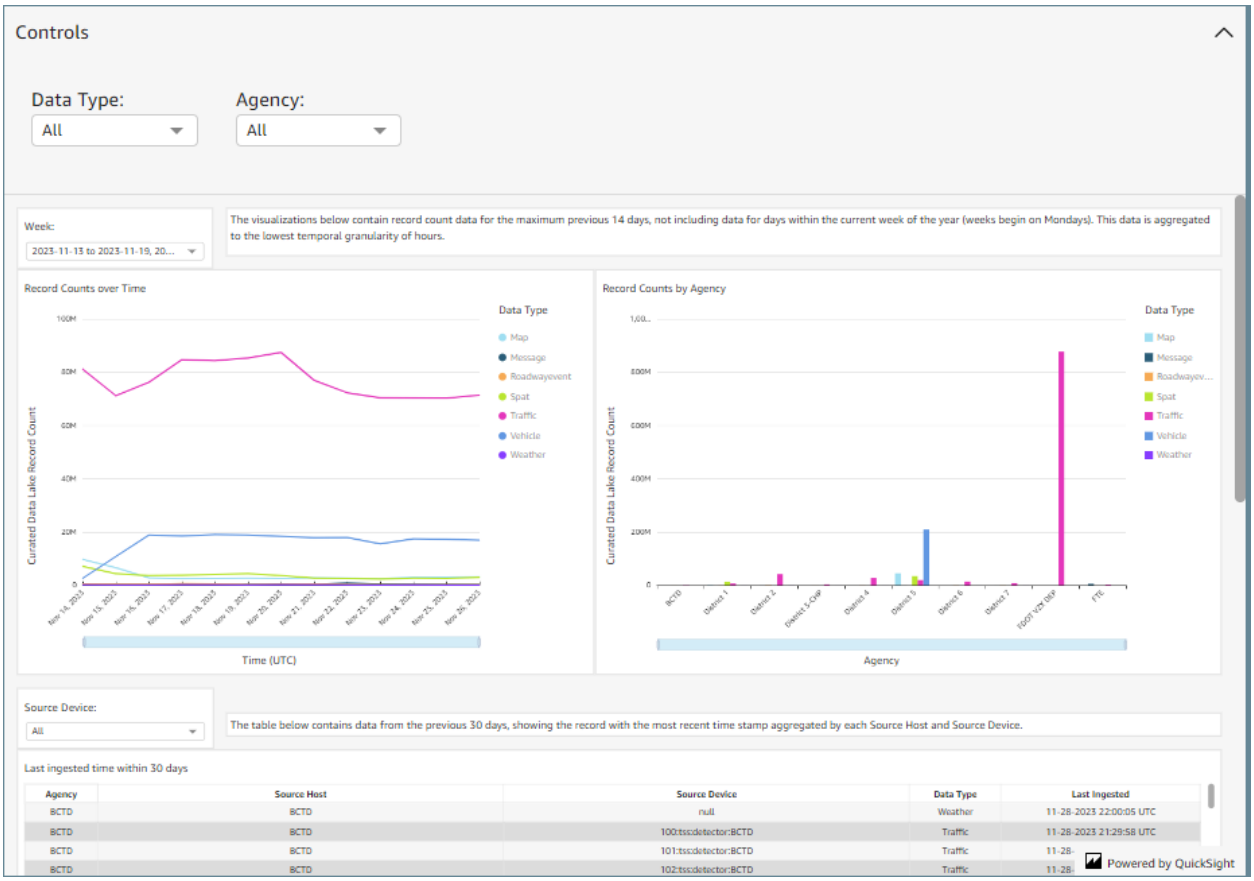
The primary motivations behind the use of QuickSight are that it allows for rapid development of visualizations, it integrates well with other AWS resources, its ability to run a query on interchangeable data sources (e.g. Athena or Redshift), and its ability to use the same analysis on different datasets as long as the field names stay consistent. The QuickSight Data Ingestion Dashboard makes use of the SPICE (Super-fast, Parallel, In-memory Calculation Engine) in-memory engine, which optimizes the speed and responsiveness of the visualizations.

If the client requirements exceed the QuickSight capabilities, other custom graphical representations may be developed in place of QuickSight without redesigning the underlying data access mechanisms.

9.7.1 Data Ingestion Dashboard

This dashboard contains multiple visualizations that allow users to quickly see what data types are being ingested into the V2XDEP and from where the data is coming from. There are several interactive filters within the dashboard which allows the user to quickly manipulate the data visualizations to meet their specific interests. Curated record counts are aggregated hourly and displayed in a temporal visualization,

which allows the user to see ingestion counts over time. Additionally, a data table contains details that show the most recently ingested time stamps for records, which are grouped by each Source Host and Source Device.



9.8 Custom Visualizations

While QuickSight is primarily used for rapid visualizations of aggregate statistics there are often cases where more customization is required. These custom visualizations are created by the software development team to meet these specialized use cases. Currently the DEP has one such visualization to exhibit Basic Safety Messages (BSMs) and roadside units (RSUs). This visualization shows the location of BSMs and the respective RSU that transmitted the data to unveil dead zones in coverage.

9.9 SunGuide TIM Dissemination

The diagram below presents a sequence diagram for the process to send TIMs to SunGuide for broadcasting via RSUs connected to SunGuide. The design is flexible enough to provide facilities to send messages directly to third parties including individual motorists, but the initial plan is that TIMs will be provided through SunGuide.

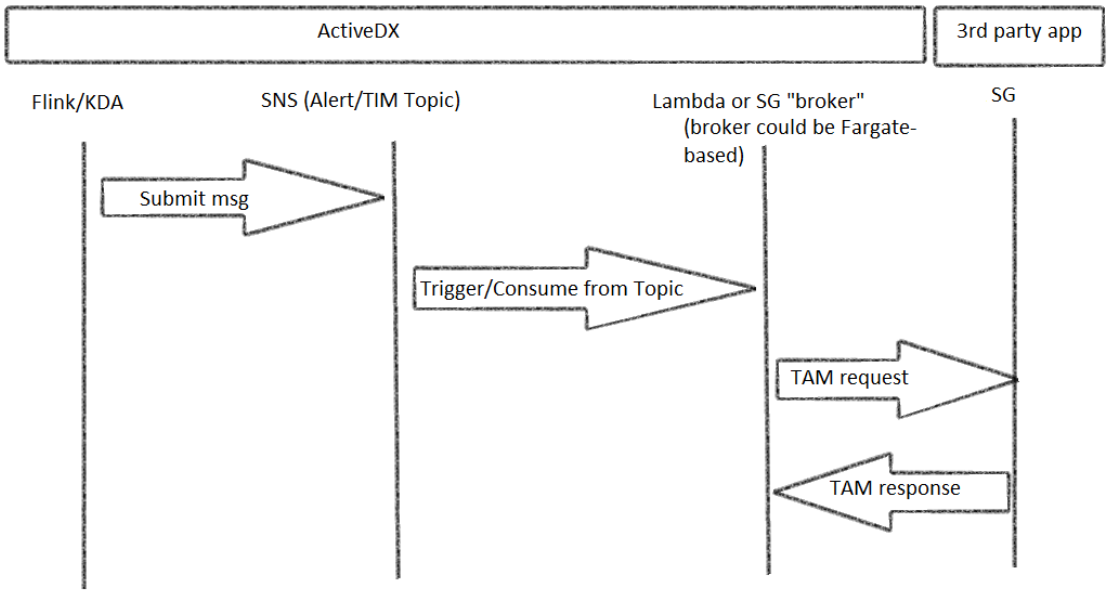


Figure 14. High-level Notification Flow

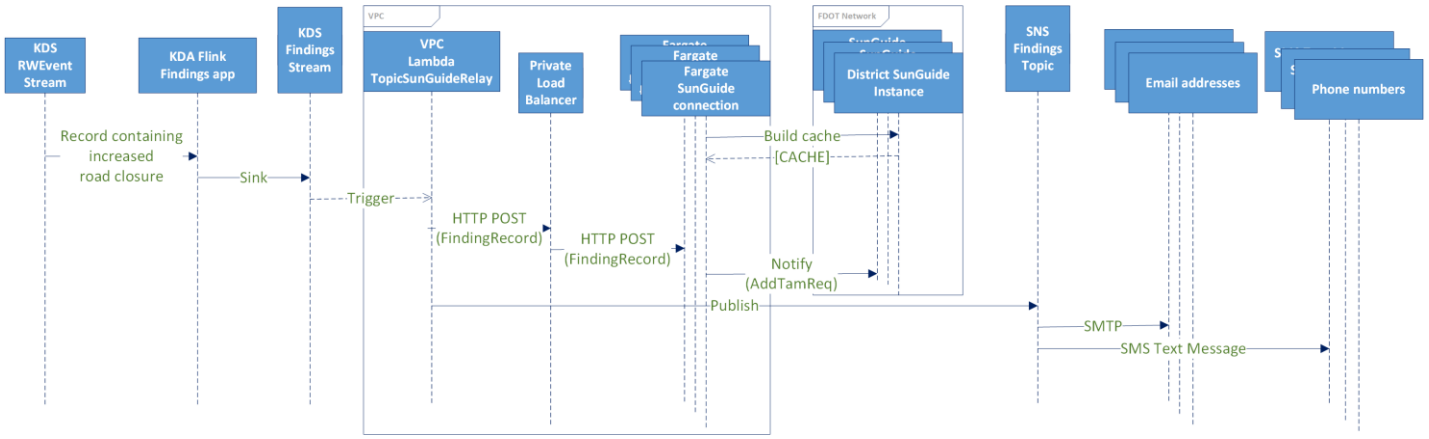


Figure 15. Detailed Notification Flow

10 Data Schemas

The Avro schemas shown below reside in source control at the location below and are current with the delivered source code.

In addition, the schema registry in the DEP is reachable at the URL below and accessible to users with the correct permissions or System developers (see Section 1.2 regarding audience).

https://us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#catalog:tab=schemaRegistries;schemaRegistriesView=view-schema-registry?registryName=StreamingSchemaRegistry

11 Data Locations

Curated data is organized by data type - not data source. The exception to that is Raw data which is stored by data source in the raw data bucket.

11.1 Curated data

The curated data is stored in the data lake, within a designated S3 bucket. The data is separated by each curated data type, as detailed within Table 8 – Curated DataTable 9 – Data Lake Bucket Path. Furthermore, each of the curated data types are partitioned in accordance with the standard path formatting as shown within Table 9 – Data Lake Bucket Path. These bucket path formats provide the ability to query the data using temporal-based criteria, and by desired schema revisions. There are Glue Tables which provide a description of the structure of the data within the data lake for each data type. From the perspective of the Athena Query Editor, the Glue Catalog makes the underlying Parquet data structure appears like a flat, nested (as opposed to relational) SQL table.

There are two variants of Glue tables. The suffix "_latest" indicates that is the most recent table format and data. All other suffixes taking the format "_{schema_revision}" excluding the highest numbered schema_revision indicate some earlier of the table definition and the data that meets that definition. The highest numbered schema_revision and latest are the same table definition and underlying data.

Table 8 – Curated Data

Data Source	Data Type
Ford	Ford
J2735 BSM	Vehicle
J2735 MAP	Map
J2735 SPaT	SPaT
HERE	Traffic
Waze	RoadwayEvent
SunGuide EM	RoadwayEvent
SunGuide TSS	Traffic
SunGuide DMS	Message
SunGuide RWIS	Weather

Table 9 – Data Lake Bucket Path

Data Type	Data Lake Bucket Path	Glue Tables	
Vehicle	bucketstack-curateddatab0f38c51-17w9it4q2mjzf/vehicle/{schema_revision}/year={YYYY}/month={MM}/day={DD}/hour={HH}/minute={mm}/*	vehicle_latest	vehicle_{schema_revision}
Map	bucketstack-curateddatab0f38c51-17w9it4q2mjzf/map/{schema_revision}/year={YYYY}/month={MM}/day={DD}/hour={HH}/minute={mm}/*	map_latest	map_{schema_revision}

Traffic	bucketstack-curateddatab0f38c51-17w9it4q2mjzf/traffic/{schema_revision}/year={YYYY}/month={MM}/day={DD}/hour={HH}/minute={mm}/*	traffic_latest	traffic_{schema_revision}
RoadwayEvent	bucketstack-curateddatab0f38c51-17w9it4q2mjzf/roadwayevent/{schema_revision}/year={YYYY}/month={MM}/day={DD}/hour={HH}/minute={mm}/*	roadwayevent_latest	roadwayevent_{schema_revision}
Spat	bucketstack-curateddatab0f38c51-17w9it4q2mjzf/spat/{schema_revision}/year={YYYY}/month={MM}/day={DD}/hour={HH}/minute={mm}/*	spat_latest	spat_{schema_revision}
Message	bucketstack-curateddatab0f38c51-17w9it4q2mjzf/message/{schema_revision}/year={YYYY}/month={MM}/day={DD}/hour={HH}/minute={mm}/*	message_latest	message_{schema_revision}
Weather	bucketstack-curateddatab0f38c51-17w9it4q2mjzf/weather/{schema_revision}/year={YYYY}/month={MM}/day={DD}/hour={HH}/minute={mm}/*	weather_latest	weather_{schema_revision}

12 Content Definitions

Refer to Table 10 for a list of terms and their associated definitions used throughout this document.

Table 10 – Definitions

Definitions	
3rd Party Interfaces	For the purpose of this document, 3rd party interfaces are meant to describe connections to or from the V2X DEP by data producers and data consumers whether they are FDOT-owned systems or not. In other words, these interfaces are 3rd party to the V2X DEP. The V2X DEP's 3rd party interfaces to FDOT District SunGuide instances are examples of 3rd party interfaces to data producers which are FDOT-owned. The V2X DEP's interface to HERE is an example of a 3rd party interface to a data producer which is not FDOT-owned. Cloud provider interfaces which are used for hosting or internal operation of the V2X DEP.
3rd Party [Docker] Container	For purpose of this document, 3rd party is defined as applications or external entities outside the scope of V2X DEP infrastructure. One instance of a Docker image in an execution environment.
Infrastructure as Code (IaC)	Managing and provisioning data center infrastructure by way of machine-readable definition files rather than through interactive configuration process.
[Docker] Image	An ordered collection of root file system changes (layers) and execution parameters for use within a container at runtime.

Definitions	
Primary Source	A data producer which can make data available to the DEP which *is* the original source of information. For example, if there a device on a private network which measures traffic at a point in the roadway and it is collected by a system like SunGuide, then SunGuide could be the primary source. If the device were instead publicly accessible (as is the case for some camera snapshots), then the device itself is the primary source and SunGuide could be a secondary source of that information.
Secondary Source	A data producer which can make data available to the DEP which *is not* the original source of information. For example, if there a device on a private network which measures traffic at a point in the roadway and it is collected by a system like SunGuide, then SunGuide could be the primary source. If the device were instead publicly accessible (as is the case for some camera snapshots), then the device itself is the primary source and SunGuide could be a secondary source of that information.
Serverless	Managed and scalable services each of which provide a piece of the technology stack (e.g. compute, storage, API integration, etc.) instead of traditional servers configured to run application software. Also, an architecture or component that operates without needing to manage a virtual machine or associated resources. Instead, resources are allocated by the cloud provider to accommodate growing or shrinking resource needs. The application cost is based on resources that are used and no pre-allocation of resources is required.

13 Acronyms

The Glossary section shall be used to define all terms and acronyms required to properly interpret the requirements contained within the requirements document.

Table 11 – Acronyms

Acronym	Definition
AARF	Automated Access Request Form
AD	Active Directory
API	Application Programming Interface
ARBM	All Roads Base Map
ATMS	Advanced Transportation Management System
AWS	Amazon Web Services
AZ	Availability Zone
BSM	Basic Safety Message
CAV	Connected and Autonomous Vehicle
CF	Cloud Formation
CV	Connected Vehicle
CDK	Cloud Development Kit
DEP	Data Exchange Platform
DIVAS	Data Integration and Video Aggregation System
DMS	Dynamic Message Sign
EM	Event Management
ECS	Elastic Container Service
ETL	Extract, Transform, Load
FDOT	Florida Department of Transportation
FTP	File Transfer Protocol
GIS	Geographic Information System
IaC	Infrastructure as Code
IAM	Identity and Access Management
ICD	Interface Control Document
ITS	Intelligent Transportation System
IP	Internet Protocol
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
JWT	JavaScript Web Token
KDA	Kinesis Data Analytics

Acronym	Definition
KDF	Kinesis Data Firehose
KDS	Kinesis Data Streams
KMS	Key Management Service
KPU	Kinesis Processing Unit
MAP	Intersection Map
MS	Microsoft
MSK	Managed Streaming for Apache Kafka
NLB	Network Load Balancer
ODBC	Open Database Connectivity
OBU	On Board Unit
OEM	Original Equipment Manufacturer
PCAP	Packet Capture
PII	Personally Identifiable Information
PSM	Personal Safety Message
PVD	Probe Vehicle Data
RDS	Relational Database Service
RITIS	Regional integrated Transportation Information System
RSM	Road Safety Message
RSU	Road Side Unit
RWIS	Road Weather Information Subsystem
S3	Simple Storage Service
SAA	Software Administration Application
SCP	Secure Copy Protocol
SDK	Software Development Kit
SLA	Service Level Agreement
SMEEd	Safety, Mobility, Environment, and Economic development
SMS	Short Message Service
SNS	Simple Notification Service
SPaT	Signal Phase and Timing
SSL	Secure Sockets Layer
TIM	Traveler Information Message
TLS	Transport Layer Security
TMC	Traffic Management Center
TSM&O	Transportation Systems Management and Operations

Acronym	Definition
TSS	Transportation Sensor Subsystem
TVT	Travel Time Subsystem
UDP	User Datagram Protocol
UI	User Interface
V2I	Vehicle to Infrastructure
V2X	Vehicle to Anything
VPC	Virtual Private Cloud
VPG	Virtual Private Gateway
VPN	Virtual Private Network
WSDOT	Washington State Department of Transportation
WZDx	Work Zone Data Exchange
XML	eXtensible Markup Language