|  |  |
| --- | --- |
|  |  |
| **Connected Vehicle Data Privacy Protection**  **Module** |  |
|  | June 13, 2017 |

This page intentionally left blank

# Revision History

|  |  |  |
| --- | --- | --- |
| Revision | Date | Changes |
| 0.1 | April 24, 2017 | * Initial Version |
| 0.2 | May 3, 2017 | * Added basic installation information; Added parameter explanations; updated language to reflect new name PPM. |
| 0.3 | May 15, 2017 | * Added more detail about the “why” behind the approach; added the geofence extension parameter. |
| 0.4 | May 18, 2017 | * Added a parameter and updated some text. |
| 0.5 | June 13, 2017 | * Added information about logging. * Added description of the random redaction capability. * Descriptions about running multiple instances added. |
| 0.6 | October 26, 2017 | * Added TIMs support. |

Table of Contents

[Revision History 3](#_Toc481577801)

[Table of Figures 5](#_Toc481577802)

[The Operational Data Environment (ODE) Privacy Protection Module (PPM) Background 6](#_Toc481577803)

[PPM Purpose 6](#_Toc481577804)

[PPM Limitations 6](#_Toc481577805)

[Architecture and Interaction with the ODE 7](#_Toc481577806)

[Code and Testing Environment 7](#_Toc481577807)

[Development Methodology 7](#_Toc481577808)

[Obtaining the Code 7](#_Toc481577809)

[Code Documentation 8](#_Toc481577810)

[Details about Dependencies 8](#_Toc481577811)

[Building the PPM and the Testing Environment 8](#_Toc481577812)

[Building the PPM 8](#_Toc481577813)

[Running and Testing the PPM 8](#_Toc481577814)

[Configuring the PPM 9](#_Toc481577815)

[Configuration File and Parameters 9](#_Toc481577816)

[ODE and Kafka Parameters 9](#_Toc481577817)

[Velocity Fiter Parameters 9](#_Toc481577818)

[BSM Temporary ID Redaction Parameters 10](#_Toc481577819)

[Geofencing Parameters 10](#_Toc481577820)

[Map File Specification 11](#_Toc481577821)

[Additional PPM Operational Details 12](#_Toc481577822)

[PPM Data Inputs and Outputs (Kafka Topics) 12](#_Toc481577823)

[Important Privacy Considerations and Limitations 12](#_Toc481577824)

[Appendix A: Glossary of Terms / Acronyms 14](#_Toc481577825)

# Table of Figures

[Figure 1: Interface between ODE and De-Identification 7](file:////Users/x4j/Documents/repos/jpo-cvdp/docs/ppm_user_manual.docx#_Toc481577723)

[Figure 2: Geofence around map segment 12](#_Toc481577724)

# The Operational Data Environment (ODE) Privacy Protection Module (PPM) Background

The United States Department of Transportation Joint Program Office (JPO) Connected Vehicle Data Privacy (CVDP) Project is developing a variety of methods to enhance the privacy of individuals who generate connected vehicle data. Connected vehicle technology uses in-vehicle wireless transceivers to broadcast and receive basic safety messages (BSMs) and traveler information messages (TIMs) that include accurate spatiotemporal information to enhance transportation safety. Integrated Global Positioning System (GPS) measurements are included in these messages. Databases, some publicly available, of message sequences, called trajectories, are being used to develop safety and traffic management applications. **BSMs and TIMs do not contain explicit identifiers that link trajectories to individuals or their vehicles; however, the locations they expose may be sensitive and associated with a very small subset of the population. Protecting these locations from unwanted disclosure is extremely important.** Developing procedures that minimize the risk of associating trajectories with individuals is the objective of this project.

# PPM Purpose

This module enhances the privacy protection provided to data generators participating in Connected Vehicle Safety Pilots. The PPM operates on streams of raw messages generated by the Operational Data Environment (ODE). It determines whether each individual message should be retained or suppressed (deleted) based only on the information in that message, a well-defined geographical boundary (i.e., geofence), and vehicle speed. Additionally, the procedure redacts the BSM identifier for a specified set of vehicles (TIMs do not include such an identifier).

Vehicle speed can be used to infer information besides explicit rate of travel. A vehicle that is not moving (0 speed) may indicate:

* The driver is parking or getting ready to depart.
* The vehicle is in stand-still traffic.
* The vehicle is at a traffic signal or stop sign.
* The vehicle has been involved in an accident.

Some of these reasons can be eliminated based on road type, e.g., traffic signals are not usually located on interstates. In some areas, these reasons may be common. While stopping in traffic or at a signal may not be sensitive to an individual’s privacy, where a person parks or the location of an accident may be considered privacy sensitive. Speed that exceeds posted limits is also considered private. **The PPM errs on the side of privacy, so messages having speeds above or below prescribed limits are suppressed**.

Personal information can be highly correlated with certain locations. These locations tend to be places we park or loiter, e.g., homes. Limited-access roads, e.g., highways, freeways, interstates, do not allow access to adjacent property. Although their precise definition varies some, it is highly unlikely a driver will access a location that can be used to infer their identity while driving on limited-access roads. Limited-access roads can be used to define an “inclusion zone” for broadcast messages in a way that significantly reduces the chance these messages could be linked with identifying information. Such “inclusion zones” are referred to as geofences. By eliminating messages whose locations are outside of a prescribed geofence, following a location trace to a location where personal information can be inferred is made extremely difficult. **The PPM uses a precisely defined geofence along a limited-access road to protect driver privacy.** The geofence is defined using parameters and map information.

Any message field that is unique and remains constant can be used to form sequences of messages being transmitted from the same vehicle. Even though this identifier may not be directly related to an individual or their vehicle, it can be used to consolidate all the messages from a particular vehicle. The patterns of travel may provide sufficient information to learn personal information about the driver. For this reason, the J2735 and J2945 specifications include requirements that identifiers in the Wireless Access in Vehicular Environments (WAVE) stack change periodically.

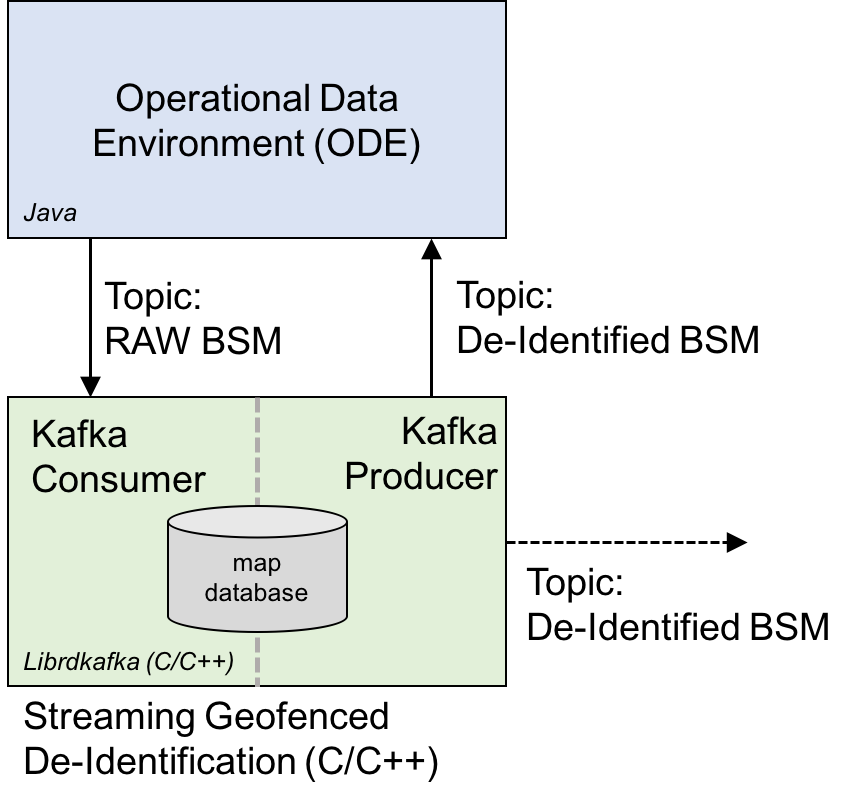
In some pilot deployments, identifiers may remain static for analysis purposes. The PPM provides a means to redact identifiers from certain vehicles. In this way, the set of vehicles whose identifiers have been redacted are made indistinguishable from one another. For either all identifiers or specified identifiers, the PPM will replace the vehicle assigned identifier with a random value. **The PPM allows a configurable set of identifiers to be redacted in a flexible way to protect driver privacy.**

# PPM Limitations

Protecting against inference-based privacy attacks on spatiotemporal trajectories (i.e., sequences of messages from a single vehicle) in general is a challenging task. An example of an inference-based privacy attack is identifying the driver that generated a sequence of messages using specific locations they visit during their trip, or other features discernable from the information in the message sequence. **This PPM treats a specific use case: a geofenced area where residences do not exist, e.g., a highway corridor, with certain speed restrictions**. **Do not assume this strategy will provide sufficient privacy protection in other uses cases.** There are alternative strategies that must be employed to handle cases where loitering locations can aid in learning the identity of the driver.

The PPM and the Operational Data Environment (ODE) use a distributed data streaming architecture called Apache Kafka. The Kafka architectures is described in detail here: <https://kafka.apache.org/documentation>. To handle very large real-time data streams, a single abstract information stream, called a Kafka topic, can be divided into partitions. Together all partitions form the single stream of information; records are not duplicated across partitions. Currently, each PPM process handles information from a single partition. This could be modified in the future if required; however, multiple partitions can be handled by launching several PPM instances each having its own configuration file.

# Architecture and Interaction with the ODE



#### *Figure 1: Interface between ODE and De-Identification*

This module performs its function as a separate compute process. It interacts with the ODE through the distributed streaming data platform, Kafka. The privacy module subscribes to an ODE topic that streams raw messages in JSON format. The privacy module filters each message it receives in this stream. It then publishes the messages it retains, with designated fields redacted in the case of BSMs, to a new Kafka topic. This simple architecture makes the privacy module independent of the ODE. Figure 1 illustrates the basic interactions between the privacy module and the ODE.

# Code and Testing Environment

## Development Methodology

The Privacy Protection Module (PPM) is being developed under Agile Development Methodologies, using an open architecture approach, in an open source environment. All stakeholders are invited to provide input to this document and the PPM. Stakeholders should direct all input on this document to the JPO Product Owner at DOT, FHWA, JPO.

This document is a living document and will be updated throughout the life of the JPO ODE project that this module is designed to support to reflect the most recent changes in the ODE design and stakeholder feedback.

## Obtaining the Code

PPM code and additional documentation is available at: <https://github.com/usdot-jpo-ode/jpo-cvdp>

PPM code and its test environment is maintained as a Git repository. The Git source code version control system and available clients is described and can be obtained at the following website: <https://git-scm.com/downloads>.

The PPM repository includes everything you need to build and test the code. The information sources below provide much more detail on PPM dependencies.

## Code Documentation

The privacy module code is documented using [Doxygen](http://www.stack.nl/~dimitri/doxygen/) and attempts to follow [Google’s C++ Style Guide](https://google.github.io/styleguide/cppguide.html). There are instructions for building the code documentation on the jpo-cvdp Github page.

## Details about Dependencies

The privacy module is written in C and C++11. It uses the following open source libraries:

* Librdkafka (<https://github.com/edenhill/librdkafka)>: an open source (BSD License) C/C++ Kafka client. Librdkafka is an active open source project with many contributors. It has been used in commercial settings.
* RapidJSON (<https://github.com/miloyip/rapidjson)>: an open source (MIT License) C++ Javascript Object Notation (JSON) parser. The library is one of the fastest open-source JSON parsers available and it is fully compliant with the RFC7159/ECMA-404.

# Building the PPM and the Testing Environment

In this manual, very basic instructions are provided to build, run, and test the PPM. This manual explains PPM configuration parameters and operations. We highly recommend using the more comprehensive build, run, and test instructions provided on the [Github page](https://github.com/usdot-jpo-ode/jpo-cvdp) for guidance on those processes.

## Building the PPM

The PPM code includes a copy of RapidJSON; however, librdkafka (see Dependencies and links to instructions above) must be installed. Also, the PPM is built using [CMake](https://cmake.org/download/). Follow the link and install that package prior to building the PPM. Once these dependencies are satisfied, the following instructions can be used to build the PPM (note: an in-source build is not necessary):

|  |
| --- |
| $ cd $BASE\_WORK\_DIR  $ git clone https://github.com/usdot-jpo-ode/jpo-cvdp.git  $ cd jpo-cvdp  $ mkdir build && cd build  $ cmake ..  $ make |

## Running and Testing the PPM

The PPM is intended to be run as a service that received data from and provides data to the ODE. It communicates using Kafka. As such, it is challenging to run and test as a stand-alone application. On the [Github page](https://github.com/usdot-jpo-ode/jpo-cvdp), there are detailed instructions outline how to standup a test environment with and without integrating with the ODE.

## PPM Logging

PPM operations are logged to two files: an information log and an error log. The files are rotating log files, i.e., a set number of log files are used to record the PPM's runtime operations. By default, these files are located in a **logs** directory rooted where the PPM is launched. The files are named **log.info** and **log.error**. The maximum size of a log.info files is 5MB and 5 files are rotated. The maximum size of a log.error file is 2MB and 2 files are rotated. Logging configuration is controlled through the command line, not through the configuration file. The command line options are described in detail on the PPM’s GitHub page.

All log entries are timestamped with the date in YYMMDD format and the time in hh:mm:ss.ssssss format. Log messages have levels or importance assigned. By using the command line options, the user can determine which messages should be logged.

### Information Log

Information log messages can be used to trace, or track, PPM operations. When the module starts up it will log the configuration it will use. For each message it consumes, the PPM will log whether it passes that message along to the output stream and what actions caused it to suppress the message or redact the BSM identifier. More information is provided in the PPM documentation on the GitHub page.

### Error Log

Ideally, the error log should be empty. If a serious error occurs, it will be written to the error log. The error may not cause the PPM to shutdown. Error messages either require configuration changes or updates to the code.

## Configuring the PPM

### Configuration File and Parameters

The PPM configuration file is a text file with a specific format. Settings in this configuration file can be used to configure the Kafka service as well as PPM operations. The Kafka-specific parameters that pertain to the PPM are discussed here; more [Kafka parameters](https://kafka.apache.org/documentation/#configuration) can be found in the [Kafka documentation](https://kafka.apache.org/documentation/).

Comments are annotated in the configuration file by starting the comment with the '#' character. Configuration lines consist of two strings separated by a '=' character; lines are terminated by newlines. Configuration files names do not follow a specific convention; however, example configuration files use the file extension, ‘.properties’ In the PPM repository, example configuration files can be found in the config directory. The following is an example of a portion of a configuration file:

|  |
| --- |
| # Configuration details for privacy ID redaction.  privacy.redaction.id=ON  privacy.redaction.id.inclusions=ON  privacy.redaction.id.included=BEA10000,BEA10001 |

The details of the settings and how they affect the function of the PPM follow:

### ODE and Kafka Parameters

* **privacy.topic.producer**: The Kafka topic name used by the PPM to write filtered messages.
* **privacy.topic.consumer**: The Kafka topic name used by the ODE (or other meesage JSON producer) consumed by the PPM. The source of the data stream to be filtered by the PPM.
* **privacy.consumer.timeout.ms**: The amount of time the consumer blocks (or waits) for a new message. If a message is received before this time has elapsed it will be processed immediately.
* **group.id**: The group identifier for the PPM consumer. Consumers label themselves with a consumer group name, and each record published to a topic is delivered to one consumer instance within each subscribing consumer group. Consumer instances can be in separate processes or on separate machines.
* **privacy.kafka.partition**: The partition(s) that this PPM will consume records from. A Kafka topic can be divided, or partitioned, into several "parallel" streams. A topic may have many partitions so it can handle an arbitrary amount of data.
* **metadata.broker.list**: This is the IP address of the Kafka topic broker leader.
* **compression.type**: The type of compression to use for writing to Kafka topics. Currently, this should be set to none.

### Velocity Fiter Parameters

* **privacy.filter.velocity**: Enable or disable message speed filtering.
  + ON : Enables message filtering.
  + Any other value : Disables message filtering.
* **privacy.filter.velocity.min**: When velocity fitering is enabled, messages having velocities below this value will be suppressed. The units are in meters per second.
* **privacy.filter.velocity.max**: When velocity fitering is enabled, messages having velocities above this value will be suppressed. The units are in meters per second.

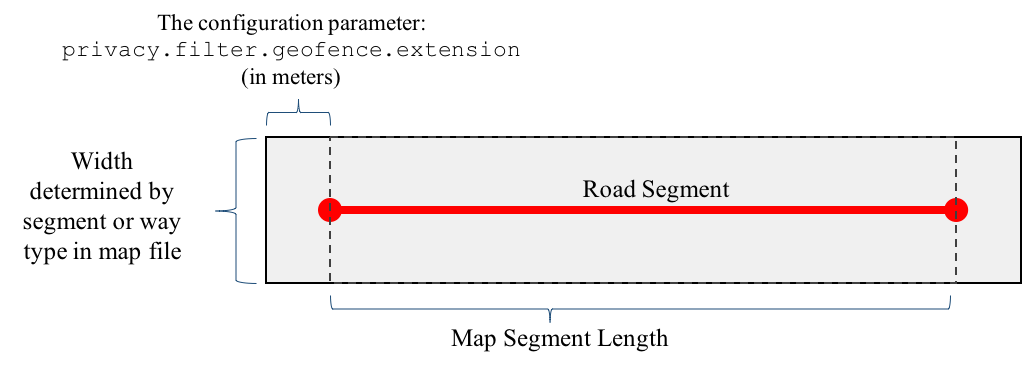
### BSM Temporary ID Redaction Parameters

BSMs have a temporary ID field. A practitioner can use temporary ID redaction parameters to control which IDs are redacted. A user can choose not to redact identifiers, redact all identifiers, or redact specific identifiers. To achieve this two levels of control provided: The first level tells the PPM whether IDs should be redacted. The second level is only active if the first level is turned on; it allows the user to specify in more detail which IDs should be redacted.

* **privacy.redaction.id**: Enable or disable the PPM's redaction function. NOTE: currently we are redacting the BSM TemporaryID field as specified in the J2735.
  + ON : enables redaction
  + Any other value : **The PPM will NOT redact BSM identifiers**.
* **privacy.redaction.id.inclusions**: If redaction is enabled, this parameter enables or disables selective id redaction.
  + ON : The PPM will use a redaction inclusion set. If a BSM is received having an id in this list, that id will be replaced with an identifier selected at random. The values in this set are defined in the privacy.redaction.id.included configuration parameter.
  + Any other value : **The PPM will redact the identifiers in all received BSMs**. In essence, the inclusion set defaults to the set of all possible 4 byte values.
* **privacy.redaction.id.included**: If redaction and redaction inclusions are enabled, this comma separated list specifies the BSM identifiers that will be redacted. BSMs having identifiers that are not in this set will not be modified.
  + Specify these ids as 4 hexadecimal-encoded byte values without quotes or prefixed with 0x.
  + Multiple identifiers can be specified by separated individual identifiers with a comma.

### Geofencing Parameters

The geofence parameters allow the user to turn geofencing on or off, specify a mapfile used to define the geofence, and further refine the geofence using a bounding box. **The geofence is not a simple rectangle or circle**. It is a complex shape determined by many individual road segments that define where vehicles of interest will be traveling. Messages transmitted by vehicles that are not on these road segments are suppressed.

* **privacy.filter.geofence** : enables or disables geofence-based filtering.
  + ON : enables the geofence.
  + Any other value : disables geofence filtering.
* **privacy.filter.geofence.mapfile** : If redaction is enabled, specifies the absolute or relative path and filename of a file that contains the map information needed to define the geofence.
* **privacy.filter.geofence.extension**: If filtering is enabled, this is one of the parmeters that determines the size of the component geofences that surround road segments.

The above figure illustrates how a “component” of the larger geofence is defined based on a single map segment. Map file attribute elements are used to define the width and length of the rectangle surrounding the map segment; this is discussed below.

The geofence is stored in a geographically-defined data structured called a quadtree. The following bounding box coordinates define the quadtree's region. The data that is stored in the quadtree is limited to those segments provided in a mapfile that are contained or intersect the bounding box region. As an example of this relationship, the coordinates specified below could bound the entire state of Wyoming; however, only the segments for the I-80 corridor would be stored within a quadtree covering Wyoming since the map file is restricted to those segments. One the other hand, the coordinates below could specify a smaller section of I-80. Then, only that subset of segments in the map file would be inserted into the quadtree and used for geofencing; the others would be ignored.

* **privacy.filter.geofence.sw.lat**: The latitude of the lower-left corner of the quadtree region.
* **privacy.filter.geofence.sw.lon**: The longitude of the lower-left corner of the quadtree region.
* **privacy.filter.geofence.ne.lat**: The latitude of the upper-right corner of the quadtree region.
* **privacy.filter.geofence.ne.lon**: The longitude of the upper-right corner of the quadtree region.

### Map File Specification

A file containing geographic elements is used to define the geofence. The file can accommodate a variety of shapes; however, in this application edges are specified. Each edge in the map file defines a single straight segment of a road. Geographically, roads have complex shapes; so, many straights edges of varying distances may be used to define a single road. Each edge is used to define a rectangular region that acts as a small “component” of the larger geofence. The map file for the I-80 WYDOT corridor is located in the jpo-cvdp/data directory; it is named: I\_80.edges

The following is a small portion of the I\_80.edges file:

|  |
| --- |
| **type,id,geography,attributes**  edge,0,0;41.24789403;-111.0467118:1;41.24746145;-111.0455124,way\_type=user\_defined:way\_id=80  edge,1,1;41.24746145;-111.0455124:2;41.24733395;-111.0451337,way\_type=user\_defined:way\_id=80  edge,2,2;41.24733395;-111.0451337:3;41.24726205;-111.044904,way\_type=user\_defined:way\_id=80  edge,3,3;41.24726205;-111.044904:4;41.24713975;-111.0444827,way\_type=user\_defined:way\_id=80 |

This file has four comma-separated elements:

* **type**: edge is used for this application.
* **id**: a unique 64-bit integer shape identifier
* **geography**: A sequence of colon-split triples representing points; each point is semi-colon split as follows: <point uid>;<latitude>;<longitude>
* **attributes**: A sequence of colon-split key=value pairs.
  + The attribute way\_type determines the dimensions of the geofence surrounding each edge.

For the WYDOT use case, WYDOT provided a set of edge definitions for I-80 that were converted into the above format.

Currently, a way\_type of undefined triggers a width and extension specifically designed for the WYDOT use case. This can be easily modified to fit a broader range of use cases and segments types.

# Additional PPM Operational Details

This application will perform rule-based de-identification. In other words, it will suppress/retain individual messages and redact BSM identifiers. Suppression will be based on the following message fields:

* Geoposition (i.e., latitude and longitude)
* Speed

The following fields will have their values redacted in an agreed upon way (randomized):

* BSM static vehicle identifiers.

The fields selected to perform suppression and redaction could be refined.

Messages records will be geofenced using the geoposition attribute, an auxiliary map database, and explicit geofences based on map road segments. The map database will contain road segments specified by WYDOT (e.g., initially the I-80 corridor). Map information can be obtained from a variety of sources. Currently, the PPM is using data provided by WYDOT. OpenStreamMap data has been used in the past. Geofences will be parameterized and the overall geofence region can be altered by adding or removing segments.

## PPM Data Inputs and Outputs (Kafka Topics)

Each PPM process subscribes to a single ODE topic and topic partition to obtain input (e.g., a Raw message stream). The PPM produces BSMs to another topic having a single partition. The output topic will have the same format as the input records with redaction / suppression occurring as described above.

## Important Privacy and Security Considerations and Limitations

The following points regarding data privacy and this system are worth considering:

* **As noted earlier in this document, the rule-based approach described here does not provide adequate privacy protection in general**. In this case, the geofenced roads do not directly connect to locations generally considered private (e.g., homes or businesses). Velocities outside of a well-defined dynamic range are considered abnormal within this geofence; therefore, we assume that behavior is due to traffic or drivers exceeding posted speed limits. The velocity rules are designed to protect against identifying vehicles involved in accidents and breaking the law.  
    
  If the geofence described here (i.e., I-80) is expanded to include additional roads, or message databases that geographically intersect this geofence are made available, additional privacy protections are needed to avoid inference-based attacks. In general, once data has been publically released it is not recallable.
* As previously discussed, the ODE will produce topics that are “sanitized” and others that are not “sanitized.” Security controls (authentication) will be needed to ensure arbitrary subscribers cannot gain access to “unsanitized” topics.
* The “raw” messages are being provided to the PPM without signatures or encryption. It is assumed the data environment is limited access since this stream may contain privacy sensitive information.
* The “filtered” messages output by the PPM are not signed or encrypted. Since these messages have been processed for privacy sensitive information, they are safer than the input stream. Currently, there is not support for data integrity (e.g., digitial signatures) on these messages; therefore, anyone could take this information and alter it. This risk is no different from using any open data source that is not signed.
* The abstractions (i.e., how and where data is stored and distributed) provided by big-data platforms like Kafka make them extremely useful and easy to scale. Understanding who has access to specific topic partitions in memory and on disk should be considered especially since this data is replicated and distributed. Theses resource could be accessible without using any of the ODE’s / Sanitization interfaces.

# Appendix A: Glossary of Terms / Acronyms

|  |  |
| --- | --- |
| Term / Acronym | Description |
| BSM | Basic Safety Message |
| TIM | Traveler Information Message |
| PPM | Privacy Protection Module |
| ODE | Operational Data Environment |
| JSON | Javascript Object Notation |
| GPS | Global Positioning System |
| CVDP | Connected Vehicle Data Privacy |
| WYDOT | Wyoming Department of Transportation |
| JPO | Joint Program Office |
|  |  |
|  |  |
|  |  |