

# **Connected Corridors: I-210 Pilot Integrated Corridor Management System**

## **Core System High-Level Design**

**June 21, 2018  
v 1.1**



Partners for Advanced Transportation Technology works with researchers, practitioners, and industry to implement transportation research and innovation, including products and services that improve the efficiency, safety, and security of the transportation system.

This page left blank  
intentionally

## Table of Contents

<b>1. Introduction</b>	<b>1</b>
1.1. Purpose of Document	1
1.2. Relation to Systems Engineering Process	1
1.3. Intended Audience	2
1.4. Document Organization	2
<b>2. An Introduction to Microservices Architecture on Amazon Web Services</b>	<b>5</b>
2.1. Microservices Definition	6
2.2. Why Use a Microservices Architecture	8
2.3. Use of the Cloud and Amazon Web Services (AWS)	12
2.4. Impact on the Design Document	13
<b>3. System Primary Objectives and Purpose</b>	<b>15</b>
3.1. Project Goals and Objectives	16
3.2. Technical Capabilities Sought	18
<b>4. High Level Design Objectives, Constraints, and Principles</b>	<b>21</b>
<b>5. Core System High Level Design</b>	<b>23</b>
5.1. Major Components	23
5.2. Field Elements	24
5.3. Data Hub	25
5.4. Decision Support	27
5.5. Corridor Management	27
5.6. Primary process flow	30
<b>6. Data Hub Design</b>	<b>33</b>
6.1. Data Sources	36
6.2. Data Pipelines	38
6.2.1. Sensing Data Pipeline	38
6.2.2. Heterogeneous Data Pipeline	39
6.2.3. Homogenous Data Pipeline	40
6.2.4. Pipeline Control	41
6.2.5. Pipeline Status and Logging	43
6.2.6. Corridor Management System-Decision Support System (CMS-DSS)	
Communications Pipeline	44

---

6.3.	External Interface/Data Gateway.....	48
6.4.	Data Hub Command Gateway.....	50
6.4.1.	Conductor.....	51
6.4.2.	Camel.....	52
6.4.3.	ActiveMQ Workflow Status Topic .....	52
6.4.4.	ActiveMQ Workflow Task Topic .....	52
6.4.5.	Monitor .....	52
<b>7.</b>	<b>Decision Support System Design.....</b>	<b>53</b>
7.1.	DSS high level design.....	54
7.2.	DSS Interface .....	55
7.3.	Response Plan Management.....	56
7.4.	Modeling .....	58
7.4.1.	Modeling techniques.....	59
<b>8.</b>	<b>Security Design.....</b>	<b>63</b>
8.1.	Minimize attack surface .....	63
8.2.	Authentication .....	64
8.3.	Data Encryption.....	64
8.4.	Principle of Least Privilege .....	64
8.5.	Automated security and process monitoring.....	65
8.6.	Automate system launch processes.....	65
8.7.	Validate all incoming data .....	65
<b>9.</b>	<b>System Interface and Message System Design.....</b>	<b>67</b>
9.1.	Data Hub Internal Messaging.....	68
9.1.1.	Data Messaging and Kafka .....	68
9.1.2.	Command Messaging and ActiveMQ .....	68
9.2.	DSS Internal Messaging.....	69
<b>10.</b>	<b>Definition of Terms .....</b>	<b>71</b>

## List of Figures

Figure 1-1 – System Requirements Specification within Systems Engineering Process.....	2
Figure 2-1 Data Pipeline Microservice Example .....	6
Figure 5-1 Core System High Level Design .....	23
Figure 5-2 Primary System Incident Flow (Subsystem) .....	31
Figure 6-1 Data Hub High Level Design .....	34
Figure 6-2 Sensing Data Pipeline Design .....	38
Figure 6-3 Heterogeneous Data Pipeline .....	40
Figure 6-4 Homogenous Data Pipeline.....	41
Figure 6-5 Pipeline Primary Control Layer.....	42
Figure 6-6 DSS-CMS Data Pipeline Configurations .....	45
Figure 6-7 Data Hub Data Gateway – ActiveMQ and Web Services Design Patterns .....	49
Figure 6-8 Data Hub Command Gateway.....	51
Figure 7-1 DSS Architecture .....	54
Figure 7-2 DSS Interface High Level Design.....	55
Figure 7-3 Response Plan Management Design.....	56
Figure 7-4 Response Plan Manager Workflow .....	58
Figure 7-5 - Modeling Component Design .....	59

## List of Tables

Table 2-1 Example Component Tasking .....	7
Table 2-2 Microservice Advantage Examples.....	8
Table 2-3 AWS Service Usage .....	12
Table 3-1 – ICM System Goals and Objectives .....	16
Table 5-1 Major System Components .....	23
Table 5-2 – Field Systems .....	24
Table 5-3 Response Plan Lifecycle.....	28
Table 5-4 CMS Management Capabilities .....	29
Table 6-1 ICM Data Sources .....	36
Table 6-2 Sensing Pipeline Data Sources.....	38

## 1. INTRODUCTION

This Connected Corridors High Level Design document provides the high level system architecture for the system to be deployed on the I-210 corridor. The system architecture described here is a direct result of the Connected Corridors System Requirements document and the work done at UC Berkeley in traffic modeling and control. This document provides the system architecture, high level design of the primary subsystems, the decisions, assumptions, constraints, and reasoning behind that architecture, and critical functions each subsystem provides for the system.

The system, to be piloted along a section of the I-210 corridor in the San Gabriel Valley area of Los Angeles County, aims to improve overall corridor performance during incidents, unscheduled events, and planned events. This is to be achieved by more efficiently managing existing systems and infrastructures, promoting cross-jurisdictional operations, and using multi-modal traffic and demand management strategies that consider all relevant modes of transportation.

### 1.1. PURPOSE OF DOCUMENT

This document provides the high level design, serving as the identification of the primary subsystems and major components as well as the basis for the selection, development, and integration of these into a system that satisfies the system requirements as defined in the Systems Requirements Document. This high level design will govern the technology platform and direction of the I-210 Pilot ICM System and serve as the basis for other Caltrans-led ICM efforts statewide.

### 1.2. RELATION TO SYSTEMS ENGINEERING PROCESS

The development of high level design is part of the systems engineering process that the Federal Highway Administration (FHWA) requires be followed for developing Intelligent Transportation System (ITS) projects when federal funds are involved. While not required for projects only using state or local funds, use of the systems engineering process is still encouraged in such cases.

The overall systems engineering process is illustrated in Figure 1-1. Developing high level design represents the next step of the System Definition and Design phase of a project (Phase 2 in the figure) following the completion of the System Requirements. High Level Design is typically derived from the requirements. The resulting design elements are in turn used to inform and guide the more detailed design of the various system and subsystem components.

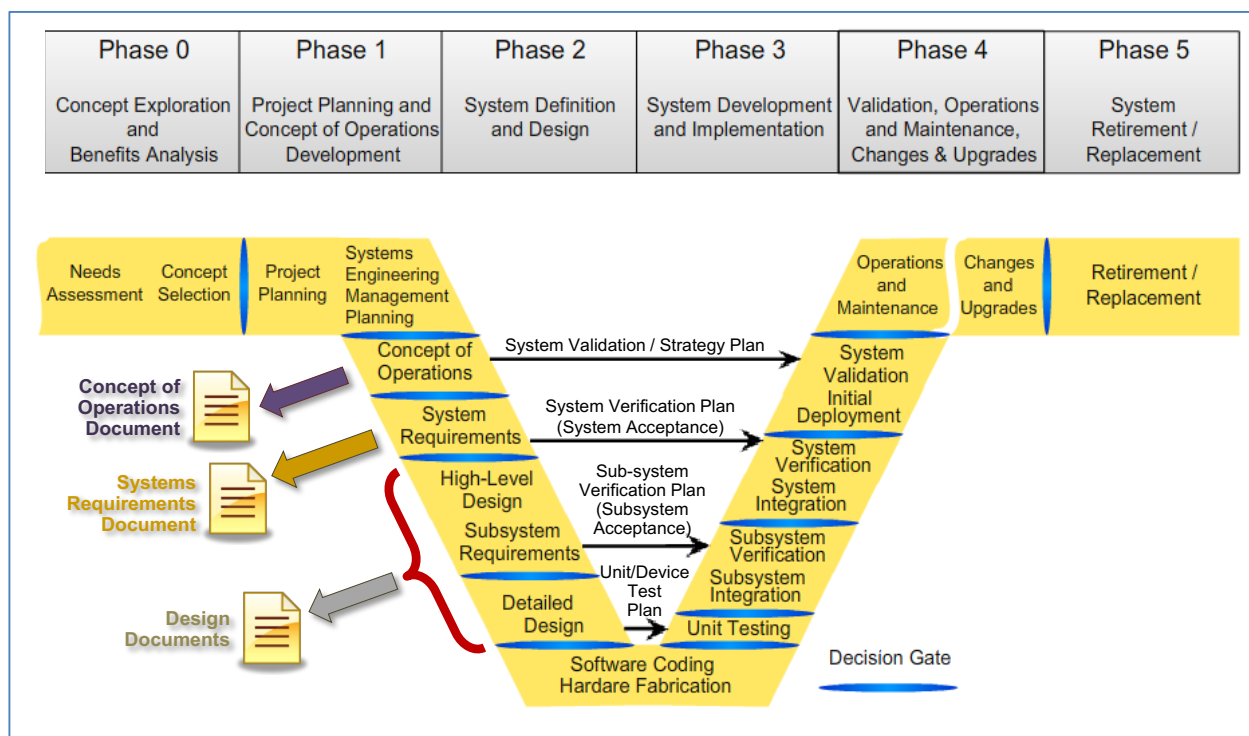


Figure 1-1 – System Requirements Specification within Systems Engineering Process

### 1.3. INTENDED AUDIENCE

The primary audience for the System High Level Design document includes personnel responsible for designing and implementing the ICM system. The audience also includes individuals from Caltrans District 7, Caltrans Headquarters, and the University of California, Berkeley, tasked with project management duties.

### 1.4. DOCUMENT ORGANIZATION

The remainder of this document is organized as follows:

- **Section 2** provides a high level overview of micro-services architectures and the use of Amazon Web Services (AWS) used in the design of this system
- **Section 3** summarizes the primary system objectives identified within the System Requirements that shape the system design.
- **Section 4** presents the primary guiding design principles and base assumptions that shape the system design.
- **Section 5** presents the key system design components and primary data flows.



- **Section 6** presents the key system components of the Data Hub including data sources, interfaces/gateways and pipelines.
- **Section 7** presents the key system components of the Decision Support System (DSS) including the rules engine, modeling interfaces, and response plan generation.
- **Section 8** presents key security design issues and implementation plans.
- **Section 9** provides design information for the system interfaces and the messaging systems, describing how information is exchanged with external systems and how it is passed between and within subsystems.

In addition, other supporting documents are available in the Document Library of the Connected Corridors website at <https://connected-corridors.berkeley.edu/resources/document-library>:

This page left blank  
intentionally

## 2. AN INTRODUCTION TO MICROSERVICES ARCHITECTURE ON AMAZON WEB SERVICES

The Connected Corridors system software design is not based on architectures and design patterns typically found in the transportation industry. Many of today's transportation systems have long production histories with significant operational experience, but as a result are based on system architectures and code that have been in existence for a decade or longer. Current transportation systems are often based on a more traditional n-tier, data center hosted software architecture with a user interface layer, application layer, and relational database layer. Such traditional designs are well suited for systems with moderate data volumes and limited size and scope.

The Connected Corridors program has begun with a blank slate, and as a result is not bound to the limitations of an existing system. Instead, Connected Corridors uses a more recent software architecture and associated design patterns more often found in the big data world, more suited for high data volumes and real-time processing at extremely large scale. The design of the I-210 system is built specifically for multi-jurisdictional environments, large data volumes, and large geographic areas, coordinating large numbers of transportation elements. It is specifically designed for a future of connected vehicles and infrastructure with the data volumes and processing requirements that will be present in that future.

To do this, the system makes use of two key design elements:

- A microservices architecture
- Cloud technology and design (specifically Amazon Web Services)

These two key design elements are specifically designed to be very responsive to both immediate and long term demands on the system. They provide a very agile system that can scale on demand to react to increases in demand for processing, such as responding to multiple traffic incidents requiring high demands on the Connected Corridors predictive modeling components. This agility also provides long term benefits, allowing the system to more easily scale to additional corridors or larger geographic areas as well as increases in data volumes that can be expected with new data sources, such as connected and automated vehicles. Using microservices and cloud technology together means that additional server and computational resources can be applied on demand, with the microservices architecture making the software responsive, and cloud technology providing the resources to the software to make that possible.

As a result, this document does not provide information regarding the infrastructure design (such as servers or data center requirements). There are no on-premise software or hardware systems to specify or purchase. Hardware specifications can be altered on demand based on system load and configuration during system operation and are not fixed for the operating life of the system.

This section will provide some basic information and explanation of these two technologies and how they work together to provide significant benefits to the program. This is provided to assist in understanding the remaining sections of this document and the design choices made in this system architecture.

## 2.1. MICROSERVICES DEFINITION

Microservices is a term that is used to describe many different designs, but in general, all microservice architectures have the following elements in common:

- Self-contained, autonomous software components that each provide a specific service or function (independent services)
- Loosely coupling of a suite of such services to provide one or more system capabilities
- Well defined, lightweight communications (APIs) between services over network connections

In the ICM system, this is the primary architectural pattern within the data hub and DSS, and the communications between the DSS, data hub, and CMS are also patterned on this design. This architectural pattern is often coupled with automated deployment, configuration, security, and monitoring capabilities.

In both the DSS and data hub, the system is built with individual services, each deployed separately. Each service has a very specific responsibility within the system. The individual services are connected using one or more messaging systems (Kafka or ActiveMQ). The communication between those services is defined by a contract, generally the Transportation Management Data Dictionary (TMDD) with some modifications required to add additional information and ensure interchangeability between different services (CMS vendors and TMCs).

For example, the data hub uses a design paradigm of a data pipeline. A typical data pipeline for high volume data looks as follows:

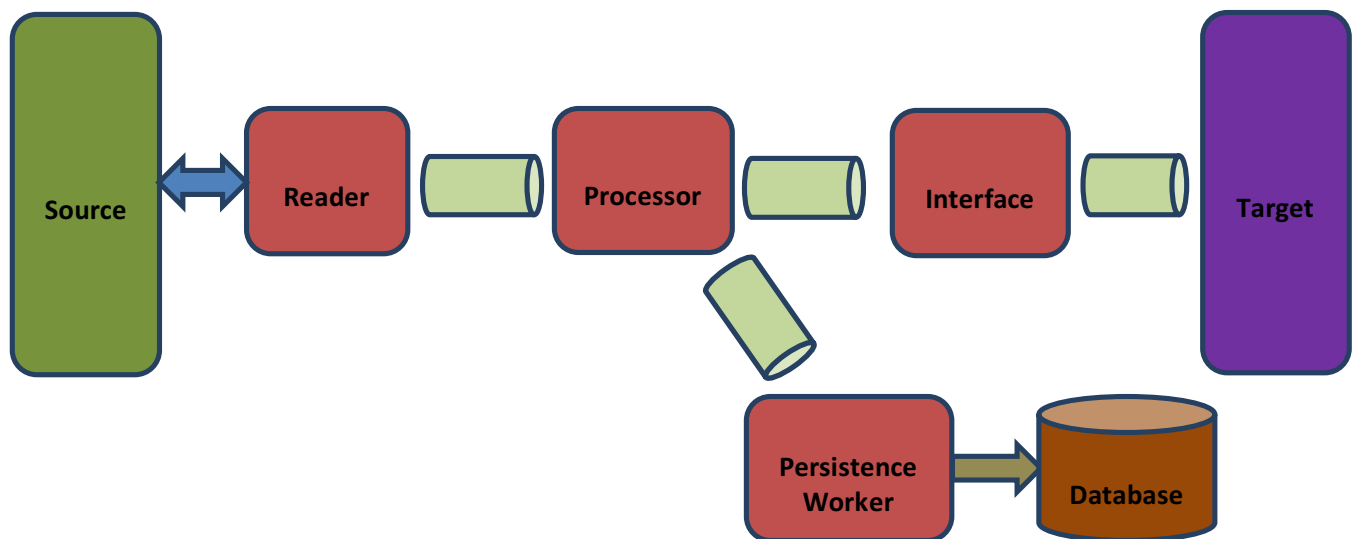


Figure 2-1 Data Pipeline Microservice Example

In Figure 2-1 the source in green is typically an external TMC, and the target in purple is either the DSS or CMS system. The data hub components are those components placed between the source and target.

The reader, processor, interface, and persistence workers are all individual services with a specific, independent task. The light green “pipes” between the services represent the messaging system used to transport messages containing data between the services. The component tasking breakdown is as follows:

**Table 2-1 Example Component Tasking**

Component	Task
<b>Source</b>	External component – not a system component. Source of data elements.
<b>Reader</b>	Maintain SOAP based TMDD conversation with source to collect data. Place data in TMDD structured message in messaging system.
<b>Processor</b>	Collect data from processing system and perform desired processing. May include quality checks, transformation, predictive analysis or other type of processing.
<b>Interface</b>	Receive data from messaging system and present to CMS or DSS. Transform data as necessary for target.
<b>Messaging System Pipe</b>	The data hub messaging system (Apache Kafka) used to provide communication between the individual services.
<b>Target</b>	Not a data hub component. Target may be CMS or DSS.
<b>Persistence Worker</b>	Receive data from the messaging system. Save data in the database. Retrieve data from the database when requested and place data on the messaging system.
<b>Database</b>	Store data.

The readers maintain a SOAP based TMDD conversation with the source and place the data received, in a TMDD structured message on a message topic (in light green). The processor, receiving the data messages off of the message topic, does any type of processing desired such as a quality check, transformation, or other process, and places its results on another message topic. Multiple processors may be used in serial or parallel to provide the desired level of granularity of the services. The interface service reads the results and provides an interface where the target can connect and receive the processed data results. A parallel path from the processor to the persistence worker allows the persistence worker service to also read the processed results and store those results in a database.

Each of the components in red are independent, autonomous services. Each has a specific function and is independent of the other services with a specific desired input and a specific output. By combining these together in different configurations via a lightweight messaging protocol (loose coupling) and a defined API such as TMDD, a specific application purpose is provided, namely the processing, quality

---

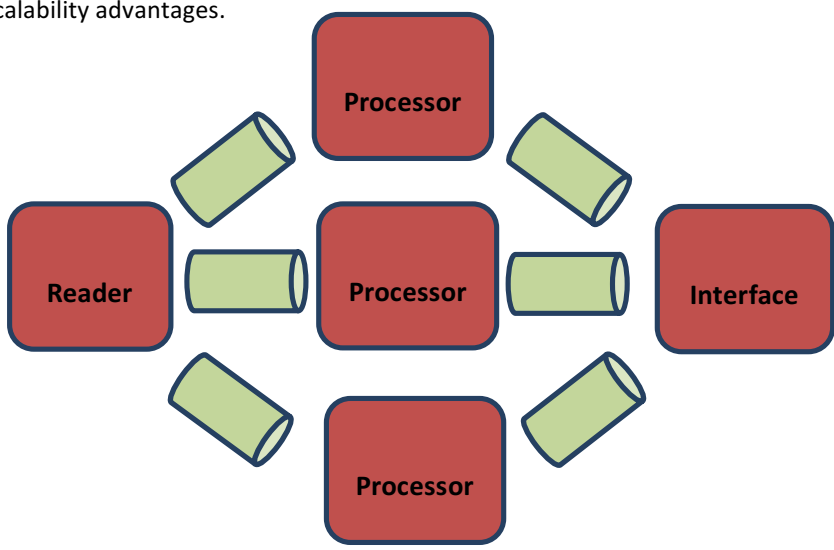
verification, and storage of data from external sources and making the data available to the CMS and DSS.

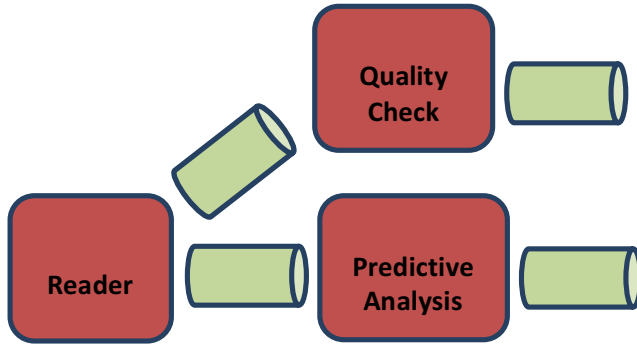
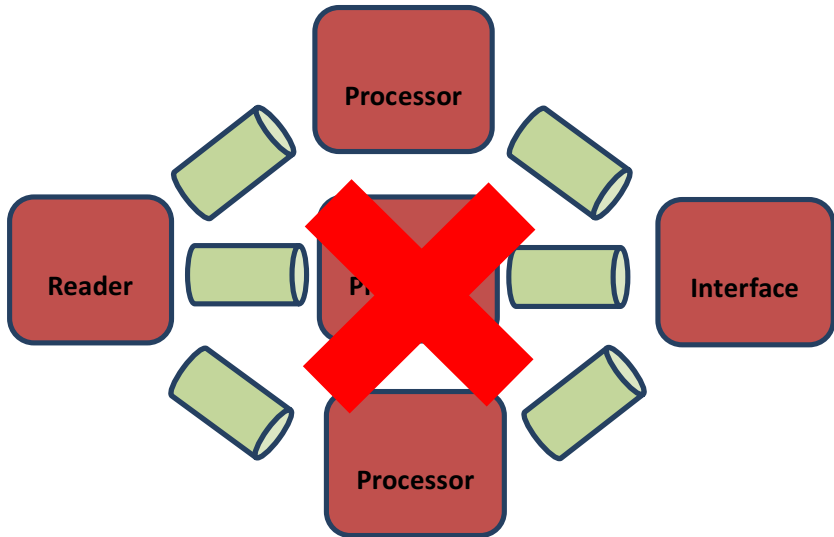
## 2.2. WHY USE A MICROSERVICES ARCHITECTURE

Using a microservices architecture provides several advantages. In general, they provide high levels of scalability, reliability, resilience to failure, parallelization, speed of processing, ability to adapt, and very high data throughput capabilities.

By separating each of the system tasks into separate services, and connecting them by messaging, some significant benefits are realized. Using the example in Figure 2-1, here are ways in which these benefits are realized:

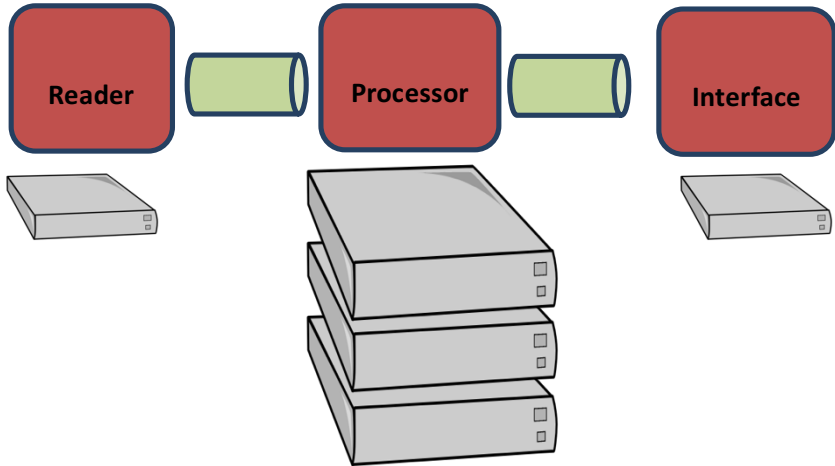
**Table 2-2 Microservice Advantage Examples**

Advantage	Method of Realization
<b>Scalability</b>	<p>Work can be parallelized as load on the system increases, either locally or for an entire pipeline. For example, if a specific task requires significant processing resources, multiple processors can be used in parallel to share the processing load. This provides significant scalability advantages.</p> 

Advantage	Method of Realization
<b>Speed of Processing</b>	<p>Work can be branched to complete separate tasking. For example, a processor for predictive analysis and a second processor for a quality check can be split into two separate paths, with independent processors for each task. This provides significant speed of processing advantages.</p>  <p>The diagram shows a linear sequence of three red rounded rectangles: 'Reader', 'Predictive Analysis', and 'Quality Check'. Between 'Reader' and 'Predictive Analysis', and between 'Predictive Analysis' and 'Quality Check', there are two parallel green cylinders representing buffers or parallel processing paths. Additionally, a single green cylinder is positioned between 'Reader' and 'Quality Check', representing a direct path or a parallel task.</p>
<b>Reliability and Resiliency</b>	<p>Failure of a single task instance may result in degraded performance for a single pipeline, but will not affect other system processes. Using multiple parallel instances of a task can ensure that even during failure, a process can continue to function. Even with a single instance of the task, the restart of a new instance will ensure that the pipeline will recover from the failure, usually within minutes. This provides significant reliability and resiliency advantages.</p>  <p>The diagram shows a circular arrangement of components. On the left is a red rounded rectangle labeled 'Reader'. On the right is a red rounded rectangle labeled 'Interface'. In the center, there are three red rounded rectangles labeled 'Processor'. The middle 'Processor' is crossed out with a large red 'X'. Green cylinders, representing data or buffers, are positioned between the 'Reader' and the 'Processors', and between the 'Processors' and the 'Interface', indicating a circular flow through multiple parallel processing paths.</p>

Advantage	Method of Realization
<b>Ability to Adapt - Incremental Upgrades and Improvements</b>	<p>A single process task can be upgraded or replaced with an alternative without affecting the other system processes. Only the process to be upgraded or replaced need be affected in a system upgrade deployment. With proper procedures, upgrades or replacements can be achieved without interruption to system operation. In the example below, a source system may be upgraded with the only impact on the system being the replacement of the reader instance. The new reader instance could be brought up while the old reader and source continues to operate. When the new reader and source are ready, the old reader is terminated and the new reader is allowed to communicate with the messaging system. New system capabilities can be added simply by adding the new service or services to the existing system without changing the current processing. This provides significant advantages for the ability to provide incremental improvements with continuous operation.</p> <pre>graph LR; subgraph Old_System [Old System]; OS[Old Source] --&gt; OR[Old Reader]; end; subgraph New_System [New System]; NS[New Source] &lt;--&gt; NR[New Reader]; NR --- MS1(( )); MS1 --- P[Processor]; P --- MS2(( )); MS2 --- I[Interface]; end; OR --- MS1; style Old_System stroke-dasharray: 5 5; style New_System fill:#fff,stroke:#000; style OS fill:#6aa84f,color:#fff; style OR fill:#c0392b,color:#fff; style NS fill:#6aa84f,color:#fff; style NR fill:#c0392b,color:#fff; style P fill:#c0392b,color:#fff; style I fill:#c0392b,color:#fff; style MS1 fill:#768d4d,stroke:#000; style MS2 fill:#768d4d,stroke:#000;</pre>



Advantage	Method of Realization
<b>Optimization and Cost Efficiency</b>	<p>Hardware requirements can be tuned to the specific needs of each process. For instance, a predictive analysis may require significant CPU and/or memory requirements, whereas a reader or interface require much smaller CPU and memory requirements. In the ICM system design, sensing data is processed using an Apache Spark cluster running on a cluster of several AWS EC2 instances. Readers are run on much smaller EC2 instances and the data hub interface is run on medium sized AWS EC2 instances. The hardware is sized for the individual process requirements. This provides significant efficiency and cost advantages.</p>  <pre>graph LR; Reader[Reader] --- Conn1(( )) --- Processor[Processor]; Conn1 --- Conn2(( )) --- Interface[Interface]; subgraph Hardware; Reader --- S1[Server]; Processor --- S2[Stack of 4 Servers]; Interface --- S3[Server]; end</pre>

## 2.3. USE OF THE CLOUD AND AMAZON WEB SERVICES (AWS)

Using the cloud, and in the I-210 corridor, specifically Amazon Web Services (AWS), enables many of the capabilities inherent in the microservices architecture. There are additional benefits to using AWS, but this section will focus on those services and benefits that are specific to the microservices architecture implementation in the I-210 ICM program.

Primary AWS services and capabilities used to achieve the advantages of a microservices architecture are detailed below.

**Table 2-3 AWS Service Usage**

AWS Service	Definition and Uses
<b>Elastic Cloud Compute (EC2)</b>	<p>Amazon's EC2 service provides resizable compute capacity on demand. The I-210 project uses EC2 to provide server resources for the different services within the microservices design. Using EC2 services allows:</p> <p>Scalability – addition or removal of EC2 resources based on the load experienced by the system</p> <p>Speed of processing – ability to parallelize work across multiple EC2 instances and size EC2 instances according to demand</p> <p>Incremental upgrades and improvements – ability to start and stop EC2 instances with different versions of software and hardware as needed</p> <p>Resilience and Reliability – ability to replace failed instances upon demand</p> <p>Optimization and Efficiency – AWS provides a selection of EC2 compute instances, allowing variation of CPU, memory, storage, and networking options depending upon system need.</p>

AWS Service	Definition and Uses
<b>CloudFormation</b>	<p>Amazon's CloudFormation service provides the ability to provision and initialize the I-210 systems environment, including networking, security, EC2, databases, virtual private cloud (VPC), and other environment resources. This service is key to:</p> <p>Scalability – Cloudformation is used to ensure resources are brought up in a consistent manner, ensuring configuration management, security, and consistent configuration.</p> <p>Incremental upgrades and improvements – CloudFormation is used to manage and execute the deployment of upgrades and improvements.</p> <p>Resilience and Reliability – CloudFormation is used to ensure proper system configuration management and reliable, repeatable deployment of system components.</p> <p>Optimization and Efficiency – CloudFormation provides automated deployment capabilities minimizing manual human intervention in the deployment process.</p> <p>Security – CloudFormation allows completely automated deployment of system components, minimizing human error in system configuration and consistent management of system and security configuration.</p>
<b>Relational Database Service (RDS) and Elastic Map Reduce (EMR)</b>	<p>RDS and EMR are managed services provided by AWS for the Postgres database and Apache Spark. These services minimize the system administration required for system operations and maintenance. The specific application code and database schemas are still the responsibility of the I-210 program, but the purchase, deployment, and management of the underlying hardware and Postgres and Spark software is managed by AWS.</p>

## 2.4. IMPACT ON THE DESIGN DOCUMENT

As a result of the flexibility of the microservices architecture and AWS services, the High Level Design specification may be different than what is typical for other projects involving complex software implementations. The design itself is significantly different, requiring both a description of the design patterns themselves as well as their implementation. Additionally, this document does not provide information regarding the infrastructure design (such as servers or data center requirements). There are no on-premise software or hardware systems to specify or purchase. Hardware specifications can be

---

altered on demand based on system load and configuration during system operation and are not fixed for the operating life of the system.

The I-210 pilot is a proof of concept, and the I-210 ICM system is designed for incremental change and improvement, with new capabilities being added and existing functions improved. It is designed so that these changes can occur during the pilot period and throughout the service life of the system. While the breakdown and separation of services will be defined in future design documents, it is intended that these breakdowns, and the way in which they are connected can be changed over time, resulting in a very dynamic and adaptable system.

This document focuses on the system and component high level designs and the high level design and architecture patterns used within the system.

### 3. SYSTEM PRIMARY OBJECTIVES AND PURPOSE

The overriding purpose of the I-210 Pilot is to reduce congestion and improve mobility along a section of the I-210 corridor in Los Angeles County through the coordinated management of its major networks: the I-210 freeway, key surrounding arterials, and local and regional transit services. The goal is to enable all corridor “actors”—transportation system managers and operators, control systems, vehicles, and travelers—to work together in an efficient and coordinated way.

These improvements will be achieved by developing and deploying the ICM system described in this high level design document. At the heart of the proposed system will be a Decision Support System (DSS) designed to help corridor system operators manage incidents, unscheduled events, and planned events more effectively. This system will use information gathered from monitoring systems and provided by predictive analytical tools to estimate current and near-future operational performance. The information will be used to develop recommended courses of action to address problems caused by identified incidents and events. More specifically, this system is expected to:

- Improve real-time monitoring of travel conditions within the corridor
- Enable operators to better characterize travel patterns within the corridor and across systems
- Provide predictive traffic and system performance capabilities
- Be able to evaluate alternative system management strategies and recommend desired courses of action in response to planned events, unscheduled events, and incidents
- Improve decision-making by transportation system managers
- Improve collaboration among agencies operating transportation systems in the corridor
- Improve the utilization of existing infrastructure and systems
- More efficiently use spare capacity to address non-recurring congestion
- Reduce delays and travel times along freeways and arterials
- Improve travel time reliability
- Help reduce the number of accidents occurring along the corridor
- Reduce the period during which the congestion resulting from an incident or event affects corridor operations
- Reduce greenhouse gas emissions
- Generate higher traveler satisfaction rates
- Increase the overall livability of communities in and around the I-210 corridor

While development of the proposed system is under the financial sponsorship of Caltrans Headquarters, the system will be developed with cooperation of the local transportation agencies that have agreed to participate in its operation, in coordination with PATH. Project activities will include the design, development, installation, testing, and operation of various components of the ICM system, as well as the development of interfaces with existing monitoring and control systems. For example, the ICM Core System will be interfaced with traffic management systems owned by Caltrans, such as the Advanced Traffic Management System (ATMS).

It is important to understand that rather than having a system delivered that meets all of the requirements in the system requirements specification on its first day of operation, the system is being developed in an iterative fashion, allowing learning and feedback between operation and

design/implementation. As a result, this design attempts to accommodate all of the various requirements by providing a scalable, flexible, and extendable platform that can address these requirements in various ways based on the pilot's experience. It is also expected that this approach will allow Caltrans and local agencies to address new discoveries and ideas that may be informed by future corridor operations and activities.

### 3.1. PROJECT GOALS AND OBJECTIVES

The primary goal of the I-210 Pilot ICM project is to improve overall corridor performance along a section of the I-210 corridor. This translates into the following specific goals:

1. Improve operational situational awareness
2. Promote collaboration among corridor stakeholders
3. Improve response to incidents and events
4. Improve travel reliability
5. Improve overall corridor mobility
6. Empower travelers to make informed travel decisions
7. Facilitate multi-modal movements across the region
8. Promote transportation sustainability by reducing impacts on the environment
9. Improve corridor safety

For each of these goals, Table 3-1 further identifies the main operational objectives. Many of the objectives are similar to those of traditional transportation improvement projects. Many, however, also focus on implementing more comprehensive travel and system status monitoring systems, improved operational forecasting, improved information dissemination to travelers, enhanced data-sharing capabilities, demand management approaches, and improved collaboration among transportation system operators.

**Table 3-1 – ICM System Goals and Objectives**

Goals	Objectives
<b>1. Improve situational awareness</b>	<ul style="list-style-type: none"> <li>• Establish minimum requirements for data collection to support system management</li> <li>• Increase data collection opportunities from arterials and local roads</li> <li>• Improve the collection of real-time operational data from non-traditional sources, such as probe vehicles</li> <li>• Develop a comprehensive corridor informational database covering all relevant travel modes within the corridor</li> <li>• Improve the quality, accuracy, and validation process of collected data</li> <li>• Increase the ability to estimate travel demand patterns in a multi-modal environment</li> <li>• Improve the ability to forecast near-future travel conditions based on known incidents, road conditions, weather, and local events</li> <li>• Develop performance metrics considering all available travel modes</li> </ul>
<b>2. Promote collaboration among corridor stakeholders</b>	<ul style="list-style-type: none"> <li>• Strengthen existing communication channels among the corridor's institutional stakeholders</li> <li>• Explore opportunities for new communication links between corridor stakeholders</li> <li>• Improve cooperation and collaboration among corridor stakeholders</li> <li>• Develop regional/joint operations concepts</li> </ul>

Goals	Objectives
	<ul style="list-style-type: none"> <li>• Identify new methods of collaboration</li> <li>• Extend corridor performance metrics to the network level</li> <li>• Investigate new types of agreements between participating agencies</li> </ul>
<b>3. Improve response to incidents and unexpected events</b>	<ul style="list-style-type: none"> <li>• Reduce the time needed to identify the existence of an incident or unexpected event</li> <li>• Reduce the time needed to respond to incidents or unscheduled events</li> <li>• Enhance the coordination of activities among first responders, traffic management agencies, and transit agencies to minimize impacts on system operations</li> <li>• Reduce the time needed to implement control actions to address congestion resulting from an incident or event</li> <li>• Reduce the time needed to disseminate recommended detours around an incident or event</li> </ul>
<b>4. Improve travel reliability</b>	<ul style="list-style-type: none"> <li>• Improve travel time predictability along the corridor</li> <li>• Reduce the impacts of incidents and events on network operations</li> <li>• Improve incident/event notification for first responders and network operators</li> <li>• Improve incident/event notification to travelers and fleet operators</li> <li>• Provide travelers and commercial vehicle operators affected by an incident or event an enhanced ability to seek alternate routes or mode of transportation</li> </ul>
<b>5. Improve overall corridor mobility</b>	<ul style="list-style-type: none"> <li>• Reduce delays incurred by travelers</li> <li>• Reduce the impacts of incidents and events on network operations</li> <li>• Efficiently use spare capacity along corridor roadways to plan necessary detours around incidents or events</li> <li>• Promote strategies to induce desirable travel demand patterns</li> <li>• Coordinate the management of freeway and arterial bottlenecks</li> <li>• Promote increases in vehicle occupancy</li> <li>• Promote increases in transit ridership</li> </ul>
<b>6. Empower system users to make informed travel decisions</b>	<ul style="list-style-type: none"> <li>• Improve the dissemination of real-time, multi-modal travel information</li> <li>• Enhance the use of infrastructure-based informational devices (freeway CMS, arterial trailblazer signs, kiosks, etc.) to provide en-route information to travelers</li> <li>• Enable individuals to receive travel information on connected mobile devices</li> <li>• Make archived historical data available to 511 services and information service providers</li> <li>• Support the dissemination of travel information by 511 services and third-party providers</li> </ul>
<b>7. Facilitate regional multi-modal movements</b>	<ul style="list-style-type: none"> <li>• Promote the integration of commuter rail and bus services with corridor operations</li> <li>• Facilitate transfers across modes during incidents and events</li> <li>• Provide relevant regional travel information to travelers</li> <li>• Direct travelers to park-and-ride facilities with available spaces</li> </ul>
<b>8. Promote transportation sustainability</b>	<ul style="list-style-type: none"> <li>• Reduce fuel consumption</li> <li>• Reduce vehicle emissions</li> <li>• Identify financially sustainable solutions for long-term system operations and maintenance</li> <li>• Encourage the use of transit, walking, and bicycling where appropriate</li> <li>• Support locally preferred alternatives compatible with corridor objectives</li> <li>• Develop and implement performance metrics reflecting environmental goals</li> </ul>

Goals	Objectives
<b>9. Improve corridor safety</b>	<ul style="list-style-type: none"> <li>• Reduce collision rates</li> <li>• Reduce the severity of collisions</li> <li>• Reduce the number of fatalities</li> <li>• Reduce the impacts of primary and secondary incidents on network operations through improved incident management</li> <li>• Improve safety for bicycles, pedestrians, and transit</li> </ul>

### 3.2. TECHNICAL CAPABILITIES SOUGHT

To help manage travel activities within the corridor during incidents, unscheduled events, and planned events, the project is seeking the following technical capabilities to support the goals and objectives identified in Section 3.1:

- Gather and archive information characterizing traffic operations, transit operations, and the operational status of relevant control devices within the I-210 corridor.
- Identify unusual travel conditions on the I-210 freeway or nearby arterials based on monitoring data provided by various traffic, transit, and travel monitoring systems.
- Identify situations in which an incident on roadways or transit facilities significantly affects travel conditions within the corridor.
- Provide corridor-wide operational evaluations to traffic managers, transit dispatchers, and other relevant system managers, including projected assessments of near-future system operations under current and alternate control scenarios.
- Identify recommended detours around incidents or routes leading to the site of an event, considering observed travel conditions within the corridor. Depending on the need, and final system capabilities, specific detours may be recommended for motorists and transit vehicles.
- Identify recommended signal timing plans to use at signalized intersections to improve and/or accommodate traffic flow influx during incidents and events and improve overall corridor mobility.
- Identify recommended ramp metering rates to use on individual I-210 freeway on-ramps and connectors to maintain overall corridor mobility.
- Identify messages to post on available freeway and arterial CMSs to inform motorists of incidents and events.
- Provide guidance to motorists on the I-210 freeway and surrounding arterials using available freeway CMSs, arterial CMSs, and arterial dynamic trailblazer signs regarding which detour to take to go around an incident or which route to follow to reach the site of an event.
- Provide information to motorists about the availability of parking and transit services to help travelers make alternate mode-choice decisions.



- Provide uniform traffic management strategies across jurisdictional boundaries during incidents and events.
- Provide information to motorists through third-party outlets, such as 511 services, navigation application providers, etc.

This page left blank  
intentionally

## 4. HIGH LEVEL DESIGN OBJECTIVES, CONSTRAINTS, AND PRINCIPLES

The core system design is governed by a few key primary design objectives dictated by the project and the requirements:

- Real time operation – The system must operate in a near real time environment. The time between an incident occurring and a response plan implementation is a critical time period. Response times should be dictated primarily by the time to notification and confirmation of the incident, and the time to fully implement a decision. The time required by the system to process information should be minimized so as to have the maximum positive impact on corridor operations.
- Speed to decision – There is a significant amount of data processing required to maintain both operator awareness and to ensure the modeling within the decision support system is updated with the latest available information. Data processing and decision processing time should be minimized.
- Quality of decision – The quality of the response plans is a direct result of the traffic estimation quality, traffic prediction quality, data processing time, data quality, and rules used to generate response plans.
- Ability to measure outcomes – There must be a high level of confidence in the system outcomes, and these outcomes must be continuously measured and monitored with processes in place to improve the systems effectiveness over time.
- Incremental deployment – The system must be able to be deployed incrementally, adding new capabilities over time.
- System flexibility – The system must be built to be flexible, as it is intended as a pilot that is adjusted and modified as experience is gained with operations. It is intended that the system will also change as new capabilities are added. In addition, this flexibility will be key to future implementations in other corridors with different integration needs and different requirements. It is also expected that transportation itself will be changing radically over the lifetime of the system, so flexibility is key to its long term viability.
- Secure operations – As the system has the capability to request changes to traffic controls across a large, complex urban corridor, security of the system is critical to its success.

Core system design is limited by the following constraints:

- Ability to be operated and maintained by Caltrans without significant licensing costs – The system must be designed with open source components as much as possible. It is not desired to have significant licensing costs required for subsequent deployments for future corridors or over long periods of time. It is intended that Caltrans will be capable of deploying, maintaining, and operating this and future deployments.
- Limited time to deploy – The deployment schedule is aggressive, creating the need to ensure significant reusability within the design so that multiple data sources can be incorporated into the system without designing new data pipelines for each one, utilizing a common set of design patterns and system libraries for new pipelines.

- Constrained development resources – Caltrans provided a fixed amount of funding so the system design must not exceed our resources with complexity and additional features.

These core design objectives and constraints are implemented with the following design principles:

- Cloud development, deployment, and operations – In order to achieve maximum flexibility in both system development and future configurations, minimize resource utilization in development and deployment, and minimize deployment time, the core system is designed for 100% cloud operations within an Amazon AWS cloud environment.
- The system is a pilot system and it expected that as we learn from the deployment of the system changes will be needed and recommended.
- Caltrans shall be capable of supporting the system with its internal resources.
- Decisions produced in the form of response plans shall be based on current corridor information with limited delays in information processing.
- Data maintained within the core system will be limited to the data required for system operation. Long term data storage shall be a function of PeMS.
- The system will be designed for future growth, incremental improvements, future transportation system changes, changes in transportation itself, geographic changes, and new and updated information sources and needs.
- Ease of deployment
- Ease of maintenance
- Portability of the system design and components to other corridors
- Highly decoupled design for flexibility, scalability, redundancy, and reliability
- Use available data standards whenever possible
- Standardized external interfaces for ease of portability and inclusion of new subsystems
- Maintenance of a separation of concerns between data management, decision support, and control functions.
- Design with state, regional, and local targeted components for future scalability and standardized deployment across the state.

## 5. CORE SYSTEM HIGH LEVEL DESIGN

The core system high level design is illustrated in Figure 5-1. This high level design provides clear separation between external systems providing data and receiving control requests (green), the data hub receiving and processing information from those external systems (red), decision support providing response plans for incidents (blue), and the corridor management system providing user interface, response plan selection and approval, and sending of control requests to external systems (purple).

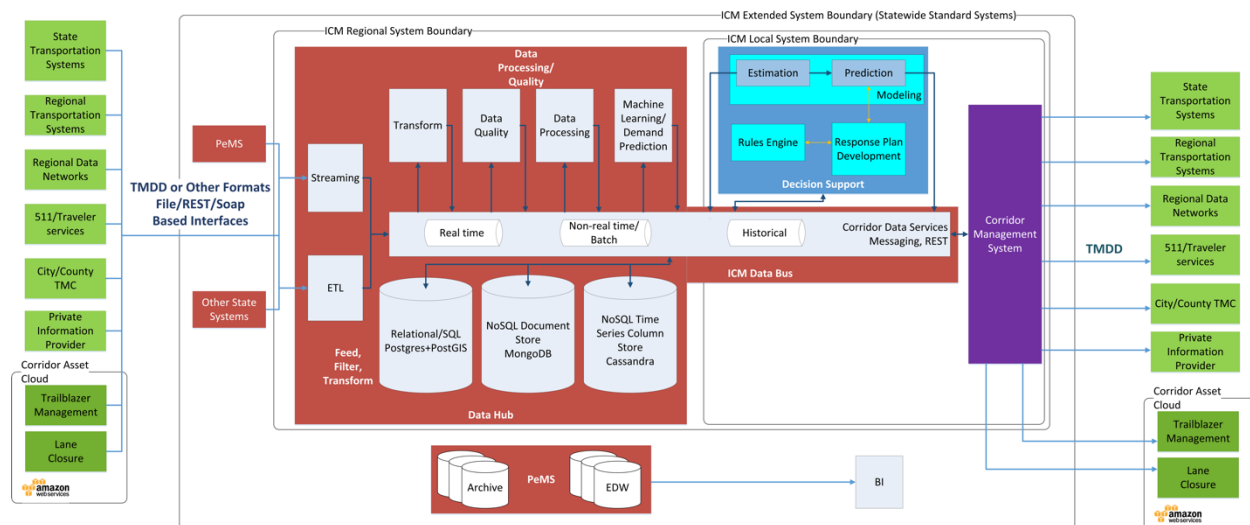


Figure 5-1 Core System High Level Design

### 5.1. MAJOR COMPONENTS

As shown in Figure 5-1, the ICM Core System is composed of three major subsystems: the Data Hub, the Decision Support System, and the Corridor Management subsystem. These subsystems, plus the external field elements the Core System interacts with, are summarized in the following table (color-coded to match Figure 5-1) and described in detail in the following pages.

Table 5-1 Major System Components

Component	Description
<b>Data Hub</b>	Provides for receipt and processing of all corridor data and a method of communication among the three subsystems, using a common set of data definitions. Provides operational data persistence and retrieval.
<b>Decision Support</b>	Provides current traffic state, response plan development, and response plan performance prediction to provide one or more recommended response plans for incidents and events.
<b>Corridor Management</b>	Provides the primary user interface for system operators, a method for users to monitor current corridor and corridor asset states, interact with the Decision

	Support System, review, evaluate, select, and approve response plans, and interact with external systems, particularly for execution and management of response plans.
<b>Field Elements</b>	Represent external providers and consumers of information and requests for action such as intersection signals, ramp meters, corridor sensing, and other elements, usually through respective transportation management systems and TMCs.

The design for the I-210 Pilot will not have a layered geographic approach, but the design is intended to allow such an approach in the future. There are three geographic layers to the design: state, regional, and local. The intent is that the Data Hub operates on a regional level (Caltrans district), with the potential for serving multiple corridors within a region. Data would be consolidated and aggregated at a state level for multiple regions via PeMS, to support larger scale archiving, business intelligence analysis, and continued improvement of corridor and ICM system capabilities at the state level. The Decision Support and Corridor Management components operate on a local level, within a single corridor. With state consolidation of data, regional data communication, and local decision support and response plan execution, a method of sharing information between corridors is enabled, while ensuring local jurisdiction control of traffic event and incident response.

## 5.2. FIELD ELEMENTS

Green elements in Figure 5-1 represent the various corridor field data sources and field element controls, including various state, regional, and local transportation systems, regional transportation data networks, and private information providers.

- On the left side of the diagram, these elements represent the various data sources for information consumed and processed by the ICM system.
- On the right side, these represent the various control interfaces to execute response plan elements selected by operators of the ICM system.

The systems that will be providing the data within the I-210 corridor include:

**Table 5-2 – Field Systems**

Source	Information Type
Pasadena TMC*	Pasadena intersection signals
Arcadia TMC*	Arcadia intersection signals
County TMC - KITS*	LA County intersection signals
	Duarte intersection signals
	Monrovia intersection signals
Caltrans ATMS**	Freeway vehicle detection
	Ramp meters
	Dynamic message system

	Incident information
Caltrans LCS*	Lane closures
Caltrans TSMSS*	Intersection signal
Trailblazer System*	Dynamic message system for rerouting
RIITS	Local video
	Caltrans video
CalPoly ODS	Environmental sensing
	Transit
HSR Lane Closure*	City lane closures
NextBus	Gold line transit

\* Indicates field system that accepts control messages

\*\* Caltrans' ATMS system accepts control messages and provides limited ICM core system control, including review, approval, and termination of response plans

Future field systems may include probe data providers (GPS location/speed), additional transit information, parking information, and others.

### 5.3. DATA HUB

Data from each of the field sources are received by the ICM Data Hub, represented in red in Figure 5-1. The primary functions of the Data Hub are:

- **Receive data from field elements via existing corridor traffic management centers and regional data networks:** Various data receivers receive data and prepare it for processing by the ICM system. These receivers are generally expected to be built for the specific interfaces defined by each field data source, both in transmission method (REST or SOAP web services, socket-based streaming, file-based, or others) and the intended initial path required for system data processing. The preferred method of information transfer for the system is the Traffic Management Data Dictionary (TMDD) standard developed by the Institute for Traffic Engineers (ITE), currently at version 3.03d (with certain modifications).
- **Process data received from field elements:** Data from field elements must be validated for completeness and data quality prior to use by downstream system components. With such a variety of data sources, often for the same type of field elements, data must be transformed into a common format and set of data semantics. When data is not provided in a standardized format by the source, the Data Hub processes the data into a standardized format, TMDD for transportation assets, GTFS for transit information, or others depending upon the data being received. For all data received, data is transformed into a common set of data definitions as well (such as a single naming standard for all streets within the corridor). However, it is critical to note that this transformation into a standardized format for processing within the data hub, while it maintains the data structures within TMDD, does not generally maintain an XML data format. Internal communications within the data hub (and the DSS) do not use SOAP protocols.

The data hub and DSS use JMS protocols for internal communications and for communications between the DSS and data hub, using either an ActiveMQ or Kafka messaging system and JSON messages. Additional processing is also completed within the data hub, either for specific metrics, calculated parameters, or predictive analytics.

- **Data messaging and communications:** Data provided to downstream systems, as well as internal system command, control, and status data—indeed, all data within the ICM system—is made available via an internal data bus. The method of data transport is via data messaging technologies, namely ActiveMQ JMS messaging or Kafka data messaging systems. The specific message technology is based on the type, size, frequency, and message persistence requirements of the data, data producer, and data consumers. Exceptions to these methods within the data hub are limited, and are generally used for large block bulk data archiving/data transport between persistence stores. In order to accommodate multiple CMS vendors, and to provide communication between the data hub's Kafka messaging system and the DSS ActiveMQ messaging system, an Apache Camel based interface between the data hub and the DSS and CMS systems is included within the data hub. The Camel interface has the ability to provide SOAP and REST based services as necessary, and translation between messaging systems.
- **Data persistence:** The Data Hub also provides data persistence capabilities, allowing for persistence of raw and processed data, system command/control/status, and other system information in a central repository. This data persistence is broken into different time layers, including live real-time data, recent data (0-90 days), aggregated data, and archived data. It includes leveraging state EDW and BI resources as a component of the data persistence capabilities, specifically as the single data store for non-operational data (data older than 90 days). Since data persistence within the data hub is limited to operational uses, and the architecture is based on a pattern of core services connected by messaging, multiple data persistence technologies are utilized specific to the needs of the system and the data being stored. Large collections of time series data, such as sensor data, is stored in a Cassandra database; large relational structures with smaller update frequencies are stored within a Postgres database, and MongoDB is used within data translation pipelines when multiple sources provide the same type of information in differing formats and implementations. NOTE: MongoDB may be used in development and early versions of the system instead of Postgres for the relational data stores in order to accommodate cost and schedule constraints.
- **Orchestration of services and workflows between the three primary ICM systems:** The three primary ICM sub-systems, data hub, CMS, and DSS, while they are independently operating systems, must coordinate actions between the three in order to operate as required. In addition, all communication between the three systems passes through the data hub. This is done to provide a standard interface and interchangeability of DSS and CMS components for future corridors. To do this, the data hub provides the ability to manage workflows, both for orchestration of its own data pipelines and services, but also between DSS and CMS systems.

An example of this orchestration is responding to an incident. Consider a workflow where a CMS system informs the system, via a user initiated event in the CMS, of a confirmed incident. The data hub receives that incident and begins management of services within the data hub and between the DSS and CMS. First it ensures that the required data pipelines are up and running,



starts them if necessary, and ensures persistence of all data being captured during the time period of the incident. Second, it informs the DSS of the incident and requests a response plan. It also informs the CMS of the status of the incident workflow as it progresses. When the DSS responds with the need for a response plan, the response plans, and the results of evaluation and selection of response plans, the data hub ensures this data is captured and stored, and forwards the selected response plan to the CMS with its evaluation and ranking. If the response plan is rejected by the CMS, the data hub stores that result and forwards the next response plan (if another response plan is suitable – ranks high enough above a do-nothing response plan). If that response plan is selected, it then is provided that information from the CMS and stores it. Then the data hub continues the workflow either with updates based on new information regarding the incident received from the CMS, or based on a periodic evaluation based on parameters governing the incidence response workflow within the data hub.

#### 5.4. DECISION SUPPORT

The Decision Support System, portrayed in blue in Figure 5-1, provides the following capabilities:

- **Corridor traffic state determination:** The Decision Support System provides corridor traffic state estimation, providing both geospatial traffic state information and traffic state metrics. Separate arterial and freeway traffic models are used and merged to provide full corridor traffic state at all times.
- **Corridor traffic state prediction:** The Decision Support System provides corridor traffic state predictions for use by corridor operators and for prediction of incident and event response plan performance. A commercial simulation engine (TSS Aimsun) is used to provide traffic predictions. Traffic predictions are generated with an initial state incorporating the estimated traffic state as well as the current state of corridor assets from the data hub.
- **Response plan development and evaluation:** The Decision Support System, upon notification of a confirmed incident, will use a rules-based approach along with traffic state estimation and prediction capabilities to develop, evaluate, and rank response plans for use by corridor operators.

#### 5.5. CORRIDOR MANAGEMENT

The Corridor Management subsystem, portrayed in purple in Figure 5-1, shall provide the following primary capabilities:

- **Presentation of corridor state:** Display of corridor state, including corridor assets (signals, ramps, cameras, dynamic messaging, transit, etc.), estimated traffic state, traffic visualization (camera output), human assets, and physical assets (vehicles, response crews). Asset state includes current operational capabilities, current operating state, current functional state (operating, failed, degraded states), time availability, controllability, etc.

- **Control of corridor assets:** The Corridor Management subsystem provides the capability to control corridor assets, taking response plans approved for execution and executing each of the individual response plan elements by sending commands to the appropriate state, regional, and local systems and entities. The Corridor Management subsystem does not directly control corridor assets, but only sends commands to the state, regional, or local systems that directly command corridor assets.
- **Presentation of response plans:** The Corridor Management subsystem presents each response plan developed by the Decision Support System, displaying in meaningful ways the various response plan elements, how they will change from the current corridor state, how they change during response plan execution, what the predicted outcomes are for each response plan, what the actual outcomes are for a response plan, as well as the various metrics on how the response plans are evaluated and their effectiveness measured. This presentation is expected to be in multiple presentation formats, including geospatial, tabular, comparative, graphical, and other solution-specific formats.
- **Response plan lifecycle management:** Each response plan developed by the system has a specific lifecycle that includes:

1. Initiation 2. Development 3. Evaluation 4. Selection 5. Approval	6. Execution 7. Monitoring 8. Close 9. Post-incident/event analysis
---	--

The Corridor Management subsystem provides a management interface for this lifecycle:

**Table 5-3 Response Plan Lifecycle**

<b>Initiation</b>	The Corridor Management subsystem presents incident and event information received from the Data Hub for review, edit, and confirmation by the corridor operators.
<b>Development</b>	The Corridor Management system provides display of status information received from the Decision Support System (DSS) regarding its decision to develop a set of response plans to an incident or event, as well as the development of those response plans. In addition, it can send commands to the Decision Support System to initiate specific DSS functions, such as mock incident evaluation, or other functions required for the ICM system.
<b>Evaluation</b>	The Corridor Management subsystem receives the results of response plan evaluation and traffic state prediction and displays those results, along with the response plans, to the corridor user. Display shall include individual plan analysis, as well as ranking and comparative analysis of response plans.
<b>Selection</b>	The Corridor Management subsystem provides the user the capability to select a specific response plan for subsequent approval and implementation.

<b>Modification</b>	The Corridor Management subsystem shall provide users the ability to make minor modifications to response plans. NOTE: Initial versions of the software will not have this capability.
<b>Approval</b>	The Corridor Management subsystem provides a selected response plan to the corridor stakeholders at the state, regional, and local levels for review and approval or rejection.
<b>Execution</b>	The Corridor Management subsystem is the sole component in the system capable of sending commands to the various systems in control of corridor assets for individual response plan component execution. It shall have displays that allow control and monitoring of corridor assets via the state, regional, and local traffic management systems and assets.
<b>Monitoring</b>	Given the capability to display asset state, manage corridor assets, and execute response plan elements, the Corridor Management subsystem shall provide an integrated display of response plan monitoring once the execution commands are sent to the field systems. Monitoring shall include displaying when commands have been executed and are complete at the field asset level, alerting users when commands or assets have failed, identifying and displaying variances between expected and actual traffic state, and the ability to take action in the event of response plan element failure for any specific corridor asset.
<b>Close</b>	The Corridor Management subsystem shall be able to return corridor assets to their normal state once traffic has returned to a normal state.
<b>Post-incident/event analysis</b>	The Corridor Management subsystem shall display a corridor post-event analysis for each incident or event. The analysis shall include information from the Corridor Management subsystem itself, especially with regard to the corridor asset response plan execution, response plan execution issues and failures, traffic analysis (expected vs. actual), and other system performance measures. The post-event analysis is primarily limited to information used by the ICM system and its components, as well as metrics calculated post incident/event regarding response plan and system performance. It is not intended to be an exhaustive analysis with extensive additional modeling analysis and “what-if” scenarios.

- **ICM System management:** The Corridor Management subsystem provides management capabilities for the ICM system, including:

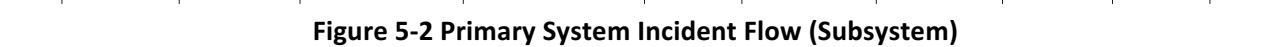
**Table 5-4 CMS Management Capabilities**

<b>Rules management</b>	The Corridor Management subsystem provides a user interface for the management of the rules used within the rules engine. This will include the ability to create basic rules and rule sets, edit rules and rule sets, archive rules and rule sets, provide configuration management for rules and rule sets, make rules active, execute rules and rule sets, and import rules and rule sets.
-------------------------	---

	Generally, rules that can be edited will be simpler rules and rules sets, as more complex rules are likely to require professional development effort and integration. NOTE: The initial implementation of the system will not include this capability.
<b>Response plan management</b>	The Corridor Management subsystem provides users the ability to update the available response plans and response plan elements. This includes adding, archiving, updating, and activation or deactivation of response plans or response plan elements for use within the corridor. NOTE: The initial implementation of the system will include only limited response plan management capability.
<b>System monitoring</b>	The Corridor Management subsystem provides users the ability to monitor the ICM system and components. Systems logs, alerts, status information, control and execution of system functions shall be accomplished by the Corridor Management subsystem. NOTE: The initial implementation of the system may not include this capability.
<b>Security</b>	The Corridor Management subsystem provides user authentication and authorization for the Corridor Management subsystem and all functions available within the Corridor Management subsystem that affect other ICM system functions, including Decision Support and the Data Hub. This does not include user authentication and authorization of individual components such as Cassandra, Postgres, or messaging; rather, it provides these services for the functions that use those components. NOTE: The initial implementation of the system will include capabilities limited to security of the CMS and its interface.
<b>Configuration</b>	The Corridor Management subsystem provides methods and interface for allowing users to change any and all configuration elements within the ICM system. As with security, this does not include configuration of individual system components such as Cassandra and Postgres, but rather for the functions that use these components. NOTE: The initial implementation of the system will have minimal capabilities for this function.

## 5.6. PRIMARY PROCESS FLOW

Figure 5-2 provides the primary system workflow illustrating the basic integration between each of the subsystems. While this illustration is not intended to capture all of the details of the interactions between systems, and the process lifetimes are not intended to be accurate within this sequence diagram, it does provide the basic sequence of events and primary communication channels for the primary workflow within the system, a response to an incident on the corridor. Secondary actions such as persistence, logging, communication dialogs, and others are not depicted in this diagram. Also note that, as described in Figure 5-1, all communication between the DSS and CMS is via the Data Hub's data bus, although the specifics of the communication channel are not illustrated in Figure 5-2.



state, and a set of fixed response plan components to develop a limited number of response plans for evaluation. It submits those response plans to the prediction engine.

- The prediction engine runs a prediction for each response plan, along with another “do nothing” prediction” based on current corridor conditions. It provides the results of those predictions to the response plan development component.
- The response plan development component uses the results of each prediction, current corridor state, and the rules engine to evaluate, rank, and select a response plan to recommend.
- The results of the response plan predictions and evaluation is provided to the CMS for operator selection.
- The CMS provides the results to an operator, who selects the response plan to be implemented and obtains approval for the response plan implementation.
- If the response plan is approved, the CMS executes the response plan by submitting C2C commands to the TMCs and other systems required to execute commands to individual corridor assets.
- The individual TMCs and other systems executing the response plan commands report the success or failure to execute the desired response plan elements, and continue to report corridor asset state.

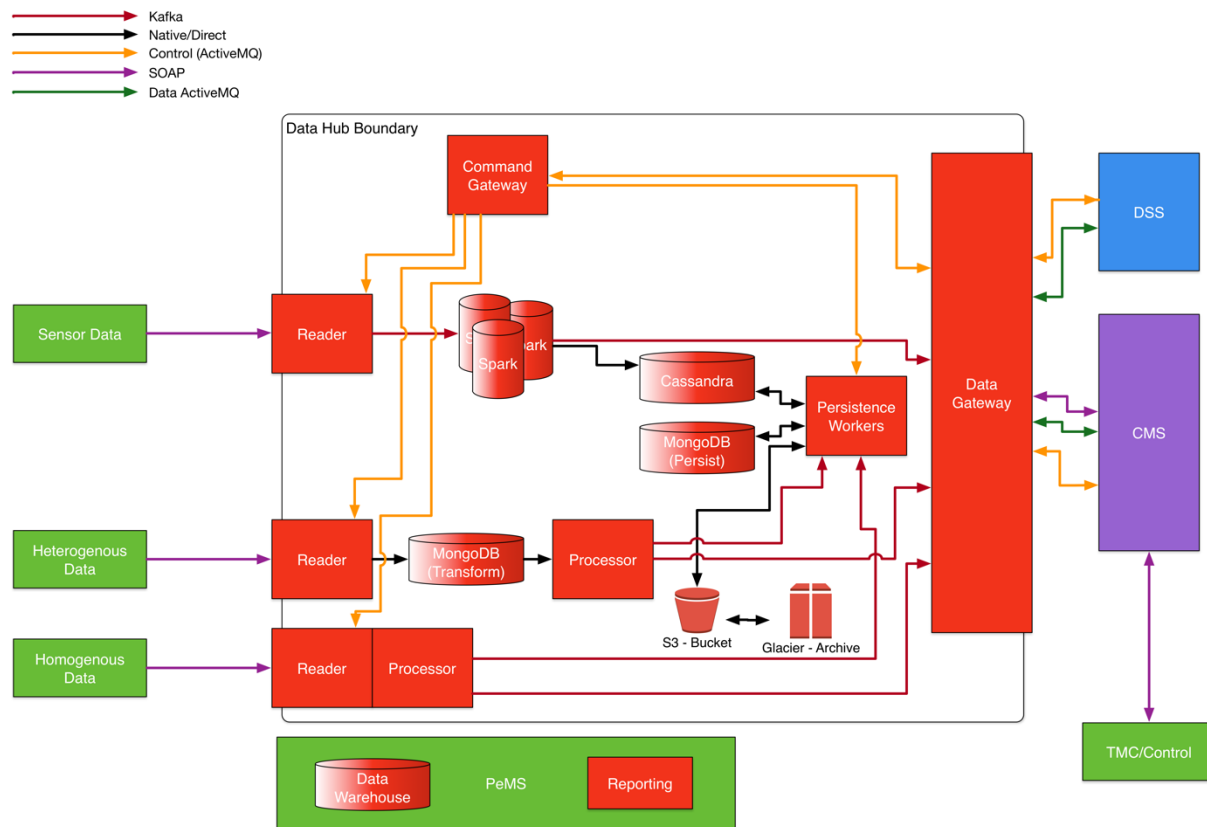
Upon completion of this primary workflow, further processes involved in response plan lifecycle management will occur, such as response plan evaluation, response plan generation to adjust to current conditions, response plan cancellation, and response plan closeout.

## 6. DATA HUB DESIGN

The data hub has six primary roles within the system:

1. Receive information from external providers of the corridor state.
2. Process information received from providers of the corridor state, evaluating the data for quality, providing common metrics and analysis of the data received, conducting predictive analytics to support estimation and prediction within the DSS, and standardizing the format and content for downstream consumption by the DSS and CMS systems.
3. Persist information received, results of processing, and overall system state and command information. Persist operationally required information locally and send to archive older information for longer term storage. Retrieve information stored upon demand (operational data for immediate retrieval, archived data for later, scheduled retrieval).
4. Secure information received, processed and stored.
5. Provide data communications between the primary systems of the ICM core system.
6. Orchestration of services between the DSS and CMS.

In order to fulfill these roles, the data hub provides a series of data pipelines to receive information, process the information, persist the information, and secure the information. The pipelines use Kafka and ActiveMQ messaging systems to transmit information that can be tapped by persistence workers to persist information to the various data stores within the data hub, and retrieve the information and place the information back on the messaging systems for downstream consumption. External interfaces, using Apache Camel, provide information and communication between the data hub, the DSS, and CMS systems. The data hub command gateway, consisting of Netflix Conductor, coupled with Camel, provides orchestration and workflow services for data hub processes as well as CMS, data hub, and DSS operations.



**Figure 6-1 Data Hub High Level Design**

This design consists of individual services connected via messaging. It is a highly decoupled design based on a group of independent services connected by two messaging platforms. Figure 6-1 provides a generalized diagram that illustrates three basic types of data pipelines. Sensor data is generally a high frequency, larger data volume pipeline that handles a larger number of smaller messages. The heterogeneous data pipeline type handles larger sized messages at lower message processing frequencies. Heterogeneous data is characterized by multiple sources providing the same type of data in varying degrees of format and content differences. The homogeneous data pipelines are similar to the heterogeneous data pipelines, but generally are built for a single data source that provides all of a specific type of data.

A description of the different types of components is provided below:

**Reader** – Readers are the beginning of the data pipeline. Their role is simple – to connect and gather the data from a data source using the protocol and format native to that data source, and to place the information within a data messaging channel for downstream processing. In general, it does little or no processing of the data, with any processing generally limited to simple parsing of large batches into individual messages.

**Spark** – Spark is an Apache Spark cluster that provides high speed, high volume, real-time data processing for sensor data. Its role is to provide data transformation, data quality, data processing, and



predictive analytics for sensor data. It receives data from readers, processes that data, and places the data into Kafka message topics for use by downstream processes.

**Cassandra** – Cassandra is an Apache Cassandra cluster that provides storage for time series data, primarily sensing data. Cassandra provides high speed, high volume, clustered storage for sensing data. This data is limited to operational data storage and is not intended as a long-term data store.

**MongoDB (Persist)** – MongoDB (Persist) is an instance of MongoDB dedicated for persistence of the more complex relational data, such as intersection signal plans, ramp meter plans, road networks, asset inventories, and organizational information. It is an operational data store and not intended as a long term data store. It is expected that long-term, this instance of MongoDB would be replaced with Postgres, an open source relational database.

**Persistence workers** – Persistence workers provide both storage and retrieval services for both MongoDB and Cassandra data stores (and, in the future, Postgres). Persistence workers listen to assigned message channels in both Kafka and ActiveMQ, and store the information in the appropriate data store. Requests can be sent to a persistence worker via a command message channel to retrieve data and place that data on a specified message channel for replay purposes.

**Processors** – Processors are used in the homogeneous and heterogeneous data pipelines to process data received from the various data sources. Processors provide data transformation, quality verification, and data processing services for data received from these sources.

**MongoDB (Transform)** – MongoDB (Transform) is an instance of a MongoDB database and is used to transform data received on a heterogeneous data pipeline. Readers store the data received from a source into MongoDB. Processors then query Mongo for the information required for processing and downstream processes including decision support and corridor management systems. MongoDB provides an ideal platform for storing information retrieved in a native format and fast querying of this information with minimal maintenance required when the source format is changed.

**Data hub command gateway (DH command gateway)** – The data hub command gateway provides routing and management of control and status messages for the data hub along with the ability to define and manage workflows that govern communications and processes. Its role is to receive messages routed from the interfaces for the DSS and CMS, as well as internal data hub control and status channels, provide configurable workflow processes, and dynamically route command messages to the appropriate channels for the receiving process. This provides encapsulation of individual services and ensures that new services, service changes, and message system modifications can be accomplished without impacting other systems, and is key to the decoupling of the system services for the entire ICM system. This gateway uses an implementation of Apache Camel and Netflix Conductor.

**External interface/Data gateway** – The external interface/data gateway encapsulates the functions of the data hub, providing a sort of switchboard for use by the CMS and DSS systems. CMS and DSS systems are not required to know the internals of the data hub. Instead they both rely on the external interface/data gateway to provide communication services with the data hub. This external interface/data gateway uses Apache Camel to provide both routing and translation services. Internal data hub messages (Kafka or ActiveMQ) are translated into the ActiveMQ or SOAP protocols used by the

CMS and DSS. Common TMDD and GTFS message formats are used in most communications, so no translation of data formats is required by the gateway.

PeMS – PeMS is a California state system that currently provides state transportation data services. PeMS will provide long term storage for system data, and reporting services for non-real time data and reporting needs.

S3 and Glacier – S3 and Glacier are AWS storage services. These two services will be used to provide long term data archiving for the purpose of retrieving information for replay or analysis within the ICM system. They are not intended as long term storage for PeMS related services.

#### NOTES:

Data hub logging is provided via a common instance of Graylog shared with the DSS. Graylog collects and indexes all logs for each individual component within the data hub, and captures control and workflow status log messages. Its role is to capture system state and error messages for monitoring and troubleshooting needs. This will include data exceptions for data not received or received late from each of the data sources for the ICM system.

## 6.1. DATA SOURCES

The following data sources have been defined for the I-210 ICM project:

**Table 6-1 ICM Data Sources**

Source	Information Type	System	Vendor	Product	C2C TMDD
Pasadena	Intersection signal	Pasadena TMC	McCain	Transparency	Planned
Duarte	Intersection signal	County TMC - KITS	Kimley Horn	KITS	Planned
Monrovia	Intersection signal	County TMC - KITS	Kimley Horn	KITS	Planned
Arcadia	Intersection signal	Arcadia TMC	Transcore	Transuite	Planned
Caltrans FW Traffic	Vehicle detection	Caltrans ATMS	Parsons	Custom	Planned
Caltrans FW Ramps	Ramp meters	Caltrans ATMS	Parsons	Custom	Planned

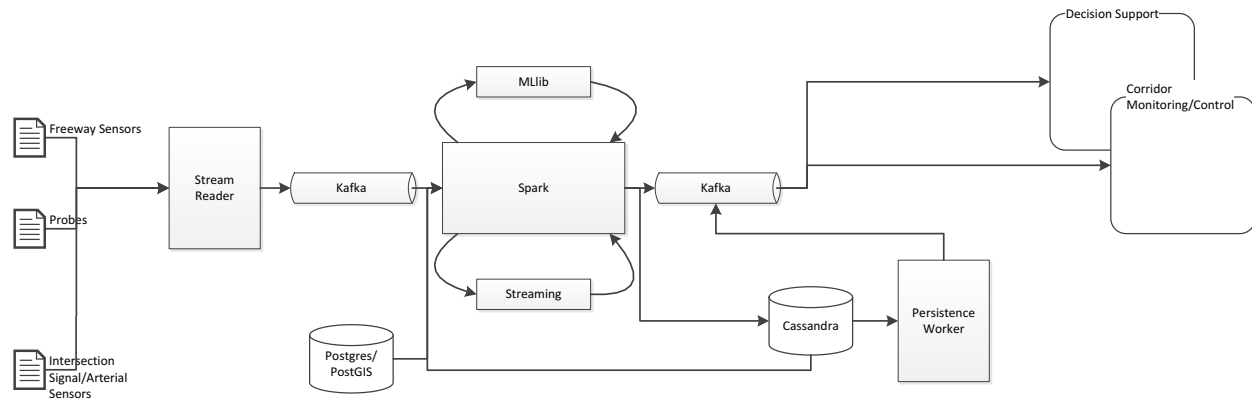
Caltrans FW CMS	DMS	Caltrans ATMS	Parsons	Custom	Planned
Corridor Trailblazer	DMS	Not yet identified		Custom	Planned
Caltrans Intersections	Intersection signal	TSMSS	Transcore	Transuite	Planned
Travel time sensing 1	Travel time	Vendor	Iteris		Unknown
Travel time sensing 2	Travel time	County TMC			Unknown
Environmental sensing	Environmental		Cal Poly	ODS	Unknown
RIITS Transit	Transit		Cal Poly	ODS	No
RIITS Video*	Video Metadata	RIITS			Unknown
Caltrans Video*	Video Metadata	via RIITS			Unknown
Caltrans FW Lane closure	Lane status	LCS			Planned
City Lane closure	Lane status	State HSR system	State of CA	Custom	Planned
LA County	Intersection signal	County TMC - KITS	Kimley Horn	KITS	Planned
CHP CAD	Incident	CAD	manual input by operator		
Caltrans incident	Incident	Caltrans ATMS	Parsons	Custom	Planned
Gold line transit	Transit	NextBus	NextBus		No

\* Video is not passed through or stored within the data hub.

## 6.2. DATA PIPELINES

### 6.2.1. SENSING DATA PIPELINE

The sensing data pipeline design is provided in Figure 6-2. In this illustration, the different sensor feeds can be seen on the left. This illustration shows the reader receiving data from a data source, placing that data on a Kafka topic, Spark consuming that data and processing the data utilizing the MLLib and Streaming Spark libraries, and then placing the processed data on a Kafka topic. Decision Support and Corridor Management systems can access that processed data directly via the data hub data interfaces. Spark also stores results on the Cassandra cluster. Persistence workers can be used to retrieve processed data when requested, and data that is stored in Postgres, such as sensor inventories, is available to Spark if needed for processing that may require such data.



**Figure 6-2 Sensing Data Pipeline Design**

The data sources listed in Section 6.1 that will be processed using this type of pipeline include:

**Table 6-2 Sensing Pipeline Data Sources**

Source	Information Type	System	Data Type	C2C TMDD
Pasadena	Intersection signal	Pasadena TMC	Vehicle detection	Planned
Duarte	Intersection signal	County KITS	Vehicle detection	Planned
Monrovia	Intersection signal	County KITS	Vehicle detection	Planned

Arcadia	Intersection signal	Arcadia TMC	Vehicle detection	Planned
Caltrans FW Traffic	FW Traffic	Caltrans ATMS	Vehicle detection	Planned
Caltrans FW Ramps	Ramp meters	Caltrans ATMS	Vehicle detection	Planned
Caltrans Intersections	Intersection signal	TSMSS	Vehicle detection	Planned
Travel time sensing 1	Travel time	Vendor	Travel time	Unknown
Travel time sensing 2	Travel time	County TMC	Travel time	Unknown
RIITS Environmental sensing	Environmental		Environmental Sensing	
RIITS Transit	Transit		Transit location (future)	
LA County	Intersection signal	County KITS	Vehicle detection	Planned
Gold line transit	Transit	NextBus	Transit location (future)	

### 6.2.2. HETEROGENEOUS DATA PIPELINE

The heterogeneous data pipeline design is provided in Figure 6-3. In this illustration, the different intersection signal feeds can be seen on the left.

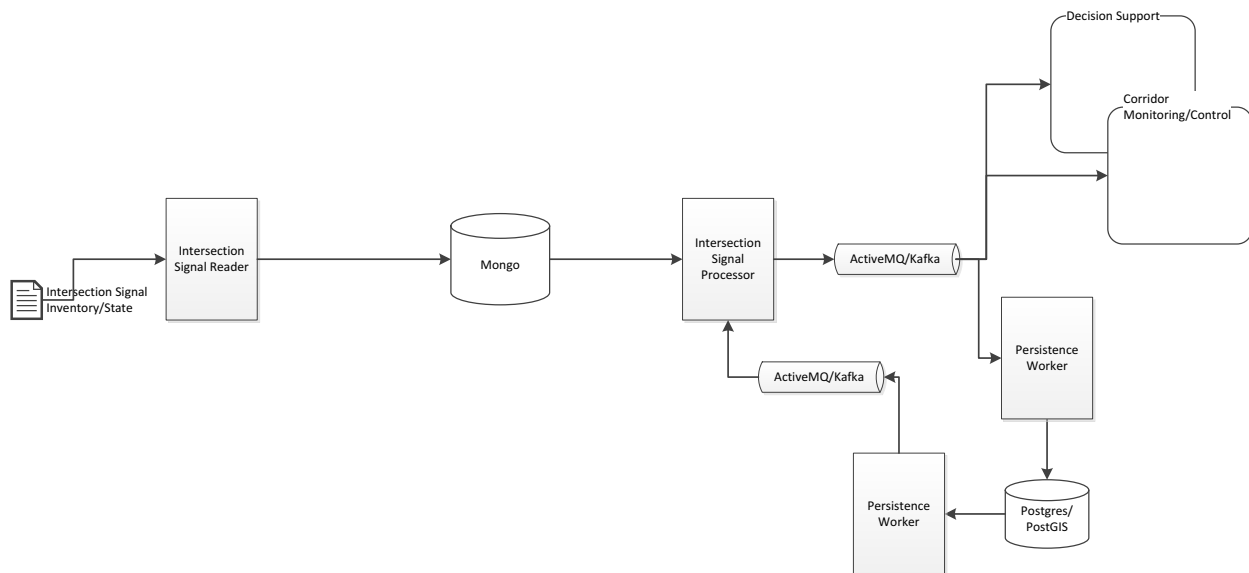
This illustration shows the reader receiving data from a data source and inserting that data in a raw message format into MongoDB. The reader retrieves data from the data source using the data source's native protocols and formats. The preferred communication standard is TMDD, using a SOAP protocol,

using either a request/receive or subscription method. The reader transforms the message from a SOAP/XML format to a JSON format prior to inserting into Mongo.

A message is sent from the reader to the processor to inform the processor of the availability of new data (not shown in the diagram).

The processor then queries Mongo to obtain a desired document with the specific attributes required to make a TMDD message containing a standardized set of information required for downstream processing. The processor also utilizes a standard dictionary of data values so that data from different sources that may be different when received, but that represent the same value, are standardized for downstream consumption. For example, if one source abbreviates boulevard as Blvd, and a second does not abbreviate the word, the processor will standardize to a single form that can be understood by the downstream processes. The processor also conducts data quality checks and any other processing required.

Once the TMDD transformation, quality checks, and any processing is completed, the processor constructs the final TMDD structured JSON message and places it on the outgoing Kafka topic. Persistence workers listening to the topic persist the information for later analysis or replay. The CMS and DSS systems receive the message from the data gateway.



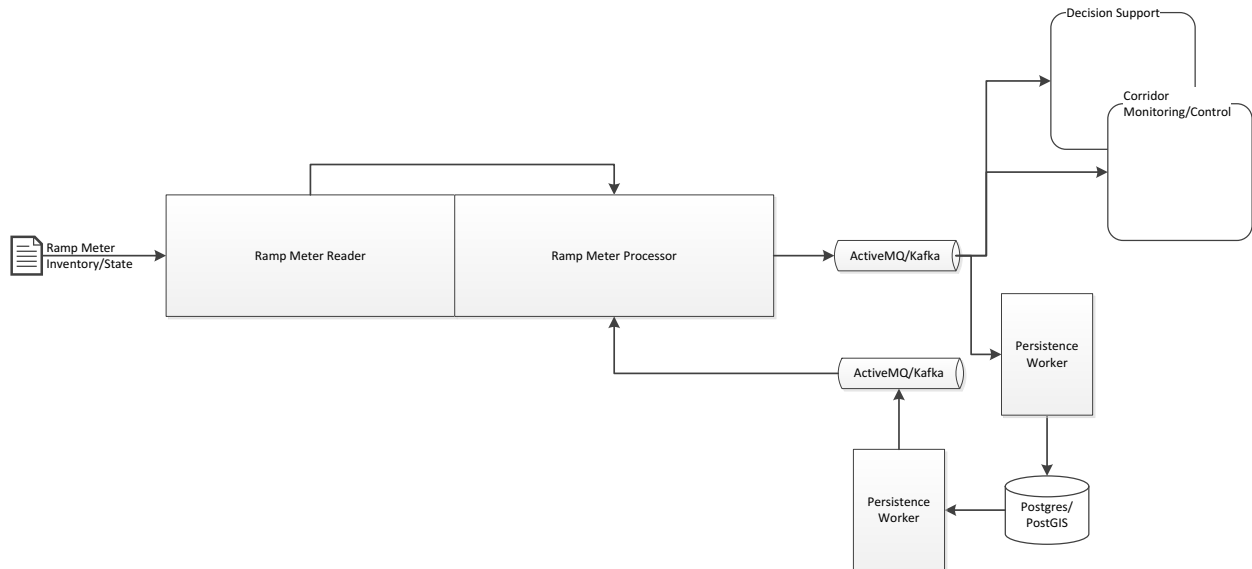
**Figure 6-3 Heterogeneous Data Pipeline**

### 6.2.3. HOMOGENOUS DATA PIPELINE

The homogenous data pipeline design is provided in Figure 6-4. In this illustration, a ramp meter inventory and state source is used as the example feed on the left. The homogenous data pipeline is identical to the heterogeneous data pipeline with only one difference. Homogenous data pipelines are used when there is only a single data source for the type of information being processed. When this occurs, there is no need to store the documents received in a “schema-less” MongoDB database, since

---

there are not multiple data standards and implementations to accommodate. Because of this, the homogenous data pipeline does not use MongoDB in the pipeline, and the reader and processor directly communicate for any data transformation, quality checks, or processing required.



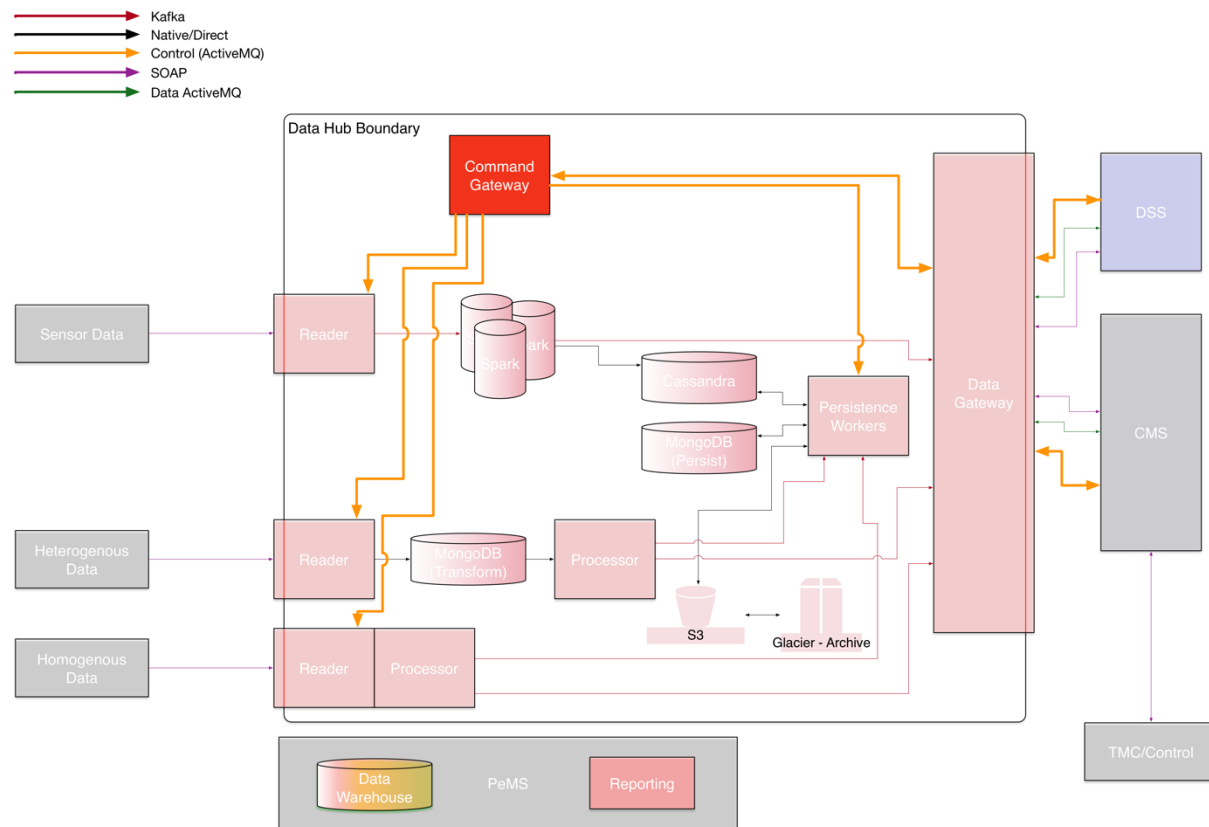
**Figure 6-4 Homogenous Data Pipeline**

#### 6.2.4. PIPELINE CONTROL

All pipelines, regardless of pipeline type, are provided with the same base control set. The controls available for each pipeline include:

1. Start pipeline processing
2. Stop pipeline processing
3. Pipeline status
4. Replay data

All external pipeline control requests are handled by the data hub command gateway and routed via ActiveMQ messaging. Requests received by the gateway are routed to an appropriate endpoint. Figure 6-5 provides a version of Figure 6-1 with the control elements and pathways for pipeline control emphasized.



**Figure 6-5 Pipeline Primary Control Layer**

#### 6.2.4.1. Start Pipeline Processing

The start pipeline processing request will be routed via the data hub command gateway to the appropriate reader within a pipeline. The request will result in these possible outcomes:

1. If the pipeline is currently not started, the reader will begin reading data and inserting it into the pipeline.
2. If the pipeline is already started, an error will be returned from the command gateway indicating the process is already started.
3. If the pipeline cannot be started for any reason, an error will be returned from the data hub command gateway.

#### 6.2.4.2. Stop Pipeline Processing

The stop pipeline processing request will be routed via the data hub command gateway to the appropriate reader within a pipeline. The request will result in these possible outcomes:

1. If the pipeline is currently running, the reader will stop reading new data and will complete insertion of any data currently in process into the pipeline.



2. If the pipeline is not currently running, an error will be returned from the command gateway indicating the process is already stopped.
3. If the pipeline cannot be stopped for any reason, an error will be returned from the data hub command gateway.

#### 6.2.4.3. Pipeline Status

The pipeline status request is a request for a subscription to pipeline status messages that are produced by the Command Gateway.

For each pipeline status request, the data hub command gateway will provide a channel for a subscription to an ActiveMQ topic containing all status messages for a specific pipeline.

The following responses are possible when requesting a pipeline status:

1. A response with the current pipeline status.
2. If the status is not currently available, an error will be returned.
3. If the request is invalid (such as the pipeline does not exist or that the request is not properly formatted), an error will be returned.

#### 6.2.5. PIPELINE STATUS AND LOGGING

All pipeline components provide logging and status information, regardless of pipeline type or component. Logging includes the logging at the application level that is published to the Graylog instance shared with the DSS.

Logging shall be configurable and set on component start and logging levels may include the following:

1. All
2. Debug
3. Fatal
4. Error
5. Warn
6. Info

Logging level will be set upon component initialization when it is started. In general, during normal operation, the following levels shall report – Error, Fatal, and Warn. The information generated by logging shall only be available via Graylog.

Status information is produced by the components and published to a pipeline status ActiveMQ topic. This information is available to external consumers via the data hub data gateway (using a data hub status request) or Graylog. At a minimum, data pipeline status messages shall be provided for the following:

1. Pipeline heartbeat – a status message indicating the pipeline is running sent at a regular interval.

2. Pipeline error message – a status message sent indicating that an error has occurred. Generally, this will include Fatal or Error log messages.
3. Pipeline warning message – a status message sent providing a warning indicating potential issues in operation.

Pipeline components may include other status messages as needed.

#### 6.2.6. CORRIDOR MANAGEMENT SYSTEM-DECISION SUPPORT SYSTEM (CMS-DSS) COMMUNICATIONS PIPELINE

Communication between the Decision Support System and the Corridor Management System is provided via the data hub and managed using the same data and communication pipeline strategy used throughout the data hub, providing:

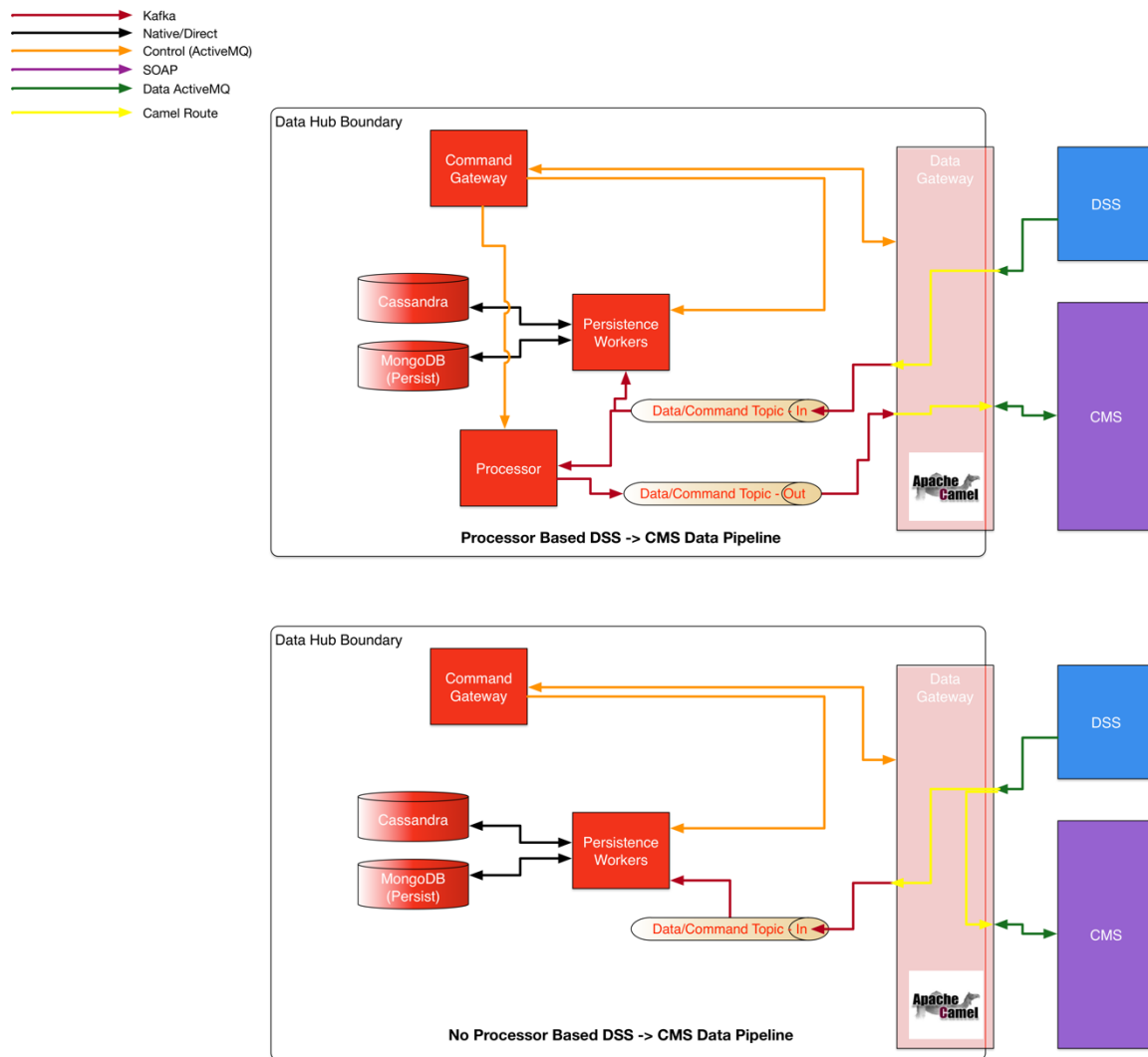
- Traffic estimation results, including geospatial information and traffic metrics
- Traffic prediction, including geospatial information and traffic metrics
- Incident information
- Response plans and response plan evaluations
- DSS Status
- CMS Status
- DSS Control
- Traffic estimation and prediction scenario information

In addition, the pipeline provides persistence within the data hub of all DSS – CMS communications. The pipeline does not include any communications to or from the Corridor Management System that do not involve the Decision Support System.

All CMS-DSS pipelines support TMDD compliant SOAP dialogs in accordance with the Connected Corridors Data Communication Specification. SOAP endpoints are presented at the Data Hub Gateway to support these conversations and are exposed to the CMS. The data for these SOAP interfaces are available for the DSS via the Data Hub Data Gateway. All command requests and responses are also provided from the Data Hub Command Gateway for required workflow management and data hub control actions. While only to be provided as necessary to support the different CMS vendors, ActiveMQ messaging may also be provided as interfaces to the CMS instead of SOAP interfaces.

All CMS-DSS communications are managed on data hub pipelines. For future use, data hubs may support multiple CMS and DSS instances, so all individual pipelines are created for each CMS/DSS pair. All messages between CMS and DSS instances shall be communicated only through pipelines designated for that specific CMS-DSS pair and shall include the source and target system identifiers within the messages.

CMS-DSS data pipelines will generally be in one of the following basic configurations:



**Figure 6-6 DSS-CMS Data Pipeline Configurations**

In Figure 6-6, two of many possible combinations of data exchange between the DSS and CMS are illustrated. The processor based DSS-> CMS data pipeline illustrated at the top of the figure is a configuration used when the information provided by the DSS requires transformation or some other processing before being passed to the CMS. Any processing is always completed within a processor in the data hub and is never done in the data gateway. This is done to provide higher reliability to the data gateway as well as higher scalability and reliability within the processor.

The lower diagram in the figure shows a pipeline without any transformation or processing, used when the formatting and content of the data requires no transformation or other processing before being passed to the CMS. The data gateway routes the data directly to the CMS from the receiving channel in the data gateway while also routing the data to a topic in the data hub for persistence within the data hub.

In both cases, the command gateway manages any workflows required by either pipeline as well as the pipelines themselves. It also manages the data gateway, dynamically creating the data routing within the data gateway. This allows the system to be modified without static changes to the data gateway, simply by modifying the workflow definitions within the Conductor component of the command gateway. New processes can be added independently of any other processes within the system.

Figure 6-8 provides an illustration of a potential configuration of the CMS – DSS pipeline, and its associated primary data topics, including:

- Incidents
- Response Plans
- Decision Results
- Status Request
- Status Response
- Estimation Request
- Estimation Metrics
- Estimation Results
- Estimation Network
- Estimation Scenario
- Prediction Request
- Prediction Result
- Prediction Metrics
- Prediction Network
- Prediction Scenario

Specific pipeline configurations will be defined in the detailed design. Note that the command gateway can listen to any of the pipelines, allowing event-based workflows. An example of an event-based workflow is the incident response workflow. The command gateway, listening to the event pipeline that receives events from the CMS, can initiate a response plan workflow upon receiving the event message. In this situation, the incident received from the CMS will not be routed directly to the DSS, but rather, it will be received by the command gateway, and the command gateway will initiate a response plan workflow. The workflow definition will define a response plan initiation request to the DSS that passes through the data gateway to the DSS, providing the incident information, as well as any dynamic routing information or other workflow related information required by the data gateway and DSS interface to process the request and manage communications with the DSS and CMS.

### 6.2.6.1. Incident

The incident pipeline provides confirmed incident information from the CMS to the DSS. This pipeline may start as an ActiveMQ message, or SOAP incident message at the data gateway. Also note that this pipeline represents a TMDD event class dialog set with both request/response, and subscriptions. Subscriptions are the preferred method of communication. The incident pipeline includes an incident subscription request to the CMS, as well as the corresponding subscription responses. The responses are provided to the DSS via the data hub as ActiveMQ messages.

#### *6.2.6.2. Response Plan*

The response plan pipeline provides response plans to the CMS. As there is no TMDD dialog for response plans, this is a Connected Corridors custom communication. Response plans are provided as an ActiveMQ message via the CMS Data Gateway. Response plans are provided in accordance with the Connected Corridors Data Specification, and include the response plan elements, event/incident information, and associated performance indicators and metrics.

#### *6.2.6.3. Status Request*

The status request pipeline is a communication channel for requesting DSS and data hub status. It is an ActiveMQ queue/message. Requests must include an identifier for the requesting system. A response will include an address for receiving status messages. Status requests are essentially a subscription request, with the response simply a location for subscribing to a continuous stream of status messages.

#### *6.2.6.4. Status*

The status pipeline provides the results from a status request.

#### *6.2.6.5. Estimation Request*

The estimation request pipeline provides a method for the CMS to request an estimation. If an estimation is currently running, the response to the request shall indicate the address where estimation results are published. If the estimation is not currently running, the DSS shall start an estimation and the response shall include a message indicating the DSS is beginning an estimation, and the address where estimation results are published.

#### *6.2.6.6. Estimation Result*

The estimation result pipeline provides the raw results of an estimation request, including for freeways the estimated density, flow, and speed for each network link within the freeway estimation network, and for arterials, the estimated density for each network link within the arterial estimation network.

#### *6.2.6.7. Estimation Metrics*

The estimation metrics pipeline provides estimation metric results, including travel time and current delay estimates for the freeway and arterial networks.

#### *6.2.6.8. Estimation Network*

The estimation network pipeline provides the current estimation network in TMDD format. This pipeline supports TMDD messaging.

#### 6.2.6.9. *Prediction Request*

The prediction request pipeline provides a method for the CMS to request a prediction. The response to the request shall indicate the address where prediction results are published. In the initial version of the software, predictions shall only be provided either as a response to an incident as part of the DSS process, or shall be provided as a replay to a previous prediction.

#### 6.2.6.10. *Prediction Result*

The prediction result pipeline provides the raw results of a prediction request.

#### 6.2.6.11. *Prediction Metrics*

The prediction metrics pipeline provides prediction metric results, included travel time and current delay predictions.

#### 6.2.6.12. *Prediction Network*

The prediction network pipeline provides the current prediction network in a TMDD format. This pipeline supports TMDD messaging.

#### 6.2.6.13. *CMS Command Status Pipeline*

Communications between the CMS and external systems or centers are captured by the Data Hub. They can also be provided to the data hub command gateway so that the data hub may respond to commands issued by the CMS system to external systems.

This CMS Command Status pipeline is a simple implementation of an ActiveMQ topic that is provided a copy of all CMS communications with any external center or system. All such communications are provided via ActiveMQ message containing a text representation of any message sent or received. Messages are keyed with the external system or center communicated with, as well as either send or receive to indicate whether the message was sent to the external center or system (send) or received from the external center or system (receive) as well as the time of message sent or receipt, as appropriate.

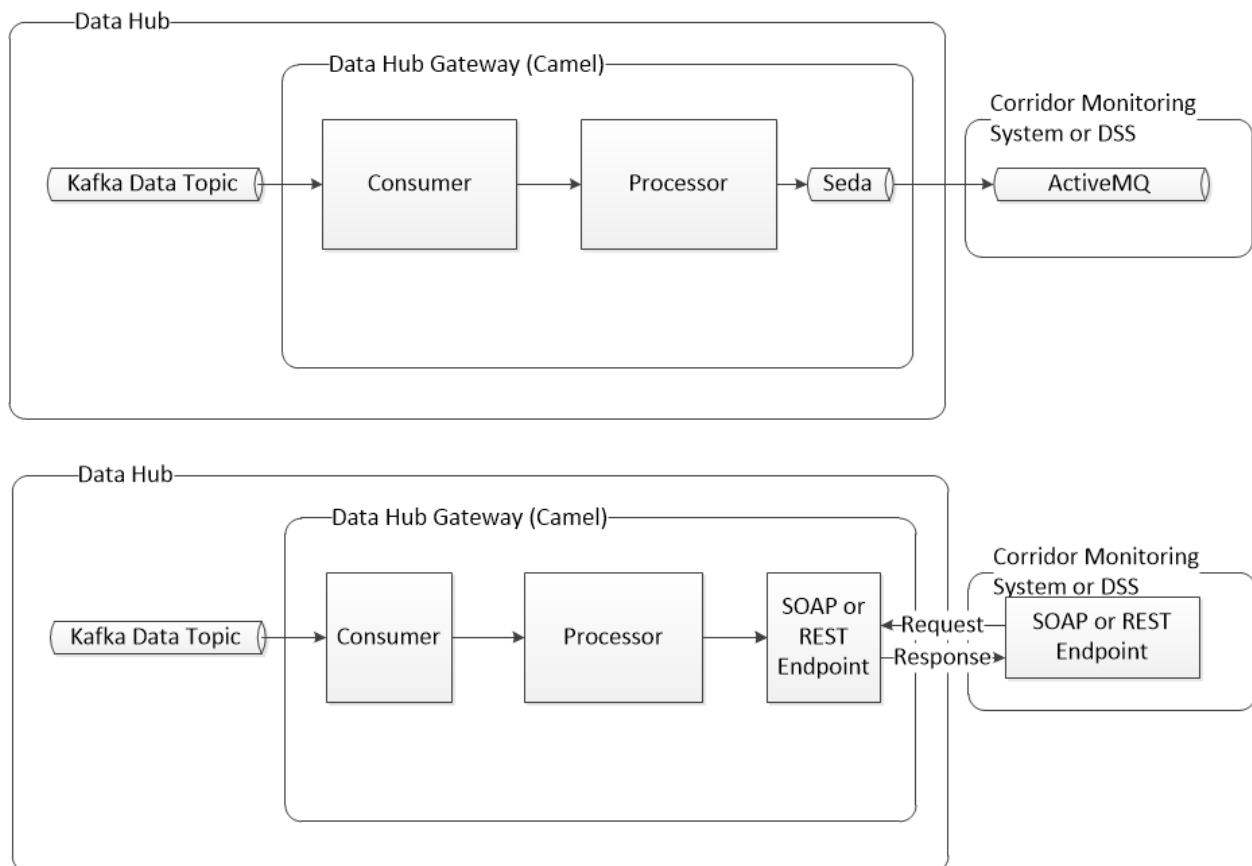
### 6.3. EXTERNAL INTERFACE/DATA GATEWAY

The data gateway provides an external interface for the data hub to the Corridor Management System and the Decision Support System. The purpose of this data gateway is:

- Provide for two-way communication between the Data Hub and the Decision Support System.
- Provide for two-way communication between the Data Hub and the Corridor Management System.

- Provide for indirect communication between the Decision Support System and the Corridor Management System.
- Encapsulate the functions of the data hub from other ICM component systems.
- Provide a secure interface to the data hub.
- Allow for the change from Kafka or ActiveMQ protocols used within the data hub to ActiveMQ messaging or SOAP based services that may be required by other ICM component systems.
- Allow for a switchboard like functionality that can be configured for different Corridor Management System providers or Decision Support Systems.
- Provide for multiple output channels to publish information to multiple downstream consumers.
- Provide event-time or processing-time based ordering of information passed to the CMS or DSS (this is provided as a result of either multiple parallel processors being used within a single data stream, multi-threaded processing of data within a processor in the data hub, or multiple Kafka partitioned messaging may result in data message sequence issues).

The Data Hub Data Gateway uses Apache Camel to provide these services. Two primary design patterns are used as shown in Figure 6-7, first for communication to external ActiveMQ brokers, and second to external SOAP or REST web services endpoints.



**Figure 6-7 Data Hub Data Gateway – ActiveMQ and Web Services Design Patterns**

Each of the two designs uses Apache Camel and implements a Kafka consumer to retrieve messages from a defined Kafka topic. Each also provides for a processor for transformation, sequencing, and routing of messages. For passing data using an ActiveMQ topic or queue hosted by an external broker, Camel's Seda component is used for asynchronous messaging and message flow regulation. For passing data via SOAP or REST web services, an appropriate SOAP or REST endpoint is provided for communication with an external SOAP or REST endpoint.

NOTE: All internal data communications within the data hub are conducted via Kafka or ActiveMQ data messaging and JSON formatting. Native data communications are implemented using TMDD or other appropriate standard formats when available. While the TMDD or other standards may not use JSON formatting, the messages within the data hub that contain this information are JSON formatted, retaining the data structures and relationships within TMDD or other applicable standard. When externally exposed, these data messages are converted back to the appropriate data format dictated by the standard used (TMDD or other).

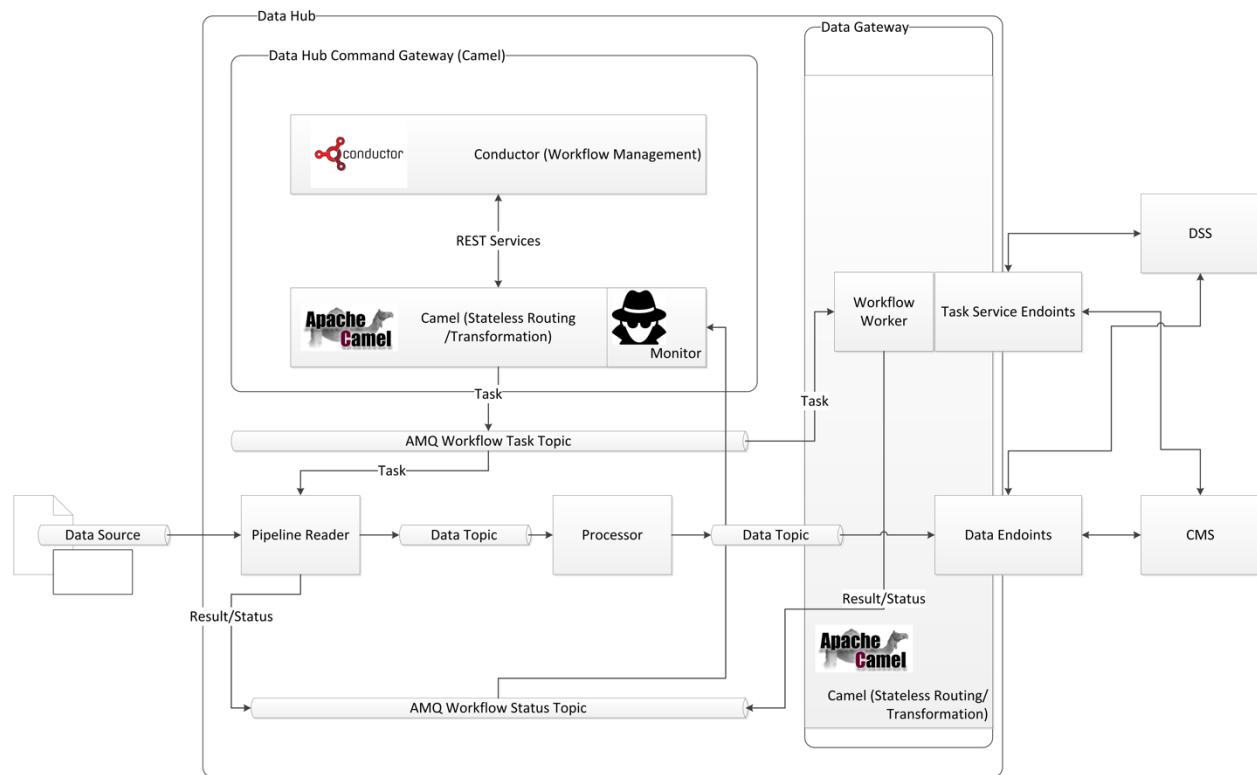
## 6.4. DATA HUB COMMAND GATEWAY

The data hub command gateway is an internal data hub component that manages all command, communication, and coordination activities of the data hub and orchestrates communication and control of the three core ICM system components. Since the DSS and CMS do not communicate directly, the data hub orchestrates workflows and communications between the two components and the internal components of the data hub. Similar to the data hub data gateway, it uses Apache Camel to route and process core ICM System commands. In addition, it uses a workflow component designed for a microservice architecture used by Netflix in its own production environment called Conductor. The command gateway provides the following capabilities:

- Receive all core system requests for services.
- Provide execution management of all requests for data hub services.
- Provide execution management of workflow processing for requests involving simple single step workflows or complex multi-step workflows.
- Persist service requests and their outcomes (success or failure).
- Manage internal operations of the data hub (starting/stopping services, restart on failure, etc).
- Orchestrate actions between the DSS, CMS, and data hub.

NOTE: The data hub command gateway is an internal data hub component. At no time are the message queues and topics that it communicates with directly connected to external systems. All queues and topics that communicate with this component are internal queues and topics connected to data hub components or the data hub data gateway (for communication with external systems).





**Figure 6-8 Data Hub Command Gateway**

The data hub command gateway consists of just a few primary component types. These include:

- Conductor and related workflow components
- Camel and related routing and transformation components
- ActiveMQ workflow status topic
- ActiveMQ workflow task topic
- Monitor

#### 6.4.1. CONDUCTOR

Conductor is used for orchestration of the individual pipelines and both internal and external requests for services. Conductor is an open source project developed by Netflix to manage some of its own production workflows, particularly with a micro-service architecture. There are a number of similarities between the use cases for which Conductor is designed and the orchestration of the individual pipelines and the requests for services from the DSS and CMS, as well as orchestration of the communications between DSS and CMS that make it well suited for use.

Conductor is used in an “as-is” configuration “out-of-the-box”. The data hub implementation uses the REST interfaces and in-memory database that comes with Conductor. No modifications are made to Conductor. It provides workflow management and control, with workflow definitions described in JSON format.

#### 6.4.2. CAMEL

To adapt Conductor to the data hub interfaces, Camel is used as an interface to the ActiveMQ and Kafka data hub messaging services. Camel provides the interface between Conductor's native REST interface and the data hub's Active MQ and Kafka data bus elements.

#### 6.4.3. ACTIVEMQ WORKFLOW STATUS TOPIC

The ActiveMQ workflow status topic is used to allow individual data hub components to provide workflow status messages to Conductor, via the Camel interface. It also is used to provide general system and component status information for use in workflow management. The data hub data gateway also provides workflow and external system status (CMS and DSS) messages via the workflow status topic. The CMS and DSS systems are never required (or allowed) to directly message via the Workflow status topic, but instead, their communications are managed by the data hub data gateway.

#### 6.4.4. ACTIVEMQ WORKFLOW TASK TOPIC

The ActiveMQ workflow task topic is how workflow tasks are dispatched from Conductor to the individual data hub components, or the DSS and CMS systems (via the data hub data gateway). As with the workflow status topic, DSS and CMS systems are never directly allowed to communicate with the workflow task topic. Instead, the data hub data gateway manages the distribution of tasks and the resulting responses via the DSS and CMS APIs.

#### 6.4.5. MONITOR

The monitor listens to the workflow status topic for system and component status messages, always providing data hub, DSS, and CMS state and status information to the Conductor workflow manager. This allows Conductor to respond to degraded system state, either by disabling specific workflows or starting other workflows to respond to system component failures.

## 7. DECISION SUPPORT SYSTEM DESIGN

The decision support system's (DSS) primary roles include:

- Provide response plans following receipt of a confirmed incident.
- Evaluate response plans and rank them based on user defined criteria.
- Provide one or more recommendations to corridor operators and stakeholders via the Corridor Management System, whether to implement a response plan and which response plan to implement for each received confirmed incident.
- Provide response plan evaluation results for corridor operators review via the Corridor Management System.
- Evaluate implemented response plan effectiveness and recommend new response plans when appropriate.

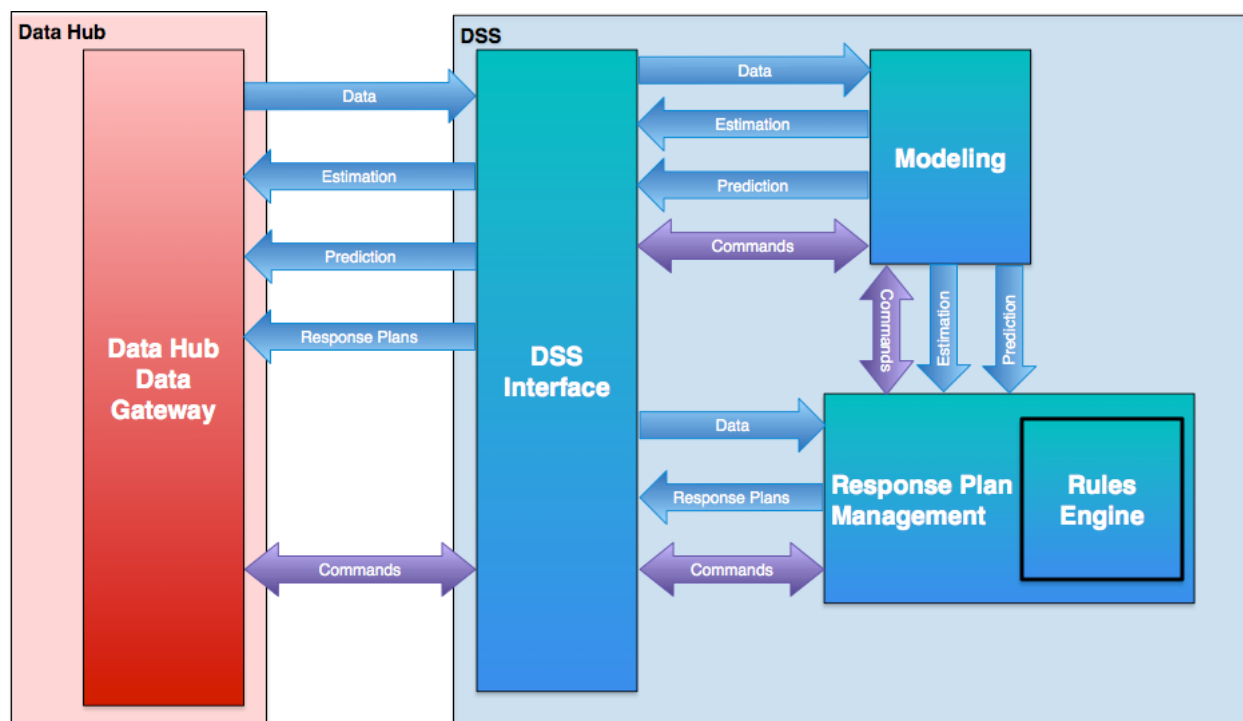
The DSS accomplishes this with three primary components:

- DSS Interface
- Response Plan Management
- Modeling

The DSS interface provides an interface for sending and receiving information with the data hub, providing routing and transformation services for the other DSS components. The response plan management component receives incident information and coordinates the development and evaluation of response plans, using a rules engine that provides configurable logic and rules to create response plans, evaluate response plans, and rank response plans. The modeling component provides traffic estimation and prediction capabilities to support response plan creation, evaluation, and ranking.

## 7.1. DSS HIGH LEVEL DESIGN

The DSS's three primary components are described in Figure 7-1. The DSS is one of the three primary ICM system subsystems, and communicates with the ICM system via the data hub's data gateway.



**Figure 7-1 DSS Architecture**

All corridor information is received by the DSS interface from the data hub data gateway. The modeling and response plan management components receive all corridor data and system status via a series of data channels available from the DSS interface.

All commands to the DSS and requests for DSS status are received via a set of command channels exposed on the data hub data gateway. These commands are in turn forwarded by the DSS interface to either the modeling or response plan management components.

All communications with the DSS via the Data Hub Data Gateway are managed via ActiveMQ messaging. Communications between the DSS interface, Response Plan Management, and Modeling are enabled via REST or ActiveMQ messaging. The response plan management and rules engine are tightly coupled via their java based APIs.

Output from the DSS consists primarily of response plans and their evaluations from the response plan management system, and estimation and prediction results from the modeling system.

## 7.2. DSS INTERFACE

The DSS interface provides a bridge between information from the data hub's data gateway and the internal components of the DSS. Figure 7-2 provides the method for passing information between the data hub and the modeling and response plan management components of the DSS.

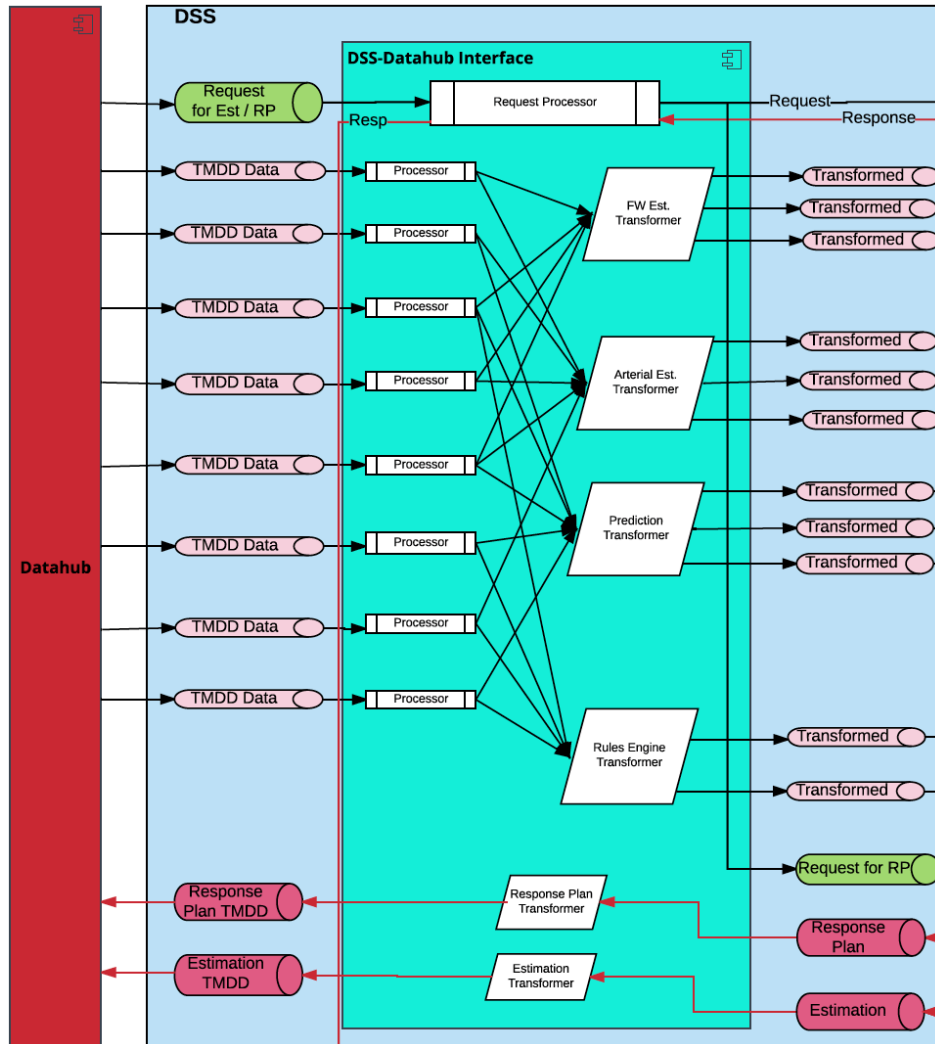


Figure 7-2 DSS Interface High Level Design

The primary data flow is as follows:

- Data is placed on DSS ActiveMQ topics by the data hub data gateway.
- Data is pulled from these topics by processors specific to the type of data on each topic. The data may receive any desired computation, aggregation, splitting, quality checks or other processing required by the DSS here.

- The data is passed to a consumer-specific transformer. This transformer prepares standardized TMDD data (or other format when TMDD is not appropriate) for the consumer, transforming it into any consumer specific object required for the consumer.
- The processed and transformed data is passed into a consumer specific ActiveMQ topic for use by the consumer.

Request processors provide routing of requests from the data hub to specific DSS components as well as providing initial receipt status back to the data hub for each request.

Response plan and estimation transformers prepare the response plans, prediction results, and estimation results, transforming them from DSS specific objects (modeling objects or response plan manager objects) to TMDD or other standardized objects.

### 7.3. RESPONSE PLAN MANAGEMENT

Response plan management provides functions for the development and selection of response plans, including coordination of the actions of the response plan management and modeling components of the DSS.

The response plan management component responds to commands from the data hub, and upon receipt, orchestrates actions between itself and modeling components to determine if a response plan might positively impact traffic conditions and if so, develop several response plans, evaluate them, and recommend a response plan for implementation. The response plan management component also has a limited selection of methods implemented so that other specific requests can be made via the data hub.

The response plan management also monitors the Decision Support System and its components. It has limited capabilities to perform specific startup activities required for DSS operation and monitor DSS component status.

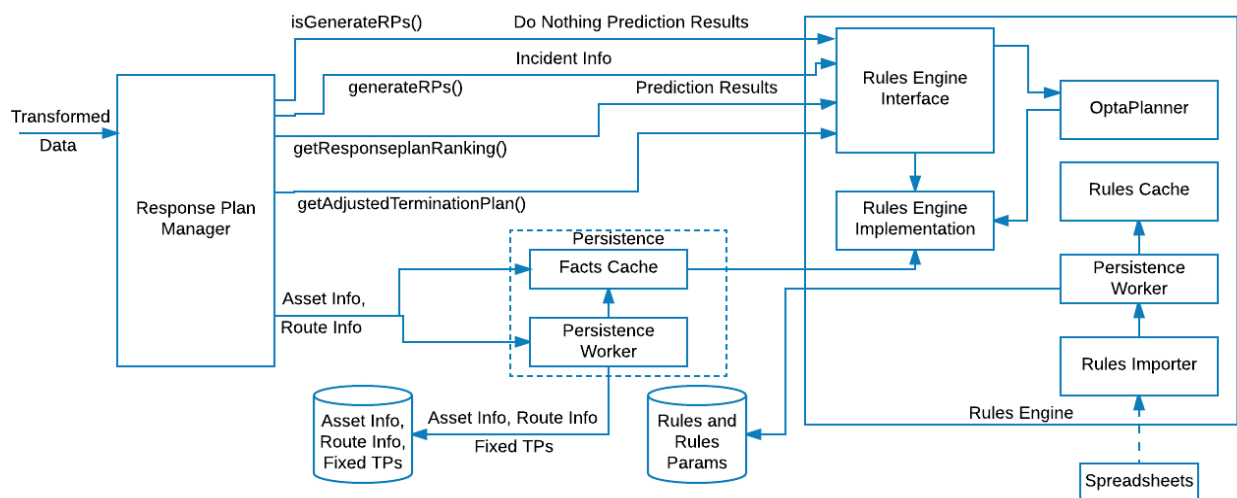


Figure 7-3 Response Plan Management Design

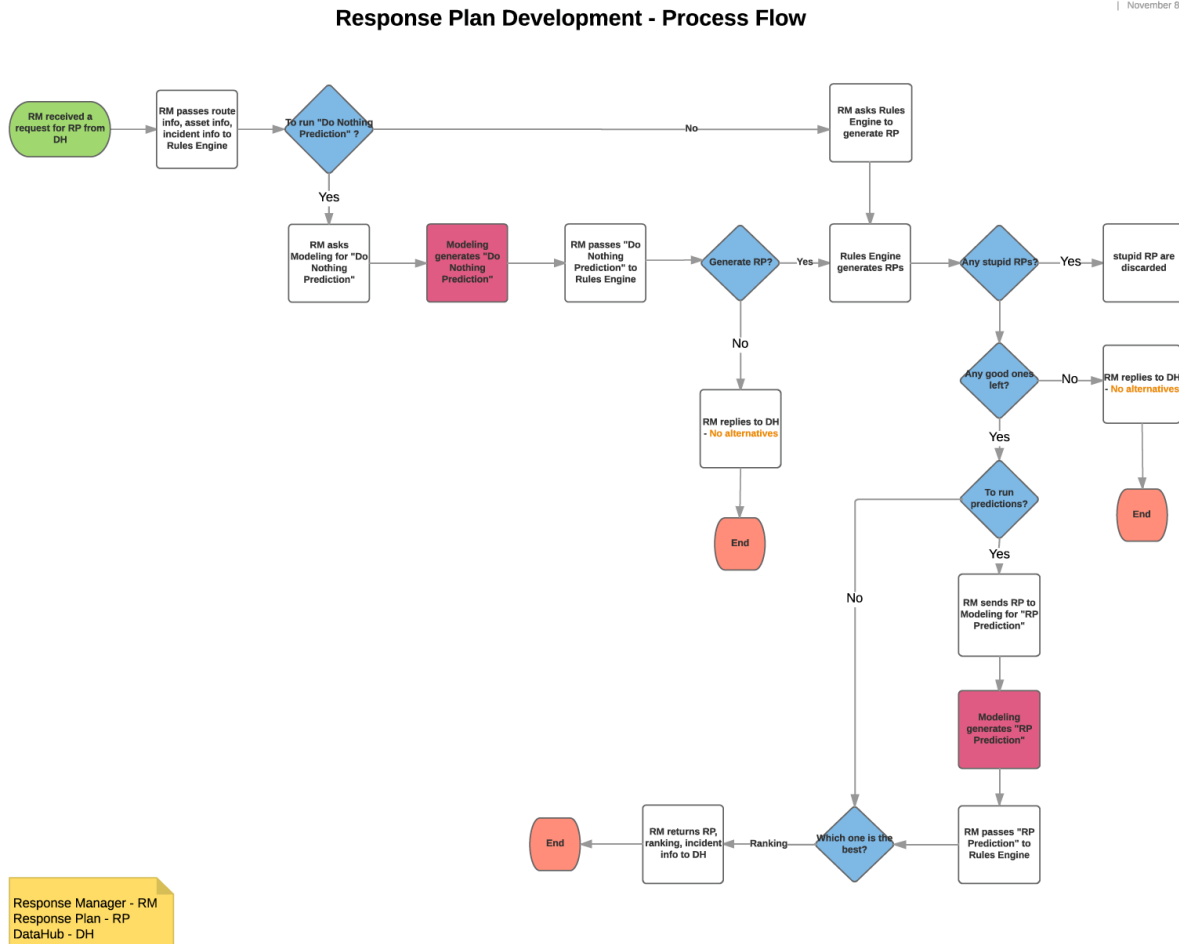
On the left of the response plan management component in the figure is the transformed data received from the DSS interface. This provides asset information, route information, and incident information to the response plan management component. This information is maintained in a local data store.

The response manager, upon receipt of a specific request or incident manages the specific workflows required to respond to the request or to provide a response plan recommendation. This includes requesting decisions and results from the rules engine, information regarding current traffic state from the modeling system, or specific predictions for response plans received from the rules engine. The response plan manager also assembles the complete response plan, and provides it as a response to send to the data hub for storage and distribution to the CMS.

The rules engine used within response plan management is a combination of java components and the open source Drools rules engine. It is designed to respond to a limited number of primary questions that can be asked by the response plan manager. These limited questions include:

- Does the current confirmed incident, with the results of a “do-nothing” prediction, require development and evaluation of response plans?
- What response plans should be evaluated? Create those response plans from a limited set of response plan components.
- Which response plan, given the response plans developed and their respective predictions, should be recommended, listed in a ranked order?

A flow diagram illustrates the basic logic used in developing response plans:



**Figure 7-4 Response Plan Manager Workflow**

The rules engine is a Java library that is included in the response plan manager component with a defined API specific to answering these defined questions, given the current state of the corridor assets, defined limits within a set of user specified rules (such as “do not use this route between the hours of 3 and 5 pm”), current traffic state, current time, and incident information.

It is intended that the final user interface for developing, editing, testing, and evaluating rules and their performance will be in the CMS. However, initially, this will be limited and require IT support to assist with rules development and uploading spreadsheets of rules to a known location.

## 7.4. MODELING

The modeling system provides two primary services: traffic state estimation and traffic prediction.





- Project Manager (in brown) – provides overall system orchestration, cloud scaling and EC2 management, workflow, and external REST interfaces. A user interface is available for estimation model development. This is currently used for internal testing and QA.
- Persistence – (green/gray) – persists traffic/traffic infrastructure data, model and model building definitions, raw and processed results.

Communication between the DSS interface and the modeling system is via a REST API exposed by the Project Manager. Data feeds from the data hub provide data via ActiveMQ data topics. Internal component communication between modeling system components is via ActiveMQ topics (data) or queues (command).

Technology stack:

- Primary components are built using Java with the Spring framework. Hibernate is used with components requiring Postgres access.
- Persistence is built with Java based persistence workers (both for store and retrieve operations) and Postgres or Cassandra for persistence.
- Messaging via ActiveMQ
- Project Manager REST services hosted on Tomcat
- Log Management via Graylog
- Amazon Web Services, including the following AWS services:
  - EC2
  - RDS (Postgres)
  - S3
  - VPC
  - IAM
  - Other services are used for code repository, build, and deployment services.

Platform/Application Servers/OS

- Most any version of Linux OS is acceptable, Ubuntu is currently used
- Amazon Web Services
- Tomcat

*General Architectural Approach:*

The approach used has been to develop specific application components targeted to specific tasks, connect them to each other via messaging or REST services depending upon their application role (back end processing or service delivery), and provide scaling capabilities that detect load on each component (CPU/thread utilization and its feeding queue state) and scale the number of instances of that component based on demand.

*Primary component descriptions:*

Model Engine – There are two methodologies applied within the model engine. The first is used for freeway estimation and implements the CTM algorithm for simulation and an ensemble Kalman filter to assimilate real-time road sensor data. This implementation uses a user defined road network, accompanying infrastructure elements including ramp meters and sensors, traffic demand, traffic

behavior information (split ratios at intersections and freeway ramps), and road/traffic characteristics required by the CTM algorithm (fundamental diagrams). CTM based simulation can be used both in estimations of current state and predictions of future state, but is used only for estimating current traffic state on freeways within the ICM system.

The second methodology applied in the model engine is used for arterial estimation. This method uses an algorithm using intersection sensing, signal timing, and cycle information to produce an estimated traffic state on the corridor arterials.

For either estimation methodology, the model engine receives a request for a model run containing the scenario to be run and the desired configuration parameters. The model engine then starts a continuing run of the model and outputs the link state (traffic density and speed) for each road network link.

Prediction Engine – The prediction engine is an implementation of TSS’s Aimsun software. It includes the Aimsun simulation software as well as a component that builds simulation models for use in the prediction workflow. To build a simulation model, this component combines a base simulation model with data from the data hub indicating current corridor asset state (signals, ramps, signs, etc), current traffic state from the freeway and arterial model engines, and the response plans from the response plan management system to create a model that can be used to evaluate a response plan. It also provides the prediction metrics for that response plan that are required for presentation to the user and passed to the rules engine for response plan recommendation and selection.

Project Manager/Project Manager App - The project manager app allows external systems to interact with the modeling system. Project manager app provides REST services and messaging to initiate actions by the rest of the system. The project manager app is served via a Tomcat application server.

Scenario Builder - The scenario builder is used to create a CTM model scenario to be run. It is rarely used, but may be used when corridor infrastructure elements are modified, such as a ramp meter. It is a back-end component that receives model components (road networks, intersection signals, ramp meters, traffic demand and split sets, etc) from the project manager app and builds a model suitable for running in the freeway model engine.

Calculators - The calculators are used to take the model engine link state data and process it for consumption, combining arterial and freeway estimation, calculating specific metrics for analysis, and providing visualization information for the CMS.

Persistence Worker - Persistence workers are components that are designed to either take data from a queue or topic (model engine data, model/scenario information, or calculator output) and persist it in one of the data stores (Postgres or Cassandra) or, upon request, retrieve data from one of the data stores and place it on the appropriate queue or topic for consumption by downstream processes.

Readers and Processors - These components are used to retrieve data pushed from the data hub and place it into a queue or topic for consumption (reader), or to retrieve data from a queue or topic and process it for consumption by the estimation or prediction engines (processor).

This page left blank  
intentionally

## 8. SECURITY DESIGN

Security design for the ICM system is based upon the following principles:

- Minimize attack surface
  - Minimal exposure to public networks
  - Isolate network subnets to protect sensitive data and processes and use DMZ for publicly exposed communication points
  - Limit or eliminate access to all internal system servers
- Authenticate all connections
  - Use certificate based authentication between systems
  - Users limited to CMS system access and CMS system authentication protocols
- Encrypt all external facing communications (data-in-motion) and sensitive data stores (data-at-rest)
- Isolate individual processes and access only by those processes or people that require access (principle of least privilege)
- Automated security and process monitoring
  - Kill all failed or non-responding processes
  - Verify configuration of all launched processes
  - Monitor system and process access and access violations
- Automate system launch processes
- Validate all incoming data

The system design, based on independent services, limited access to data stores, segmented responsibilities, and cloud deployment, provides significant opportunity for enhanced security. Each of the areas listed above are described below.

### 8.1. MINIMIZE ATTACK SURFACE

There are several methods to be employed to minimize the systems attack surface. These include the following:

- Secure network connections to the ICM system.
  - Use of dedicated fiber connections and/or secure VPN connections to the various TMCs involved in providing information and executing response plans.
  - Use of a secure AWS DirectConnect connection provided by AT&T's NetBond service.
  - All connections to external systems pass through District 7 TMC Foreign Entity (FE) firewall.
  - All connections are secured via VPC security group inbound and outbound access rules.
- No direct connections to the public internet.
- The system runs within a Virtual Private Cloud, separated into "public", "shared", and "private" subnets. Two "public" subnets exist, one to receive data from TMCs, and one for the CMS to manage and execute response plans. "Public" subnets are exposed only to private, protected networks within TMCs and only through secure connections (described above). "Shared" subnets are used for the messaging systems, providing a secure bridge between "public" and "private" subnets. "Private" subnets are used for system processing and data storage components.

- One goal of this deployment is to use automation for system launch and deployment, with no human intervention required. This allows for deployment without SSH keys on individual AWS EC2 instances, since, in the case of system failure, the response is to isolate or kill the failed system, and, using automation, deploy a new system in its place. Without SSH keys, there is no access to the OS of any deployed system.

## 8.2. AUTHENTICATION

Strong authentication shall be required for all system access, including all connections to internal services. This will include:

- Two-factor authentication required for AWS administration
- Use certificate based authentication for external systems communication

The users will be limited to access of the CMS system interface. No other access shall be granted to system users. Authentication of users shall be based on the security protocols and implementation of each individual vendor. Since there are three different CMS systems that will be deployed at different times based on their evaluation period, authentication methods may differ and implementation shall certainly differ.

## 8.3. DATA ENCRYPTION

Encryption, via SSL, shall be used for all communications with external systems. This includes communication with all data hub readers from TMCs and with each CMS solution to TMCs.

In addition, while it is not expected to be required, all data stored in any of the databases, or AWS data store such as S3 shall be reviewed with Caltrans for the need to encrypt the data store. If any data element is to be encrypted in storage, it shall be encrypted when transported internally via the messaging systems.

## 8.4. PRINCIPLE OF LEAST PRIVILEGE

The principle of least privilege is defined as ensuring that every machine, process, and user are only allowed to access the information and resources required for its operation and purpose. All other access is prohibited.

The ICM system shall adhere to this principle, using the following design elements:

- Access to EC2 instances containing databases are only granted to persistence workers, specific readers (for MongoDB transformation pipelines), Spark, and PMA (modeling).
- EC2 instances requiring only access to message brokers are restricted to only those message brokers required.
- No SSH access to EC2 instances.
- TMC and any other external access is limited to specific data hub readers designated for communication with that TMC and CMS components required for communication with the TMC.

- DSS, CMS, and data hub systems shall encapsulate all access to internal components through their specific gateways and interfaces either in public (externally facing communications) or shared (internal communications). No other access shall be granted.
- AWS IAM groups, users, and roles shall be configured in accordance with principle of least privilege.
- Access to Graylog, database, and message broker administration shall be granted only to system administrators for each specific service.

NOTE: CMS security is dependent upon specific implementation of each CMS system and configuration of security within the CMS system.

## 8.5. AUTOMATED SECURITY AND PROCESS MONITORING

While automated security and process monitoring is not likely to be configured fully upon the initial launch, it is intended that this will be implemented during production operations in close cooperation with Caltrans IT. This shall include the use of CloudTrail and CloudWatch to accomplish the following:

- Kill all failed or non-responding processes
- Verify configuration of all launched processes
- Monitor system and process access and access violations
- Quarantine or kill and relaunch any suspected compromised systems or operations
- Conduct regular automated AWS inspections of the system configuration

## 8.6. AUTOMATE SYSTEM LAUNCH PROCESSES

To ensure consistent system launch, delivery of system services, and secure system configurations, all system launch processes will be automated using CloudFormation or, in the case of some modeling components, use of the EC2 APIs and programmatic launch of components from controlled system images. This will include the automation of system upgrades and patches. This will minimize the possibility of the introduction of inconsistent and incorrect security and system configurations, and provide the ability to more easily monitor for and correct system deficiencies.

## 8.7. VALIDATE ALL INCOMING DATA

To prevent denial of system services or security issues, as well as ensure proper operation of the system, all data received by the data hub shall be validated to ensure that is well-formed and within reasonable tolerances for system processing (volume, content, timing, etc.)

This page left blank  
intentionally



## 9. SYSTEM INTERFACE AND MESSAGE SYSTEM DESIGN

There are several primary interfaces defined for the ICM system. These include:

- Data hub interface to external data sources, primarily the different TMCs providing corridor asset information such as asset inventories or asset state (such as intersection signal cycle information).
- Data hub to DSS interface
- Data hub to CMS interface
- CMS to external command targets, primarily the different TMCs that will execute response plan actions based on TMDD commands received from the CMS.

In general, communications with external TMCs and data providers or consumers is accomplished via TMDD SOAP based messaging, using the Connected Corridors modified version of TMDD. This includes the data hub communication with external data sources which provide the data necessary to operate the ICM system and the CMS interface to the command target TMCs which execute commands received from the CMS. The data hub to CMS interface has a SOAP interface using the same modified TMDD specification for use if desired by CMS vendors. Extensions to the TMDD specification for response plan exchange and approval workflows have also been added for data hub to CMS and CMS to Caltrans ATMS interfaces.

The data hub to DSS interface uses ActiveMQ and JSON formatted TMDD-like messages. The data hub to DSS interface uses the data hub's data gateway and the DSS interface to communicate, with the data gateway's Apache Camel implementation pushing messages onto the DSS's ActiveMQ broker topics and queues, and the DSS interface providing transformation and routing for the messages within the DSS, as well as providing a corresponding path for messages from the DSS to the data hub's data gateway.

In addition to the SOAP interface provided by the data hub data gateway for the CMS, optionally the CMS may use ActiveMQ and the data gateway will communicate with the CMS in the same manner as the DSS.

Within data hub and DSS, the use of independent processes communicating via messaging is a key design element, which used properly provides a number of significant benefits.

- Parallelization of processing
- Scalability of services for high data volumes
- Improved resilience, simplified redundancy, and fast failover
- Improved flexibility with the ability to add additional capabilities or change system configuration without interruption to existing services
- Ability to add additional corridors with minimal effort, cost, and disruption
- Ability to maintain, patch, or upgrade the system with minimal or no disruption in service

A critical element of this design is the messaging systems that link the independent services. There are two primary messaging systems used to accomplish the linking of these services and maintain communication within and between system components:

- ActiveMQ
- Kafka

## 9.1. DATA HUB INTERNAL MESSAGING

The data hub uses both Apache ActiveMQ and Apache Kafka messaging systems for connecting its internal services. It is important to note that the design of the data hub is based on loosely coupled services connected by messaging, and orchestrated by the command gateway. The messaging only provides data transport and command communications. The services the messaging provides have no knowledge of the other services either providing the data used by those services, nor the services consuming the data they provide. Only the orchestration service provided by the command gateway has any knowledge of the full suite of services, and as a result of the workflows inherent in the connections made and defined within its workflow engine (Conductor). This has significant advantages in scalability, speed of processing, and flexibility, but also comes without transactionality. There is the potential for message loss that must be addressed within the workflows defined within Conductor and within the design of service recovery in the event of system failure.

### 9.1.1. DATA MESSAGING AND KAFKA

Kafka is a clustered, high volume, high speed, highly scalable, persistent messaging system engineered specifically for real time data pipelines and streaming data applications. It was developed initially by LinkedIn, and released as an open-source product in 2011.

The data hub uses Kafka for moving data between its various services, as a data transport mechanism for its data pipelines. The data hub will use the following configurations for Kafka messaging:

- A minimum of three Kafka nodes, scaled by message volume experienced
- The use of multiple Kafka partitions per message topic is preferred, however, single partitions are used when messages are to remain event-time ordered per data source
- Zookeeper for process coordination

Data may be encrypted in transit if necessary.

A naming convention for data hub topics will be utilized:

Host.Org.Corridor.AssetType.Description

An example of this might be: CT.D7.210.IntersectionSignal.ProcessedPlanInventory, where the Host is CT (Caltrans), the Org is D7 (District 7), the Corridor is 210, the AssetType is Intersection Signal, and the Description is Processed Plan Inventory. This allows for deployment in multiple districts and multiple corridors and maintains a human readable naming convention.

### 9.1.2. COMMAND MESSAGING AND ACTIVEMQ

ActiveMQ provides a robust and reliable messaging system that has been in use in production systems since 2004. It has been an Apache project since 2007, with many years of reliable operation.

The data hub uses ActiveMQ for its command messaging, providing a communication mechanism for the orchestration of services conducted by its command gateway. The general mechanism is to have a topic for services to provide status to the command gateway, and a virtual task topic used for the command gateway to send commands to the individual services. This extends also to the data gateway for

configuration of dynamic endpoints required for workflows that involve the DSS and CMS, as well as communication of requests, responses, and status information exchange with the CMS and DSS.

## 9.2. DSS INTERNAL MESSAGING

The DSS uses Apache ActiveMQ as its only messaging method. As with the data hub, it is important to note that the design of the DSS is also based on loosely coupled services connected by messaging. The primary difference from the data hub however, is that the DSS is a combination of subsystems – primarily the modeling and the response plan management components. Orchestration is not centralized. Orchestration of the modeling system is provided by the Project Manager Application. The response plan management does not require orchestration of independent services, and workflows are governed by the response plan management component. The DSS Interface provides a critical function of distribution and transformation of messages for each consuming system. A shared ActiveMQ messaging system provides all messaging for the DSS and its subsystems for data, command, and status information.

As with the data hub, the messaging only provides data transport and command communications. The services the messaging provides have no knowledge of the other services either providing the data used by those services, nor the services consuming the data they provide. Only the orchestration service provided by the Project Manager Application for modeling, or the response plan management component of the response plan management system has any knowledge of the full suite of services of their respective subsystems. The DSS interface provides the gateway to the data hub, and ActiveMQ provides the bridge between response plan management and modeling. In the same manner as the data hub, this has significant advantages in scalability, speed of processing, and flexibility, but also comes without transactionality. There is the potential for message loss that must be addressed within the workflows defined within Project Manager Application and within the design of service recovery in the event of system failure.

This page left blank  
intentionally

## 10. DEFINITION OF TERMS

Term	Definition
ActiveMQ	Apache ActiveMQ, <a href="http://activemq.apache.org">activemq.apache.org</a> is an open source software messaging system
Aimsun	Aimsun ( <a href="http://aimsun.com">aimsun.com</a> ) is a commercial traffic simulation software system.
Alert	Notification sent by the ICM system to individuals or units. Alerts may be displayed on screen, sent by email, sent by text message, sent by radio message, or sent by telephone.
Amazon Web Services (AWS)	A commercial cloud computing service - <a href="http://www.aws.amazon.com">www.aws.amazon.com</a>
API	Application Programming Interface (API) is a definition of how a software component communicates with external components
Application layer	A software architecture or design layer that provides core application functions for a software system.
Archive	Data that has been stored for historical purposes and can be retrieved upon request, usually to a location and using a storage method that has large capacity and slower retrieval times.
ATMS	Advanced Traffic Management System
Authentication	Verifying a user's identity.
Authorization	Verifying a user's permissions to view specific data elements or perform specific functions.
Availability	A description of whether an asset is available for use in a response plan or not.
Camel	Apache Camel, <a href="http://camel.apache.org">camel.apache.org</a> , is an open source enterprise integration platform, providing software components for routing of system messages.
Cassandra	Apache Cassandra, <a href="http://cassandra.apache.org">cassandra.apache.org</a> , is an open source table based NoSQL database system.
CloudFormation	An AWS service that provides the ability to code and automate AWS services provisioning and ultimately, system environments and components.
CloudTrail	An AWS service that logs user actions within an AWS environment.
CloudWatch	An AWS service that provides monitoring services within an AWS environment.
CMS	Changeable message sign. Includes both fixed and mobile devices.
Command Gateway	A component of the data hub that provides service orchestration and workflow services for the data hub and for processes that cross data hub/DSS and data hub/CMS boundaries.
Configuration Management	Maintaining a timeline of changes to an entity, ensuring traceability of changes in time, content, and author of the change.

Term	Definition
Corridor Asset	<p>Any corridor element available for use within a response plan or that provides information to the ICMS. Assets include the following types of elements:</p> <ul style="list-style-type: none"> <li>• Intersection traffic signals</li> <li>• Ramp meters</li> <li>• Organizational units or individuals (people resources)</li> <li>• Equipment</li> <li>• Mobile or stationary CMS elements</li> <li>• Traffic sensors and other measurement devices</li> <li>• Communication elements (511, HAR, third party information providers)</li> <li>• Parking facilities</li> <li>• Transit elements</li> </ul>
Corridor Management System	<p>One of the three primary ICM system components. The corridor management system provides a user interface and resulting way for all of the program stakeholders to interact with the system and provides the ability to execute response plans through the various stakeholder systems.</p>
Corridor State	<p>Information describing the state of the corridor at a specific point in time. State information includes:</p> <ul style="list-style-type: none"> <li>• Corridor road network closures</li> <li>• Corridor road network lane blockages</li> <li>• Incident information</li> <li>• Event information</li> <li>• Asset inventory</li> <li>• Asset state</li> <li>• Sensor information</li> <li>• Transit information</li> <li>• Transit state</li> <li>• Traffic conditions (density, flow, velocity) on the road network</li> <li>• Response plans currently implemented or in the process of being implemented</li> </ul>
Current Traffic State	<p>Determining a value of traffic density, flow, and velocity for each link in the road network at the current time and with the data available at the current time. Also includes values for current turn volumes and ratios at each turn movement within the road network.</p>

Term	Definition
Data Gateway	A component of the data hub that provides an interface between the data hub and the DSS as well as between the data hub and CMS systems. The data gateway also provides routing services for messages sent or received to the DSS or CMS and internal data hub components.
Data Hub	A core component of the ICM system which has primary responsibility for receiving, processing, storing, and providing data for all ICM system components.
Data Quality	A measure of the quality of data being received by the ICM system. Factors considered in data quality of a specific asset or type of assets include: <ul style="list-style-type: none"> <li>• Percent of working assets</li> <li>• Individual asset state, including level of asset degradation</li> <li>• Percent of time reliable data is provided by the asset</li> <li>• Specific filtering or algorithmic verification of incoming data specific to the asset or asset type</li> </ul>
Database Layer	A software architecture or design layer that provides long term data storage for a software system.
Decision Support System	A core component of the ICM system, providing traffic conditions, incident and event information, forecasts of traffic, proposed response plans and associated traffic forecasts, asset inventories and asset availability, maintenance information, organizational information, road network conditions, and previous corridor planning and study information to users to support corridor operations and decision making.
Delay	A measure of the typical time a traveler would experience along a route over and above the time the traveler would experience at free-flow traffic conditions.
Demand	A measure of traffic demand (flow) at an entrance to the road network or between specify entry and exit points.
DMS	Dynamic Message Sign. This is the same as a <i>CMS</i> (see above).
DMZ	A method of providing security for enterprise systems by providing a physical or logical network that separates external or public facing software system layers from private software system layers.
Do Nothing Prediction/Response	A traffic prediction based on a response plan/the response plan itself that includes no changes to any corridor assets' normal, preprogrammed, responses to traffic behavior.
Drools	An open source Java based rules engine software component. (drools.org)
EC2	An AWS service providing computing capabilities (virtual or dedicated server resources) on demand
EDW	Enterprise Data Warehouse is a type of software system that provides consolidation, reporting, and analysis services for a business, usually with multiple data sources.
Elastic Map Reduce (EMR)	An AWS service that provides hosted high volume, high speed data processing services.

Term	Definition
ensemble Kalman filter	A Monte Carlo implementation of the Bayesian update problem, used within the traffic estimation system
Event	A planned or unplanned occasion or activity occurring within the corridor that is not caused by traffic activity but affects traffic conditions. Examples include road maintenance activity, a major sports event, a public event such as a parade, and a concert or arts activity.
Geospatial	Relating to location on the earth.
Glacier	An AWS service providing archival data storage services.
Graylog	Graylog (graylog.org) is an open source software system log storage, management, and reporting services.
GTFS	General Transit Feed Specification. This is a data format used to represent transit routes and schedules on electronic maps.
HAR	Highway Advisory Radio, used for communicating to travelers.
IAM	An AWS service that provides user security services within an AWS environment (Identity Access Management)
ICM	Integrated Corridor Management
ICM Core System	The core technical functionalities of the ICM system. The Core system is comprised primarily of the decision support, data hub, and corridor management system components.
ICM Environment	All the components—including people, organizations, hardware, and software—involved in the functioning ICM system
Incident	Traffic-related incident, such as an accident or disabled vehicle.
Incident Confirmation	Positive confirmation within the system of an identified traffic incident.
Incident Identification	Identification of a traffic incident.
Inventory	A collection of assets.
JMS	Java Message Services, JMS, is a component of the Java Enterprise Edition that provides a messaging standard and components and libraries for use in software systems to implement that standard.
JSON	Javascript Object Notation - a method of describing data or javascript software objects for exchange between software systems
Jurisdiction	Geographic and asset ownership or control by a specific organizational or governmental entity.
Jurisdictional Restriction	A restriction, generally on a corridor asset or road network element, imposed by an organizational or governmental agency.
Kafka	Apache Kafka, kafka.apache.org, is an open source software messaging system



Term	Definition
LCS	Lane Control Signal. Same acronym is also used for Lane Closure System.
Link	A defined section of road.
Microservices Architecture	<p>A method of software system architecture, using a combination of software components and messaging to meet its objectives, that is characterized by the following:</p> <ol style="list-style-type: none"><li>1. Utilization of smaller, independent software components with singular or very limited functions and purpose.</li><li>2. Loosely-coupled connection of those software components by messaging, generally REST, JMS or similar technology.</li><li>3. Well defined, lightweight communications between services over network communications</li></ol>
MLLib	An Apache Spark software library that provides machine learning services within the Spark framework.
MongoDB	An open source document based NoSQL database system (mongodb.com).
n-tier	A method of software system architecture that is characterized by "n" layers of software functional breakdown. A typical n-tier architecture, characterized as 3-tier architecture has three independent software design layers - user interface, application, and database layers.
Netflix Conductor	Netflix Conductor is an open source service orchestration software system that provides workflow management and service coordination for microservice based systems.
Node	A point of connection between two or more links, often located at intersections, freeway ramp diversions or ends, changes in lane configuration, or changes in road attributes (such as speed limits).
Operational Status	The working state of a corridor asset—generally working, degraded, or not functional, depending upon the capabilities of the asset.
Persistence	Storage of information in a permanent store, such as a database or file system.
Pipeline	A series of software components connected via a well defined API that together process a stream of data.
Post-event	An event or action taken after a traffic incident and removal or release of response plan elements and after the end of the response plan duration.
Postgres	An open source relational database, <a href="http://www.postgreSQL.org">www.postgreSQL.org</a> .
Probe Vehicle	A vehicle equipped with sensors allowing them to record the position, speed, and travel direction of the vehicle at regular intervals or when coming into proximity of roadside devices.
Project Manager Application	A DSS component that provides an API to the estimation modeling components as well as management and orchestration of estimation modeling services.

Term	Definition
Real-Time Data	Real-time data denotes information that is delivered immediately after measurement. Depending on the system providing the data, this may include data that was measured a few seconds or a few minutes ago. In transportation systems, this typically means data that 15-minute old or less.
Relational Database Service (RDS)	An AWS service that provides hosted relational database services.
Response Crew	Any organizational (human and equipment) assets that respond to an incident or event.
Response Plan	<p>A collection of actions prepared and evaluated by the ICM system for implementation in response to an event or incident. Response plans may be in the following states:</p> <ul style="list-style-type: none"><li>• <i>Development</i> - The selection and assembly of response plan elements</li><li>• <i>Evaluation</i> - System generation of traffic forecast based on the response plan and analysis of the forecast and other response plan components</li><li>• <i>Proposed</i> - Recommended by the system for implementation based on the evaluation of the plan</li><li>• <i>Selection</i> - Selection of a plan to be submitted for approval</li><li>• <i>Active</i> - Approved and in implementation</li></ul> <p>Response plans may include one or more of the following deployment elements:</p> <ul style="list-style-type: none"><li>• Recommended traffic reroutes around an incident or event</li><li>• Intersection traffic signal changes</li><li>• Ramp meter changes</li><li>• Organizational asset deployments</li><li>• Equipment deployments</li><li>• CMS changes</li><li>• Communications</li></ul> <p>Required additional supporting elements of a response plan include:</p> <ul style="list-style-type: none"><li>• Approval requests and responses (if the response plan is proposed for implementation)</li><li>• Traffic state at the time of response plan development initiation</li><li>• Traffic forecast based on the response plan deployment elements</li><li>• Geographic area of impact (also known as area of influence)</li><li>• Corridor asset state at the time of response plan development initiation</li><li>• Initiating incident or event information</li></ul>

Term	Definition
	<ul style="list-style-type: none"><li>• Implementation results, including success or failure of each response plan action and traffic state information throughout the response plan duration (if the response plan is deployed)</li></ul>
Response Plan Development	Creation of one or more response plans in response to an incident or event by the ICM system.
Response Plan Implementation	Execution of response plan deployment elements.
Response Plan Manager	A component of the DSS that provides control and orchestration services for the development of response plans
REST	Representational State Transfer (REST) is a software design method that provides communication between software components via HTTP protocol.
Route	An interconnected collection of road links that create a single continuous path between any two points in the road network.
Rule	A single element of logic, expressed within a format and dialog that the rules engine can understand and process.
Rule set	A collection of rules and any instructions for their execution intended to be executed as a group within the rules engine.
Rules Engine	A core component of the ICM system that includes an off-the-shelf (commercial or open-source) software system that allows users to define, edit, or delete rules that govern specific logic applied to specific processes. The rules engine executes those rules at run time in the context of a process when the process is invoked. A rules engine is specified within the ICM system to allow users to define identification of traffic incidents, when response plans are to be developed, what response plan elements will be included within a response plan, and to allow the logic of these processes to be redefined by the users over the lifetime of the system.
S3	An AWS service provide data storage services.
Seda	An Apache Camel component that provides Staged Event Driven Architecture behavior, essentially providing a blocking queue to exchange messages between producer and consumer components within Camel.
Sensor	A corridor asset that senses and reports to the ICMS a measurement of the state of the asset or traffic.
SOAP	Simple Object Access Protocol (SOAP) is an XML based communication method for exchange of information using HTTP protocols
Spark	Apache Spark, <a href="http://spark.apache.org">spark.apache.org</a> , is an open source cluster computing platform.
SSH	Secure Shell - a cryptographic protocol for operating network services
SSL	Secure Sockets Layer - a cryptographic protocol for communicating over a computer network
TMC	Traffic Management Center

---

Term	Definition
TMDD	Traffic Management Data Dictionary, which is a standard for communications between traffic centers.
Tomcat	An open source Apache web application server ( <a href="http://tomcat.apache.org">tomcat.apache.org</a> )
Traffic Forecast	A prediction of the future state of traffic density, velocity, and flow for each link in the road network.
Traffic State	The current traffic density, velocity, and flow for each link in the road network.
Transit State	The state of one or more transit providers, including the transit inventory in operation, the working state of each asset, and each asset's location.
Travel Time	<p>The time it takes to travel between two defined points along a specified route on the traffic network. Three types of travel time can be distinguished:</p> <ul style="list-style-type: none"><li>• Point travel time—Travel time observed at a given point in time within the road network</li><li>• Predicted travel time—Expected future travel time along a given route based on a traveler or vehicle starting a trip at the current time and encountering various predicted traffic conditions along his trip</li><li>• Experienced travel time—Travel time obtained by measuring the time it actually took for a person or vehicle to travel along a given route.</li></ul>
Two-Factor Authentication	Authentication method that requires two forms of identification. A common two-factor authentication method is to use a username/password combination with an additional method, such as an additional hardware key device.
User interface layer	A software architecture or design layer that provides a visual interface and corresponding controls to interact with the software system
Virtual Private Cloud (VPC)	An AWS service providing isolation of cloud computing components into a contained, defined computing environment
Visualization	The collection and display of information by the system for the user.