

# Integrated Vehicle-to-Infrastructure Prototype (IVP)

## IVP Design

[www.its.dot.gov/index.htm](http://www.its.dot.gov/index.htm)

**May 20, 2016**

**FHWA-JPO-14-TBD**



U.S. Department of Transportation

Produced by Battelle Memorial Institute  
U.S. Department of Transportation  
Office of the Assistant Secretary for Research and Technology  
Federal Highway Administration, Office of Operations Research and Development

## **Notice**

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

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

---

# Technical Report Documentation Page

1. Report No. <b>FHWA-JPO-14-TBD</b>		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle Integrated Vehicle-to-Infrastructure Prototype (IVP) IVP Design				5. Report Date May 20, 2016	
				6. Performing Organization Code	
7. Author(s) Battelle				8. Performing Organization Report No. 100036384-3.3	
9. Performing Organization Name And Address Battelle 505 King Avenue Columbus, Ohio 43201				10. Work Unit No. (TRAIS)	
				11. Contract or Grant No. DTFH61-12-D-00040-T-13008	
12. Sponsoring Agency Name and Address Federal Highway Administration Office of Operations Research and Development Turner-Fairbank Highway Research Center 6300 Georgetown Pike McLean, VA 22101				13. Type of Report and Period Covered Draft Report	
				14. Sponsoring Agency Code	
15. Supplementary Notes Deborah Curtis, FHWA GTM					
16. Abstract  This report documents the Software Design Document (SDD) for the prototype development and demonstration of the Integrated V2I Prototype. This SDD is a representation of a system/software design that is to be used for recording design information, addressing various design concerns, and communicating that information to the Integrated V2I Prototype stakeholders.  This document provides a representation of the Integrated V2I Prototype software system created to facilitate analysis, planning, implementation, and decision making. It is a blueprint or model of the Integrated V2I Prototype software, communications, and to some extent, the hardware systems. The SDD is used as the primary medium for communicating design information.					
17. Key Words Vehicle-to-Infrastructure, Integrated Vehicle-to-Infrastructure Prototype Development, Integrated V2I Prototype Design			18. Distribution Statement Final Release		
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages 59	
				22. Price	

# Table of Contents

<b>Table of Contents.....</b>	<b>i</b>
<b>Executive Summary .....</b>	<b>1</b>
<b>Chapter 1 Introduction.....</b>	<b>2</b>
<b>Chapter 2 Overview .....</b>	<b>3</b>
Project Objectives .....	3
Supported Applications .....	3
User Needs .....	4
<b>Chapter 3 System Description.....</b>	<b>6</b>
System Architecture .....	6
Core Components .....	7
Plugin Components.....	7
External Subsystem Components .....	10
DSRC Radio .....	10
GPS Receiver .....	10
Position Correction Provider .....	10
UTC Provider .....	10
Traffic Signal Controller .....	10
Quasi-Static Message Storage .....	11
ITS Dynamic Message Sign .....	11
Local Weather Provider.....	11
Road Weather Data.....	11
<b>Chapter 4 Core Components .....</b>	<b>12</b>
IVP Message Router .....	12
IVP JSON Message Structure .....	12
IVP Subscription Request Message Structure .....	13
DSRC Message Manager .....	15
IVP System Monitor .....	16
Configuration and Status Database.....	18
Admin Web Interface .....	19
System Administration .....	26
The IVP Admin Web Portal API .....	26
J2735:2015 Message Library .....	28
Networking.....	28
Security .....	28
The IVP Plugin API.....	28
<b>Chapter 5 Plugin Components .....</b>	<b>29</b>
Generic Plugin Structure.....	29
Generic GPS Plugin .....	30
Position Correction Plugin .....	35
UTC Plugin .....	36
MAP Plugin .....	37
SPAT Generator Plugin.....	37
DSRC Message Manager Plugin.....	38

INFLO Plugin .....	39
<b>Chapter 6 Hardware .....</b>	<b>40</b>
<b>Chapter 7 Database Schema.....</b>	<b>42</b>
installedPlugin.....	44
pluginStatus .....	45
systemConfigurationParameter .....	45
pluginConfigurationParameter .....	46
plugin .....	47
messageType .....	48
pluginMessageMap.....	49
eventLog .....	50
messageActivity .....	50
user.....	52

## List of Tables

Table 7-1. installedPlugin Relational Database Table Columns .....	44
Table 7-2. installedPlugin Table Column Constraints.....	44
Table 7-3. installedPlugin Table Relationships .....	44
Table 7-4. pluginStatus Relational Database Table Columns.....	45
Table 7-5. pluginStatus Table Column Constraints .....	45
Table 7-6. pluginStatus Table Relationships .....	45
Table 7-7. systemConfigurationParameter Relational Database Table Columns .....	45
Table 7-8. systemConfigurationParameter Table Column Constraints.....	46
Table 7-9. pluginConfigurationParameter Relational Database Table Columns .....	46
Table 7-10. pluginConfigurationParameter Table Column Constraints .....	46
Table 7-11. pluginConfigurationParameter Table Relationships .....	47
Table 7-12. plugin Relational Database Table Columns .....	47
Table 7-13. plugin Table Column Constraints .....	47
Table 7-14. plugin Table Relationships.....	48
Table 7-15. messageType Relational Database Table Columns.....	48
Table 7-16. messageType Table Column Constraints .....	48
Table 7-17. messageType Table Relationships .....	49
Table 7-18. pluginMessageMap Relational Database Table Columns.....	49
Table 7-19. pluginMessageMap Table Column Constraints .....	49
Table 7-20. pluginMessageMap Table Relationships .....	49
Table 7-21. eventLog Relational Database Table Columns.....	50
Table 7-22. eventLog Table Column Constraints .....	50
Table 7-23. messageActivity Relational Database Table Columns .....	51
Table 7-24. messageActivity Table Column Constraints.....	51
Table 7-25. messageActivity Table Relationships .....	52
Table 7-26. user Relational Database Table Columns.....	52
Table 7-27. user Table Column Constraints .....	52

## List of Figures

Figure 2-1. Supported V2I Applications.....	4
Figure 3-1. IVP Detailed System View with Candidate Message Handlers .....	6
Figure 3-2. The IVP Core Components .....	8
Figure 3-3. IVP Plugin Components.....	9
Figure 4-1. JSON Message Structure .....	13
Figure 4-2. IVP JSON Message Class Structure.....	14
Figure 4-3. IVP Subscription Request Message.....	15
Figure 4-4. Message Router Class Structure .....	15
Figure 4-5. System Monitor Class Structure .....	17
Figure 4-6. Plugin & Support Class Structure .....	18
Figure 4-7. Database Context Class Structure .....	19
Figure 4-8. The IVP Administration Portal Login.....	21
Figure 4-9. The IVP Administration Portal.....	22
Figure 4-10. The IVP Event Log Page .....	23
Figure 4-11. The IVP Message Activity Page.....	24
Figure 4-12. The IVP System Status Page .....	25
Figure 4-13. The IVP Administration Portal User Administration Page .....	27
Figure 5-1. An "Empty" manifest file.....	29
Figure 5-2. The Manifest File for the Generic GPS Plugin .....	31
Figure 5-3. The Manifest File for the Generic GPS Plugin (cont.).....	32
Figure 5-4. The Generic GPS Status Page .....	34
Figure 5-5. The Manifest File for the Position Correction Plugin .....	35
Figure 5-6. The Manifest File for the UTC Plugin .....	36
Figure 5-7. The Manifest File for the MAP Plugin .....	37
Figure 5-8. The Manifest File for the SPAT Generator Plugin .....	38
Figure 5-9. Sample UDP DSRC Message.....	39
Figure 5-10. The Manifest File for the INFLO Plugin .....	39
Figure 6-1. Platform Hardware and Specifications .....	40
Figure 6-2. The Arada Systems' LocoMate™ GO OBU .....	41
Figure 7-1. Database Schema .....	43

# Executive Summary

This report documents the System Design Document (SDD) for an Integrated V2I Prototype (IVP) platform that is a key research activity within the DMA program.

This document presents a representation of a system/software design that is to be used for recording design information, addressing various design concerns, and communicating that information to the Integrated V2I Prototype stakeholders. A representation of the Integrated V2I Prototype software system has been created to facilitate analysis, planning, implementation, and decision making. A blueprint or model of the Integrated V2I Prototype software, communications, and to some extent, the hardware systems are presented. The structure of the hardware and software system has been structured to satisfy the requirements identified in the Integrated V2I Prototype requirements specification. It is a translation of requirements into a description of the structure and behavior of the system, the hardware and software components, the interfaces, and the data necessary for implementing the software solution.



# Chapter 1 Introduction

The Integrated application Prototype concept incorporates vehicle-to-infrastructure (V2I), infrastructure-to-vehicle (I2V), and center-to-center communications (referred to collectively as V2X). The automated V2X communications are predicated on Dedicated Short-Range Communications (DSRC) capabilities and associated infrastructure, but communications are not constrained to DSRC. For example, operations centers will communicate with each other largely through secure telecommunications networks, while existing infrastructure will communicate using established protocols over serial connections.

This report documents the System Design Document (SDD) for the Integrated V2I Prototype platform.

This SDD is a representation of a system/software design that is to be used for recording design information, addressing various design concerns, and communicating that information to the Integrated V2I Prototype stakeholders.

This document provides a description of the Integrated V2I Prototype software system created to facilitate analysis, planning, implementation, and decision making. It is a blueprint or model of the Integrated V2I Prototype software, communications, and to some extent, the hardware systems. The SDD is used as the primary medium for communicating design information.

The SDD illustrates how the hardware and software system will be structured to satisfy the requirements identified in the Integrated V2I Prototype requirements specification. It is a translation of requirements into a description of the structure and behavior of the system, the hardware and software components, the interfaces, and the data necessary for implementing the software solution.

The overall approach to this SDD is based on the guidance described in IEEE Std 1016-2009, the *IEEE Standard for Information Technology – Systems Design – Software Design Description*.

Chapter 2 presents an overview of the Integrated V2I Prototype platform, focusing on the input and output messages of the core functionality. Chapter 3 presents a description of the system components, specifically message definition and routing, communication with infrastructure components and Internet-hosted services. Chapters 4 and 5 detail the core and plugin components respectively. Chapter 6 outlines the Integrated V2I Prototype hardware components. Chapter 7 describes the details of the underlying configuration database.

# Chapter 2 Overview

## Project Objectives

The objective of the Integrated V2I Prototype (IVP) project is to identify, develop, implement, test, document and deploy a roadside prototype system that supports an integrated, interoperable deployment of multiple V2I safety, mobility, and environmental applications.

IVP is an interface system supporting the collection, integration and dissemination of data between infrastructure and vehicles for a wide variety of applications:

- Signal Phase and Timing
- Mapping (Intersections and Road Segments)
- Other Roadside Equipment (i.e. signage, detectors)
- Positioning / Corrections
- Communications (DSRC, cellular)
- Security
- Road Condition and Weather Data

It is not the intention of this document to describe the details of each V2I application itself. Rather, this document is intended to describe the IVP platform architecture capabilities to support these and other applications through an extensible message architecture. This document describes the core features of the IVP architecture, as well as the implementation interfaces to a number of useful external subsystem components to the IVP platform. This document also describes in some detail the interfaces between the listed V2I applications and the core of the IVP platform, to the extent that those applications have been implemented, in the case of several simpler, utility features of the platform, or ported, as is the case with the INFLO and SPAT applications. The details of the interfaces to external subsystem components are described in the IVP Interface Control Document (ICD), while the details of particular V2I applications can be found elsewhere.

## Supported Applications

Figure 2-1 lists the applications and application groups that the IVP platform is designed to support. Note that while the initial deployment of the IVP platform will only implement a subset of DMA applications, nonetheless the IVP platform has been designed to be readily extended to accommodate any of the listed applications using the techniques outlined in this document.

<p><b>Dynamic Mobility Applications</b></p> <ul style="list-style-type: none"> <li>• INFLO Speed Harmonization (SPD-HARM)</li> <li>• INFLO Queue Warning (Q-WARN)</li> <li>• RESCUE Incident Zone (INC-ZONE) - Low latency comm for V2V; High latency comm for V2I</li> <li>• FRATIS – High latency communications</li> </ul> <p><b>Multi-Modal Intelligent Traffic Signal System</b></p> <ul style="list-style-type: none"> <li>• Intelligent Traffic Signal System</li> <li>• Transit Signal Priority</li> <li>• Pedestrian Mobility</li> <li>• Freight Signal Priority</li> <li>• Emergency Vehicle Priority</li> </ul> <p><b>AERIS Applications</b></p> <ul style="list-style-type: none"> <li>• Eco-Signal Operations <ul style="list-style-type: none"> <li>• Eco-Traffic Signal Timing</li> <li>• Eco-Approach and Departure at Signalized Intersections</li> <li>• Eco-Traffic Signal Priority</li> <li>• Connected Eco-Driving</li> </ul> </li> <li>• Dynamic Low Emissions Zones</li> <li>• Dynamic Eco-Lanes</li> </ul>	<p><b>Transit Applications</b></p> <ul style="list-style-type: none"> <li>• Pedestrian Crossing Warning (PCW)</li> </ul> <p><b>V2I Safety Applications</b></p> <ul style="list-style-type: none"> <li>• Red-Light Violation Warning (RLVW)</li> <li>• Stop Sign Gap Assist (SSGA)</li> <li>• Curve Speed Warning (CSW)</li> <li>• Stop Sign Violation Warning (SSVW)</li> <li>• Railroad Crossing Violation Warning (RCVW)</li> <li>• Spot Weather Information Warning (SWIW)</li> <li>• Oversize Vehicle Warning (OVW)</li> <li>• Reduced Speed Zone Warning (RSZW) – Speed Reduction and Lane Closure Advisories</li> <li>• Reduced Speed Zone Warning (RSZW) – Lane Closure Alerts &amp; Warnings</li> </ul> <p><b>Road Weather Connected Vehicle Applications</b></p> <ul style="list-style-type: none"> <li>• Enhanced Maintenance Decision Support System (MDSS).</li> <li>• Information for Maintenance and Fleet Management Systems.</li> <li>• Weather-Responsive Traffic Management.</li> <li>• Motorist Advisories and Warnings.</li> <li>• Information for Freight Carriers.</li> <li>• Information and Routing Support for Emergency Responders.</li> </ul>
--	--

Source: Battelle

**Figure 2-1. Supported V2I Applications**

## User Needs

Multiple applications require a local communication and computational/processing platform for:

- Message Handling across Multiple Interfaces
  - Integrating data from multiple sources and compiling messages for delivery to vehicles and drivers and nomadic devices via multiple communication methods
  - Obtaining and aggregating data from multiple vehicles and nomadic devices and delivery to Traffic Management Entity
  - Distribution of TME messages to local vehicles
- Local Infrastructure Based Computation and Processing:

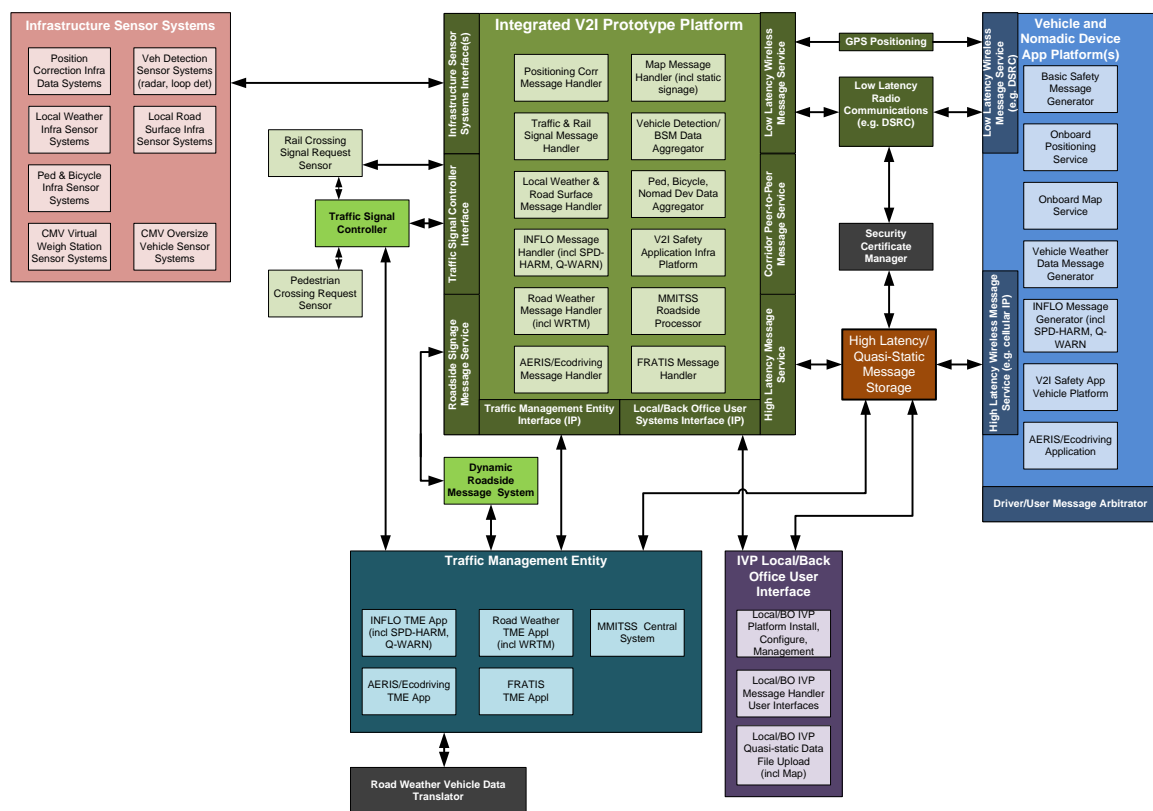
- e.g. Local computation of safe speeds and safe stopping distances using real time weather and road condition data for crash imminent V2I safety scenario such as Reduced Speed (Work Zone) Warning and Spot Weather Information Warning
- e.g. Aggregation of vehicle weather data for efficient communication to Traffic Management Entity for Weather Responsive Traffic Management
- e.g. MMITSS “Intersection level” functions including MAP and SPAT broadcast manager, equipped vehicle tracker, priority request server, and interface to traffic signal controller

# Chapter 3 System Description

The Integrated system architecture is described in the *Integrated V2I Prototype System Requirements and Architecture Document*. High-level details are repeated here for readability. For more detailed architecture information please consult the architecture.

## System Architecture

Figure 3-1 illustrates the physical components of the Integrated V2I Prototype system.



Source: Battelle

**Figure 3-1. IVP Detailed System View with Candidate Message Handlers**

The components of the IVP architecture fall into two main categories:

- Core Components that are responsible for providing generic message routing capabilities
- Administrative details of the IVP platform, and Extension Modules, or Plugins, which implement the functionality of specific V2I applications.

## Core Components

The core components of the IVP platform architecture are application neutral blocks of functionality whose primary functions are to manage message routing among the application-specific components and support the infrastructure upon which the application specific components reside.

The core components are:

- IVP Message Router – The C/C++ program that implements the Publish/Subscribe features of the IVP platform
- IVP System Monitor – The C/C++ program that oversees the behavior of the IVP platform, including plugin status, user access, and overall system health
- Configuration and Status Database – A MySQL database that holds status information about each plugin, user accounts, and current system status exposed through the Admin Web Portal interface
- Admin Web Portal – The PHP web portal through which a remote user is able to configure, monitor, and manage the IVP platform
- J2735:2009 Message Library – A support library that translates ASN.1 payloads that originate from and are destined to the DSRC radio into machine readable formats
- FHWA Battelle SPAT Message Library – A support library that translates Battelle-specific SPAT messages into machine readable formats
- IVP Plugin API – A library of entry points into the IVP system code that simplifies development of plugin modules and allows access to system resources on the platform
- SSH Interface – Enables secure, remote access to the IVP platform resources.

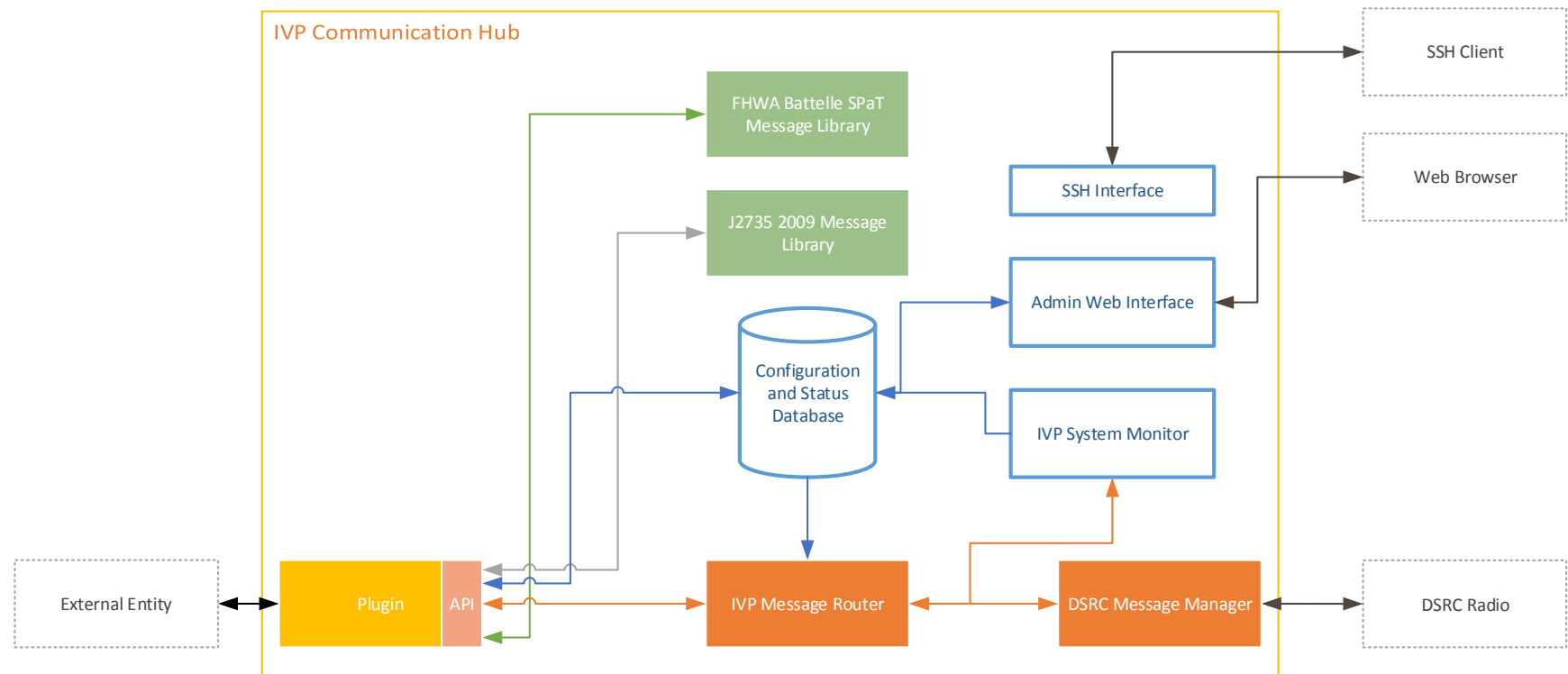
These components, illustrated with their connections to each other in Figure 3-2, are described in detail in Chapter 4.

## Plugin Components

Plugin modules are the application specific pieces of code on the IVP platform. Plugin modules are responsible for processing data extracted from external peripheral components, and generating status or other information that is published to the IVP router, or, controlling or communicating with external components based on processing performed from messages received from other application plugins installed on the IVP platform.

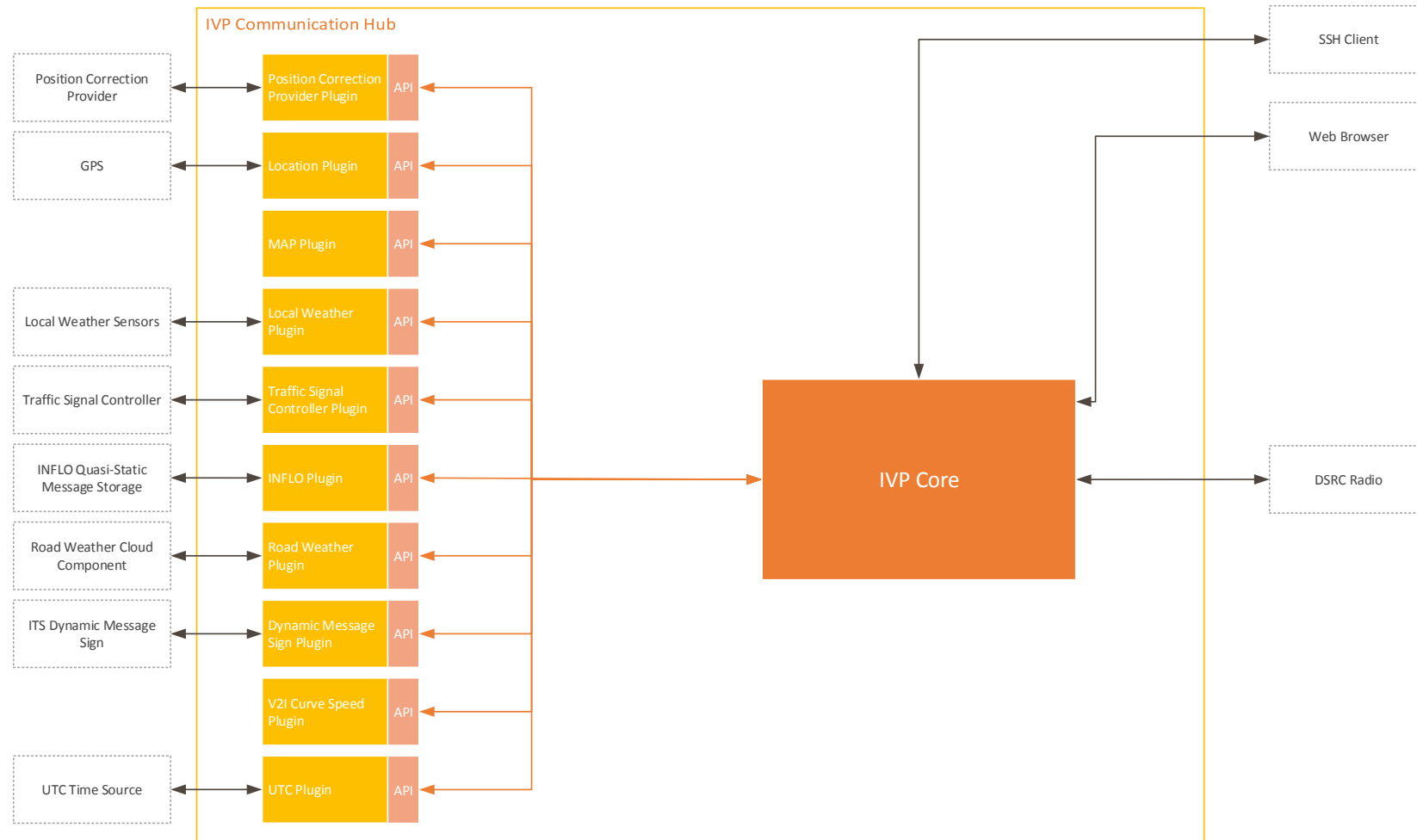
The plugin architecture addresses the user need for efficient computation and processing in the local infrastructure. The plugin API provides access to messages and data in an efficient manner. For example, a specific message routed through the system only needs sent once, but can be received by multiple plugins for further processing.

The currently implemented plugin components, shown with their connections to the core components of the IVP platform in Figure 3-3, are described in detail in Chapter 5.



Source: Battelle

**Figure 3-2. The IVP Core Components**



Source: Battelle

**Figure 3-3. IVP Plugin Components**



## External Subsystem Components

The IVP Platform interacts with a number of external subsystems through software plugin modules. This design document is intended to describe the interactions of the IVP platform with those plugin modules. The details of the interfaces between the plugin modules and the external components described in this section are presented in the IVP Interface Control Document (ICD). The following is a brief description of the external components which the IVP platform that the initial development of the IVP project is intended to connect with.

### DSRC Radio

The Dedicated Short Range Communications (DSRC) radio is the primary means of achieving V2I communications for the IVP platform. As a component of equipment positioned along highways, at traffic intersections and other locations to support wireless communications between connected vehicles and infrastructure, the IVP platform's DSRC functionality is a critical feature of the RSU configuration demonstrated, but which may be implemented in many different ways that are not specified. The DSRC RSU must adhere to the RSU 3.1 specification.

### GPS Receiver

A Global Positioning System (GPS) receiver attached to the IVP platform provides both location information and a precise time source for system that are deployed without continuous internet connectivity. Systems deploy with reliable internet connectivity may substitute a surveyed location plugin for GPS equipment, and use an internet-based NTP provider as the precision UTC time source required for system operation.

### Position Correction Provider

A source of data from a network of base stations providing RTCM differential corrections for location and time position that allows rover to calculate a more accurate determination of position than an unassisted rover typically offers. Position data may be obtained from a server via an internet connection, or from a DSRC source using standardized J2735 messages to encapsulate position corrections.

### UTC Provider

If reliable internet connectivity is available, a precise source of Coordinated Universal Time (UTC) is provided by an internet-based Network Time Protocol (NTP) provider. If no reliable internet connectivity is available, an accurate source of UTC time may be delivered by external GPS equipment.

### Traffic Signal Controller

The initial implementation of the IVP platform utilize the Econolite TSC connected to the IVP platform. As developed in previous work the TSC will transmit relevant phase and timing information on regular intervals to the IVP platform. The TSC must send the Traffic Signal Controller Broadcast Message as defined in Table 4 of document 60606-018A\_SPaT\_ICD FINAL.

## **Quasi-Static Message Storage**

The Quasi-Static Message Storage (QMS) component's role may be occupied by a Traffic Management Entity (TME) or Traffic Management Center (TMC)-like entity. For the BCO demonstration, the QMS will be the recipient of INFLO messages for both the Queue Warning (Q-WARN) and Speed Harmonization (SPD-HARM) applications.

## **ITS Dynamic Message Sign**

A Dynamic Message Sign (DMS) is an electronic traffic sign on roadways used to give travelers information about special events. These events include traffic congestion, accidents, incidents, roadwork zones, or speed limits. During system development, a simulated DMS will be utilized.

## **Local Weather Provider**

The Local Weather Provider component may be fulfilled by an Environmental Sensor Station (ESS) consisting of meteorological instruments connected to the IVP platform, or by internet-based road weather information system (RWIS) providing regional road weather data.

## **Road Weather Data**

Plugins can be created to communicate directly with a Road Weather Information System (RWIS) to collect road weather data from environmental sensors. This data can be used locally by the plugin or routed through the IVP system for processing by other plugins.

# Chapter 4 Core Components

## IVP Message Router

The IVP Message Router is the central node for all communications in the IVP Communication Hub. The IVP Message Router allows plugins to connect to it, subscribe to messages, and publish messages for sharing to other plugins or external via the DSRC radio. The IVP Message router also has dedicated connections to the DSRC Message Manager for transmission over the DSRC Radio, and to the IVP System Monitor for system analysis and statistics. The IVP System Monitor logs key events to the database, but does not log messages. Message logging can be enabled on the RSU for transmission logs.

Plugins subscribe to data messages by message type. Messages are received by the plugin via a TCP / IP interface over a configurable port. The IVP Message Router forwards all messages of a message type to plugins which have subscribed to messages of that message type. The IVP Message Router also connects to the DSRC Message Manager and forwards all messages that have the `dsrc_routing` flag set to true, regardless of message type. Both low latency and high latency messages are routed to the DSRC radio when the `dsrc_routing` flag is true. The DSRC Message Manager has a queue to arbitrate multiple concurrent messages sent to the DSRC radio. Messages are sent in the order received.

The IVP Message Router receives incoming messages from the DSRC Message Manager. The IVP Message Router forwards all messages regardless of type to the Message Monitor.

The IVP Message Router reads configuration parameters from a Configuration Database. Plugin properties such as subscribed messages types, message properties such as whether to route to the DSRC Message Manager, as well as various system characteristics, are stored in the Configuration Database.

### Configuration Parameters

- Max Plugins (int)
- Plugin Port (int)

## IVP JSON Message Structure

Internal communications between plugins in the IVP Communication Hub use the IVP JSON Message Structure, as sample of which is illustrated in Figure 4-1. The message allows any type of payload formatted as a string to be sent through the IVP Communication Hub. The Header contains the needed information for recipients to decode the payload. The fields in the header are required. They are:

- source – String representing the creator of the message. Must be unique
- type – String representing the parent type of the payload. Examples: J2735, NEMA, etc.
- sub-Type – String representing the sub type of the payload. Examples for a J2735 message: RTCM, MAP, SPAT, etc.

- encoding – String representing the encoding of the payload. Examples: ASN.1, PER, BER, JSON, ASCII, etc.
- timestamp – UTC Timestamp in milliseconds from epoch, represented as a long
- dsrc\_routing – Boolean letting the IVP Message Router know if the message is to be sent out the DSRC radio

```
{
  "header": {
    "source": "Position Correction Plugin",
    "sub-type": "RTCM",
    "type": "J2735",
    "encoding": "ASN.1",
    "timestamp": 1404995791000,
    "dsrc_routing": true,
  },
  "payload": "data goes here"
}
```

**Figure 4-1. JSON Message Structure**

The payload of the message encoded in whatever format is required of the message definition. The IVP JSON Message class structure is shown in Figure 4-2.

## IVP Subscription Request Message Structure

When a plugin is installed on the IVP platform, the plugin must communicate with IVP Message Router to inform it of the messages it wishes to subscribe to. The IVP Subscription Request message is sent from a plugin on successful connection to the IVP Message Router. The message contains a list of messages that a plugin wants to receive. An example is shown in Figure 4-3.

- messages – A list of message type names
- message – A message type name
- plugin\_name – The name of the requesting plugin

The Message Router class structure is shown in Figure 4-4.

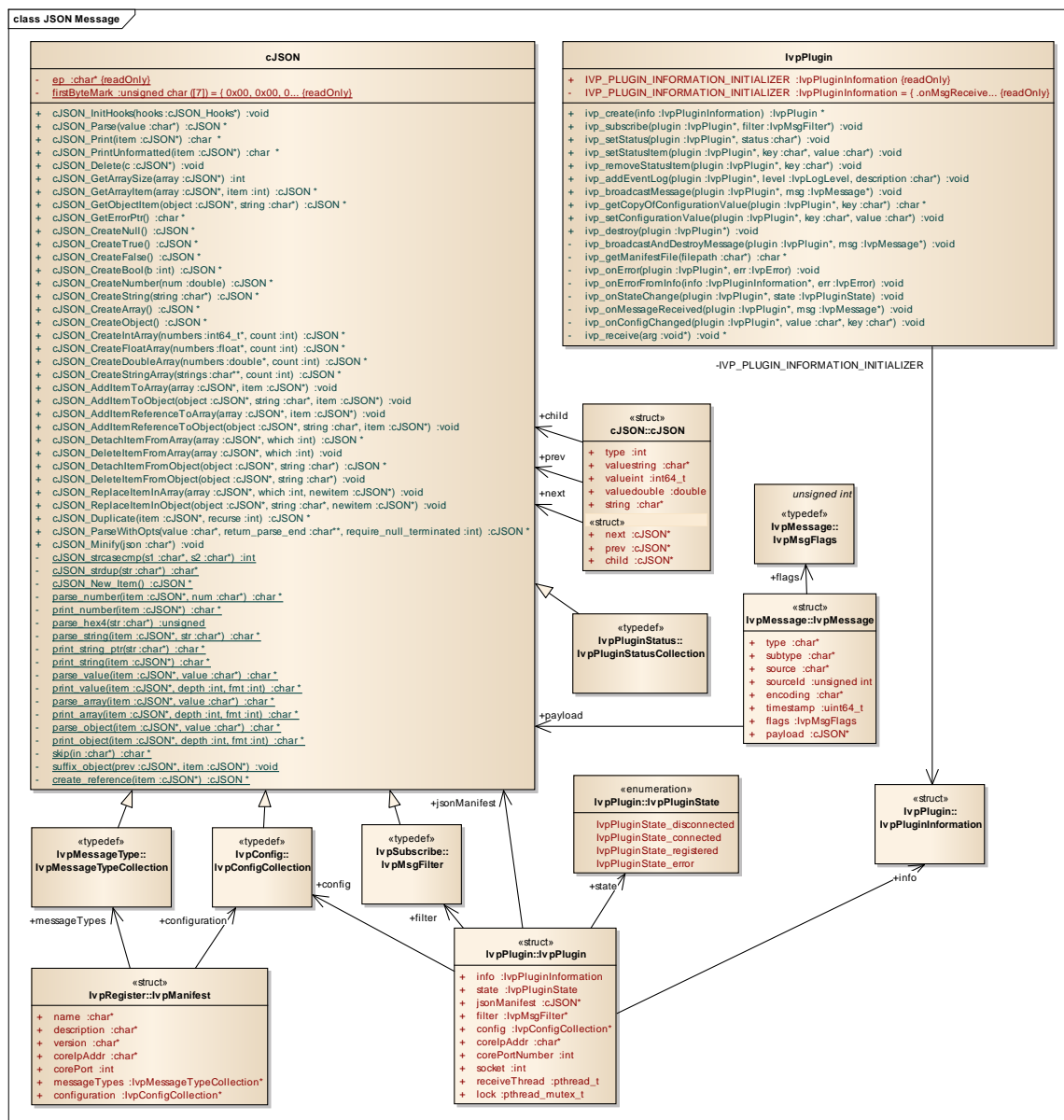
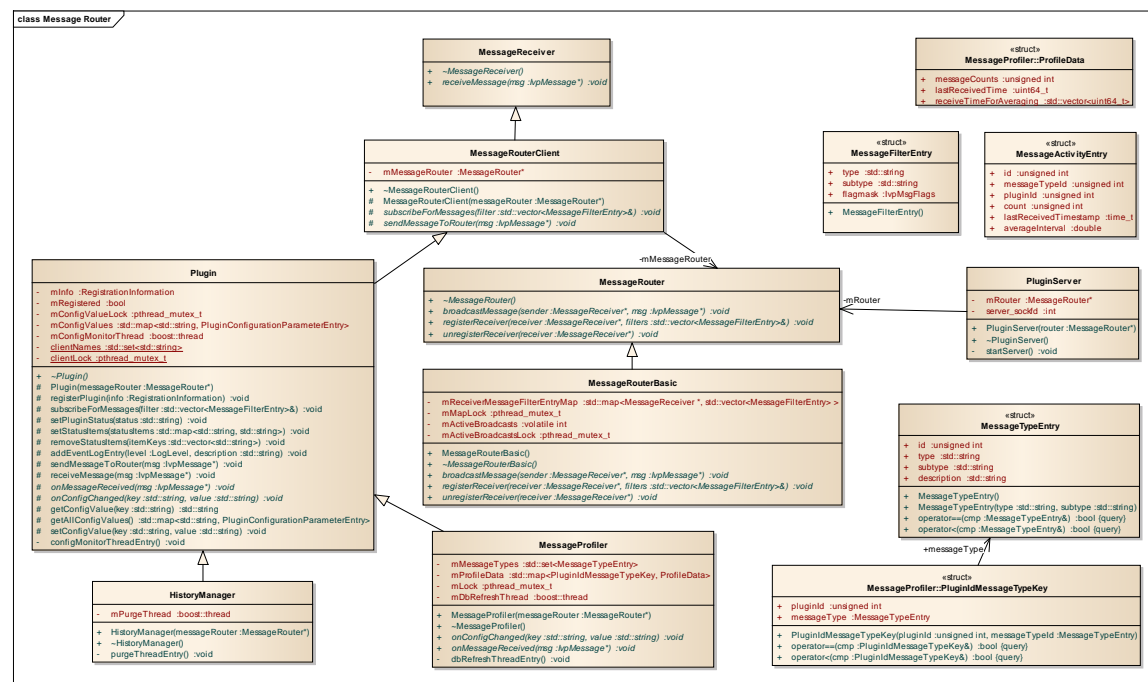


Figure 4-2. IVP JSON Message Class Structure

```
{
  "messages": [
    "message type name 1",
    "message type name 2"
  ],
  "plugin_name": "My Plugin Name"
}
```

### Figure 4-3. IVP Subscription Request Message



### Figure 4-4. Message Router Class Structure

## DSRC Message Manager

The DSRC Message Manager handles the communications to and from the IVP Message Router and DSRC Radio. The DSRC Message Manager handles the packaging and un-packaging of DSRC messages for the IVP Communication Hub. Any DSRC messages that need rerouting back through the DSRC radio are handled by the DSRC Message Manager.

The DSRC Message Manager has a dedicated connection to both the IVP Message Router and the DSRC Radio. The DSRC Message Manager receives messages from the IVP Message Router, and prepares them to be sent to the DSRC radio, ensuring only properly formatted DSRC messages are sent to the DSRC radio. The DSRC Message Manager receives incoming DSRC messages from the DSRC radio, unpacks the messages, formats them into the IVP JSON message format described in

Figure 4-1, and sends the results to the IVP Message Router. The DSRC Message Manager relays any DSRC messages from the DSRC radio that are flagged as “relay”.

## IVP System Monitor

The IVP System Monitor is an application running on the IVP Communication Hub, a part of the IVP Core, which monitors the status of each installed plugin in the system. It receives all data transmissions sent through the IVP Message Router, analyses the data, and updates statistics per message per plugin. If the IVP Message Router detects that a plugin has not been active for some time, it will restart that plugin automatically.

The IVP System Monitor has a dedicated connection to the IVP Message Router and receives all data transmitted through the IVP Message Router. The IVP System Monitor logs the number of messages received and last received timestamp per message type since last boot per plugin in the messageActivity table in the Configuration and Status Database. The IVP System Monitor logs the average time between messages in the messageActivity table by message type and source. The IVP System Monitor monitors data message output of each plugin, and restarts a plugin if no message is received in the configurable amount of time. The IVP System Monitor logs every startup time per plugin. The IVP System Monitor monitors the event log and removes messages based on a configurable max size. The IVP System Monitor starts the plugins on start up after a delay time so that the “core” is up and running.

### Configuration Parameters

- Max Event Log Size

The class supporting the IVP System Monitor and Plugin Support are shown in Figure 4-5 and Figure 4-6.

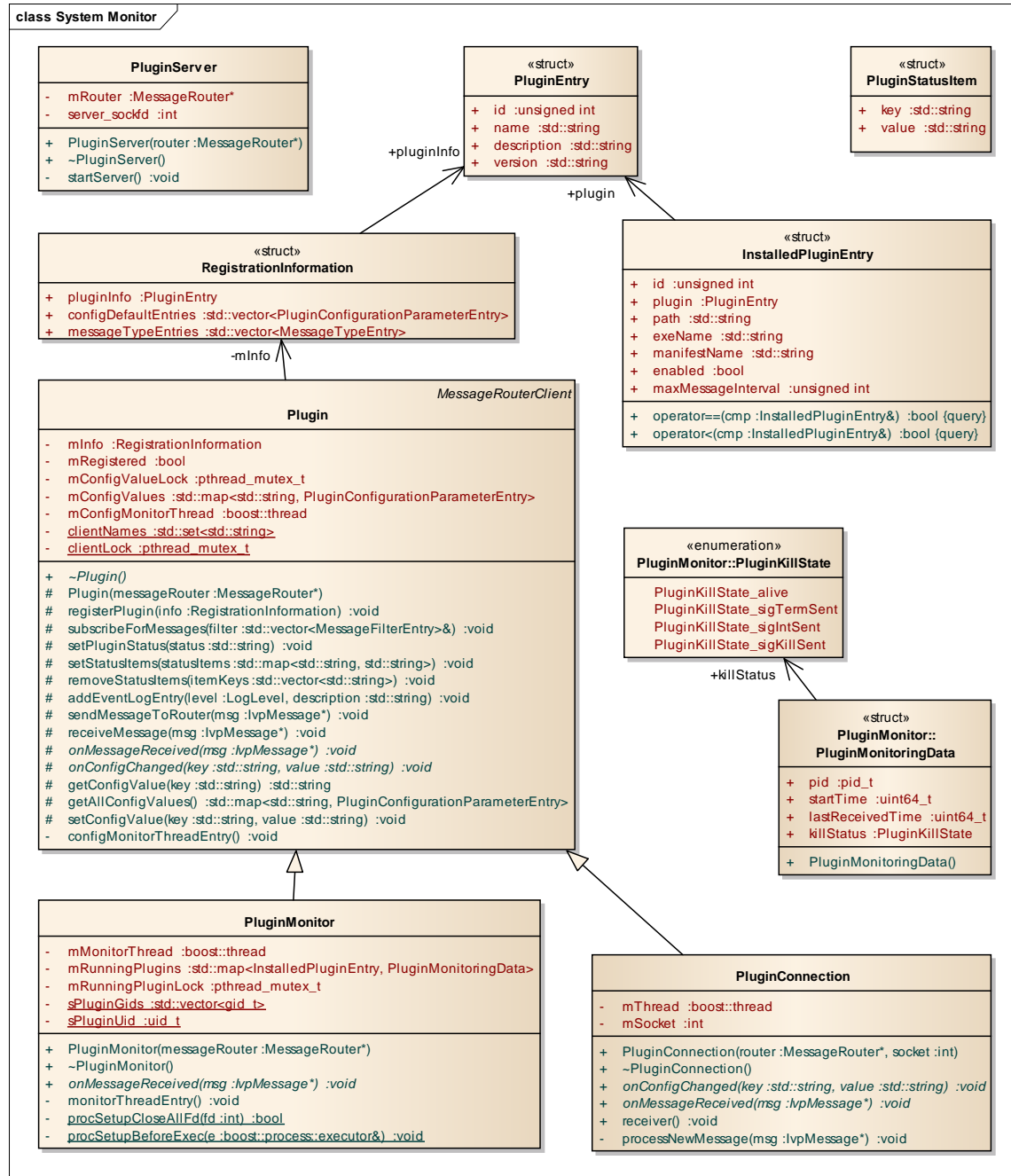


Figure 4-5. System Monitor Class Structure



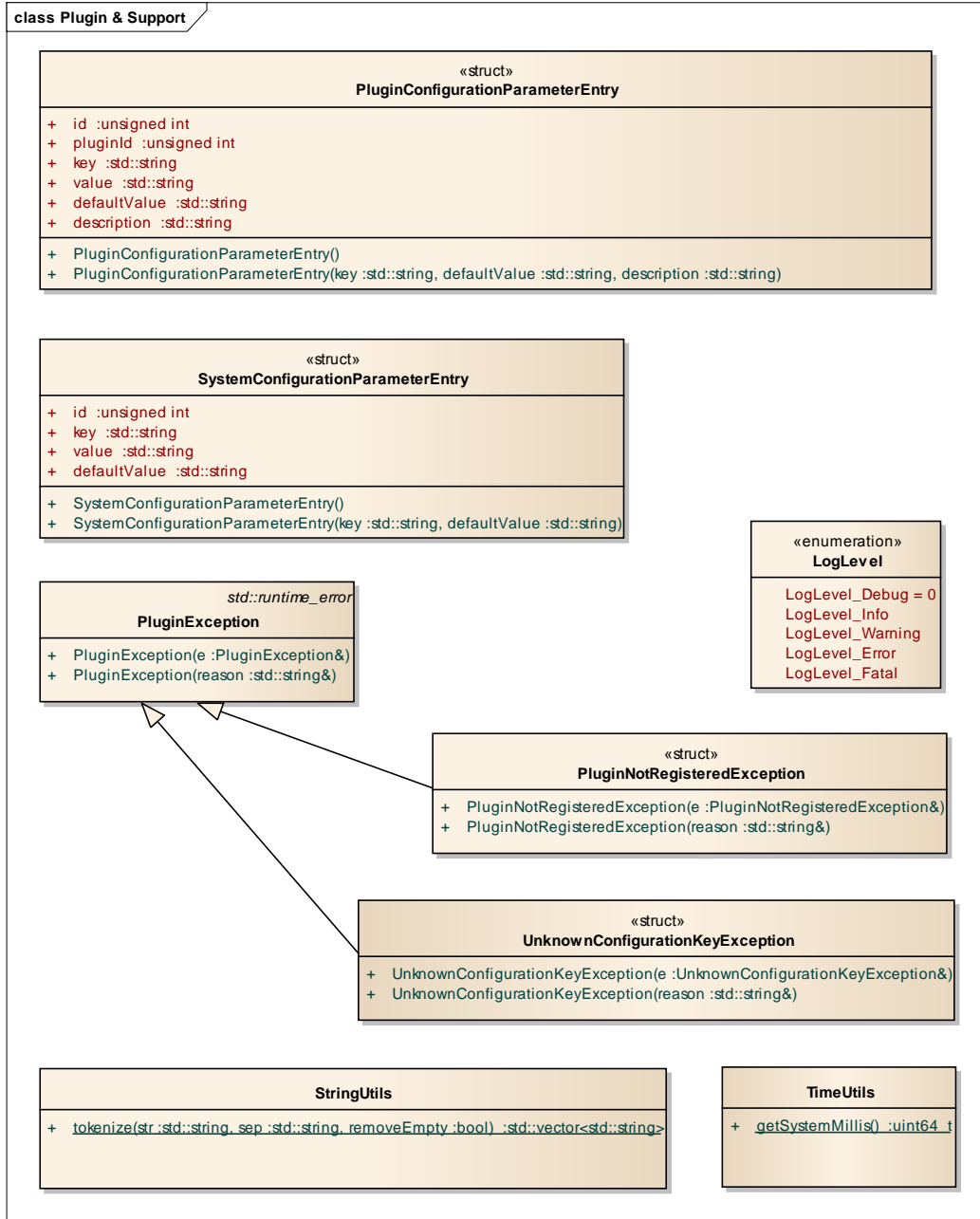


Figure 4-6. Plugin &amp; Support Class Structure

## Configuration and Status Database

The Configuration and Status Database is the database for the IVP Communication Hub. Entries for each of the plugins installed on the IVP Communication Hub are recorded in the database along with their configuration and the configuration of the IVP Core. Statistics and status are also stored in the Configuration and Status Database which show a snapshot of the data being sent through the system.

Figure 4-7 shows the classes implementing the access layer to the MySQL configuration and status database.

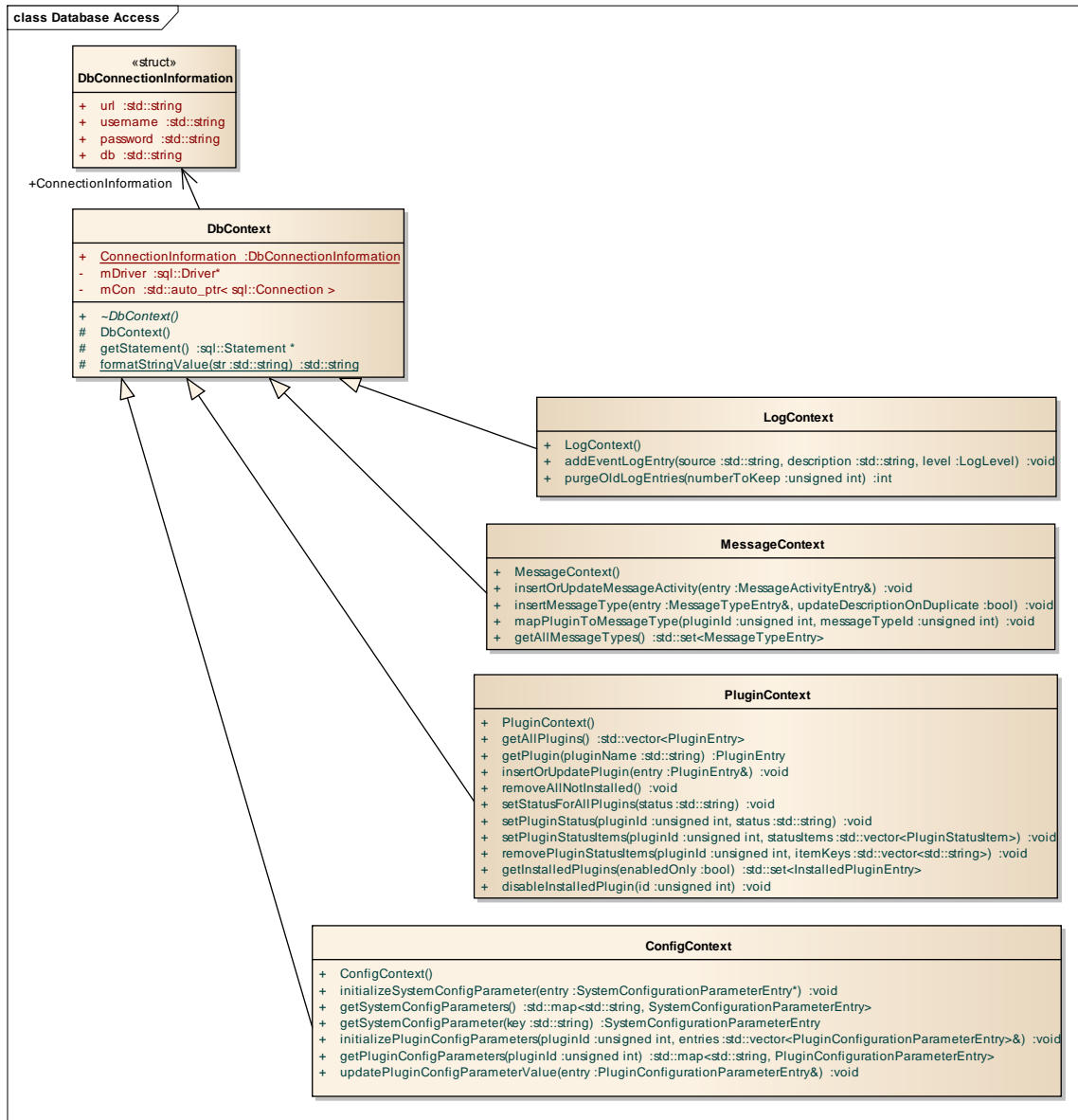


Figure 4-7. Database Context Class Structure

## Admin Web Interface

The Admin Web Interface has two distinct functions. The administration side allows the deployment of new plugins, the configuration of plugins on the system including enabling the plugin and any parameters needed, and the configuration of the system. Another function of the administration side of the Admin Web Interface is the ability to remotely reboot the system. The status side of the Admin

Web Interface shows the current status of the system including the plugins currently running, and statistics on the data being sent by each plugin.

The Admin Web Interface has an authentication method to limit access to this functionality of the IVP Platform. The Admin Web Interface authenticates against credentials in the Admin Users table of the Configuration and Status Database.

The Admin Web Interface allows a user with admin privileges to deploy and configure new plugins to the IVP Platform. The Admin Web Interface allows the configuration the name, location, and command line parameters for a new plugin. These configuration items are subsequently stored in the Configuration and Status Database.

The Admin Web Interface lists all plugins installed on the system, and their status. Plugins can be enabled and disabled through the Admin Web Interface. The Admin Web Interface allows a user with admin privileges to reboot the system. The Admin Web Interface displays information from the Message Activity Table in the Configuration and Status Database.

The web browser can be any major web browser (Internet Explorer, Firefox, and Chrome) used to view the Administration Web Interface of the IVP Communication Hub.

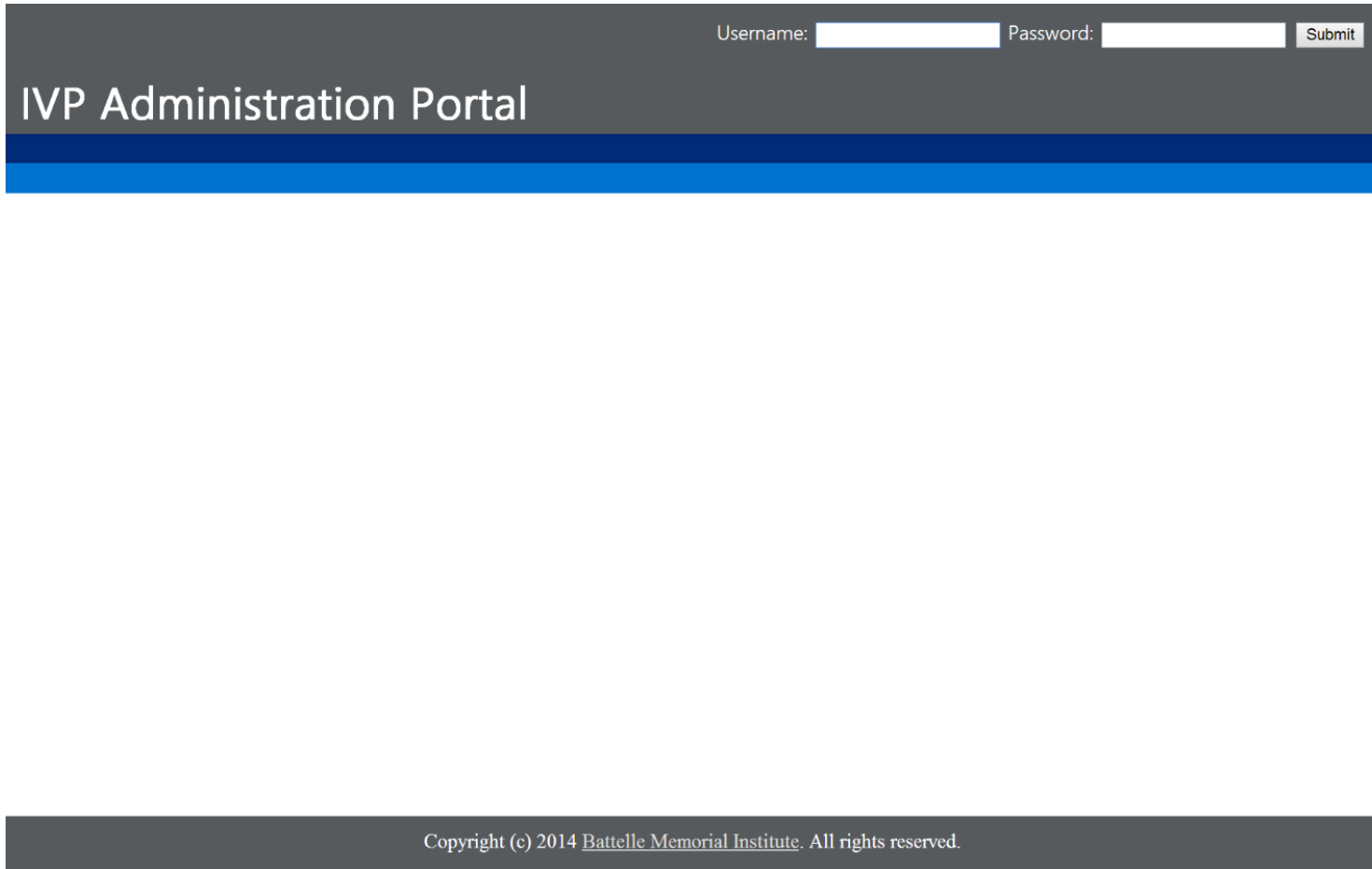
User interactions with the web portal are divided into three access levels, as follows:

#### View only

- Login to portal: The user is presented with a form for entering a username and password. (See Figure 4-8)
- Logout of portal: Logs the user out and returns to the login page. (See Figure 4-9)
- View event log: A user may view log data from oldest to most recent, displaying the log severity, the source plugin module of the message, and the log message description, color coded by severity:
  - Warning: Black font on canary yellow (#FFFF99) background
  - Error: Black font on salmon pink (#FF9999) background
  - Fatal: Tomato red (#FF4444) font on a black background
  - Data for this portal display is refreshed from the IVP MySQL database every two seconds. (See Figure 4-10)
- View message activity: A user may view a summary of all log messages recorded in the IVP MySQL database. Each message displayed includes the name of the plugin module that issued the message, the message type and subtype, a count of the number of messages seen since the last restart of the IVP core, the data/time of the last message seen, and the average interval between messages. (See Figure 4-11)

#### Application Administration

- View IVP system status: The system status includes the name, description, version, whether or not the plugin is enabled, and status for each plugin installed on the local IVP system. (See Figure 4-12)
- View module status: Enables an application administrator to view the configuration parameters and status fields for a single plugin module. Configuration parameters are displayed with their default and current values. Status fields are listed as key/value pairs, where the key is the unique identifier within the plugin for the status, and the value is the key's current value. All values for module status are extracted from the underlying IVP MySQL database.



Username:  Password:

# IVP Administration Portal

Copyright (c) 2014 [Battelle Memorial Institute](#). All rights reserved.

**Figure 4-8. The IVP Administration Portal Login**

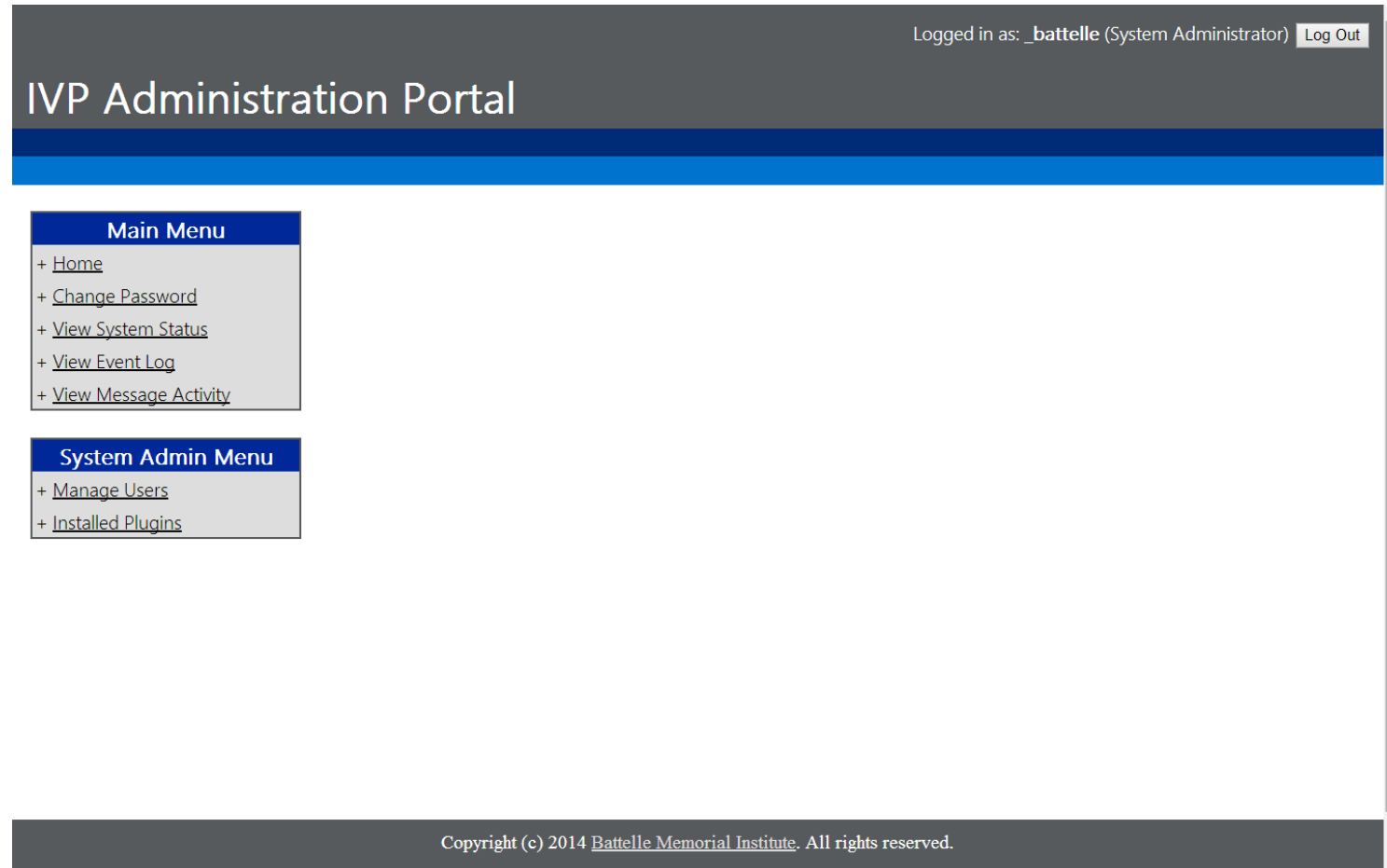


Figure 4-9. The IVP Administration Portal

Logged in as: **\_battelle** (System Administrator) [Log Out](#)

## Event Log

**Main Menu**

- + [Home](#)
- + [Change Password](#)
- + [View System Status](#)
- + [View Event Log](#)
- + [View Message Activity](#)

**System Admin Menu**

- + [Manage Users](#)
- + [Installed Plugins](#)

Level	Source	Description	Timestamp
Info	ivpcore.PluginMonitor	Plugin registered	2014-08-22 02:14:13
Info	ivpcore.MessageProfiler	Plugin registered	2014-08-22 02:14:13
Info	ivpcore.HistoryManager	Plugin registered	2014-08-22 02:14:13
Info	Ivp Core	IVP Core shutting down: Received SIGTERM	2014-08-22 02:14:12
Debug	-----	-----	2014-08-22 02:14:12
Info	Ivp Core	IVP Core Starting	2014-08-22 02:14:12
Info	ivpcore.PluginMonitor	Plugin registered	2014-08-22 02:13:57
Info	ivpcore.MessageProfiler	Plugin registered	2014-08-22 02:13:57
Info	ivpcore.HistoryManager	Plugin registered	2014-08-22 02:13:57
Debug	-----	-----	2014-08-22 02:13:55
Info	Ivp Core	IVP Core Starting	2014-08-22 02:13:55
Info	Ivp Core	IVP Core shutting down: Received SIGTERM	2014-08-22 02:13:52
Warning	Generic Gps	Plugin deconstructing	2014-08-22 02:13:40
Info	Generic Gps	Plugin registered	2014-08-22 02:13:00
Info	ivpcore.PluginMonitor	Plugin registered	2014-08-22 02:12:58
Info	ivpcore.MessageProfiler	Plugin registered	2014-08-22 02:12:58
Info	ivpcore.HistoryManager	Plugin registered	2014-08-22 02:12:58
Debug	-----	-----	2014-08-22 02:12:57
Info	Ivp Core	IVP Core Starting	2014-08-22 02:12:57
Info	Ivp Core	IVP Core shutting down: Received SIGTERM	2014-08-22 02:08:11

Showing 1 to 20 of 20 records

[Clear Log](#)

Copyright (c) 2014 Battelle Memorial Institute. All rights reserved.

Figure 4-10. The IVP Event Log Page

Logged in as: **\_battelle** (System Administrator) [Log Out](#)

## IVP Message Activity

### Main Menu

- + [Home](#)
- + [Change Password](#)
- + [View System Status](#)
- + [View Event Log](#)
- + [View Message Activity](#)

### System Admin Menu

- + [Manage Users](#)
- + [Installed Plugins](#)

Plugin Name	Message Type	Message Subtype	Count	Last Timestamp	Average
Generic Gps	NMEA	VTG	39	1970-01-01 00:03:49	0
Generic Gps	NMEA	RMC	39	1970-01-01 00:03:49	0
Generic Gps	NMEA	GSV	39	1970-01-01 00:03:49	0
Generic Gps	NMEA	GSA	40	1970-01-01 00:03:50	0
Generic Gps	NMEA	GGA	40	1970-01-01 00:03:49	0

1 Showing 1 to 5 of 5 records

Copyright (c) 2014 [Battelle Memorial Institute](#). All rights reserved.

Figure 4-11. The IVP Message Activity Page

# IVP System Status





**Main Menu**

- + [Home](#)
- + [Change Password](#)
- + [View System Status](#)
- + [View Event Log](#)
- + [View Message Activity](#)

## System Admin Menu

- + Manage Users
- + Installed Plugins

## Plugins

Plugin Name	Description	Version	Status	Enabled	
Generic Gps	Exposes GPS data from a device's NMEA stream. Also has the ability to spoof a NMEA stream using configurable lat/long values	0.0.3	Stopped / Disconnected		
ivpcore.HistoryManager	Core element that is responsible for purging old log and history data	0.0.2	Running		
ivpcore.MessageProfiler	Core element that is responsible for profiling the statistics of received messages	0.0.2	Running		
ivpcore.PluginMonitor	Core element that is responsible for starting/stopping installed plugins and monitoring the status of the plugins	0.0.2	Running		

1 Showing 1 to 4 of 4 records

Copyright (c) 2014 Battelle Memorial Institute. All rights reserved.

**Figure 4-12. The IVP System Status Page**



## System Administration

- Clear Event Log: Allows a system administrator to completely clear the IVP event log, without restarting the IVP core. The administrator is prompted for a confirmation before this action is taken.
- Change access level: Allows an administrator to promote or demote the access level of a user. The valid access levels recognized by the IVP platform are:
  - View Only – Allows the user the ability to view system and plugin status only
  - Application Administration – Provides the user with the ability to install, remove, and manage plugin modules on the IVP platform
  - System Administration – Elevates the user to have access to user administration functions.
- Change password
- Create new user account: Allows a system administrator to create a new user account. The administrator must specify the new account name, typically an email address, and select the new user's access level, as described above. (See Figure 4-13)
- Edit user account
- Remove user account: Enables a system administrator to delete a user's account from the system. The system administrator is prompted for confirmation before deleting the specified account.
- Reset password: Enables a system administrator to set the password of any user defined in the system to a random 8 character string consisting of only lower case letters and/or the digits from 0 to 9.
- Display user account list: The current user is shown a list of all user accounts defined in the system, along with each user's access level and operations.

## The IVP Admin Web Portal API

In order to provide a more secure interface for the IVP platform, all access to the IVP Configuration and Status Database from the portal pages is mediated by the IVP Admin Web Portal API. This arrangement reduces the potential for certain types of web site hacking attempts and attacks known as "SQL injection" attacks. This arrangement also serves to encourage separation of visual interface components from processing components, and allow for a more robust and manageable model/view/controller (MVC) design for the portal code.

Logged in as: **\_battelle** (System Administrator) [Log Out](#)

# IVP System Users

### Main Menu

- + [Home](#)
- + [Change Password](#)
- + [View System Status](#)
- + [View Event Log](#)
- + [View Message Activity](#)

### System Admin Menu

- + [Manage Users](#)
- + [Installed Plugins](#)

## User List

User Name	Access Level	Operations
-----------	--------------	------------

## Create New User

User Name:

Access Level:

- View Only
- Application Administrator
- System Administrator

Copyright (c) 2014 [Battelle Memorial Institute](#). All rights reserved.

**Figure 4-13. The IVP Administration Portal User Administration Page**

## J2735:2015 Message Library

The payloads of messages destined for or originating from the DSRC radio are almost entirely encoded in the ASN.1 formats defined by the J2735:2015 standard. In order for plugin modules to conveniently access the data in the message payloads encoded this way, the IVP platform includes a library that performs the necessary translations to and from the standard format.

The library's functions are accessed by plugin modules through calls to the IVP platform API.

## Networking

The IVP Connection Hub supports both IPv4 and IPv6, but all external components do not. The DSRC Message Manager supports both IPv4 and IPv6 communications to the RSU. The Signal Controller only supports IPv4.

## Security

The IVP Connection Hub will be accessible on a Virtual Private Network (VPN) using secure shell (SSH) network protocol for secure data communications. This connection will allow Battelle to make changes to the IVP Connection Hub software without the need of physical access to the IVP Connection Hub.

This interface allows administrator's access to the IVP Communication Hub without physical access. The interface is secure, uses standard off the shelf components, and allows the admins to do more than through the Admin Web Interface.

The SSH Interface is a COTS product that allows secure access to the IVP Platform's underlying OS. This capability is only accessed by users with appropriate credentials.

Network security to the IVP Connection Hub is handled by the installing agency. For our testing and demonstration purposes, a VPN server with certificate access was used to restrict access to the IVP Connection Hub.

Over-the-air security is provided using certificates and signing of the DSRC messages. The DSRC Message Manager Plugin has a Boolean configuration parameter "Signature" that is passed to the radio to specify whether messages are signed and encrypted.

## The IVP Plugin API

The Plugin API is the foundation of all plugins in the IVP Communication Hub. The API has the functionality to communicate to the IVP Message Router, subscribe and publish messages, read configuration information, and encode and decode messages using the message libraries.

# Chapter 5 Plugin Components

Plugin modules are the means of extending the functionality of the IVP platform.

## Generic Plugin Structure

All IVP platform plugin modules have the same general structure and interact with the core of the IVP platform via standard means. In order to successfully be installed and execute on the IVP platform, a plugin module is required to have:

- A unique name, taken from the name of the deployment zip file for the plugin
- A manifest file, containing the configuration parameters for the plugin and their default values
- An executable file
- A zip file for deployment, containing the manifest file and the executable file for the plugin itself

Plugins on the IVP platform are deployed in a standard location in the platform's directory structure. The directory structure where the plugin is deployed is generated automatically by the system during installation. The name of the plugin deployment directory is taken from the name of the plugin's zip file.

When a plugin is removed (deleted) from the system, the deployment directory and all data associated to the plugin is removed from the file system as well. Log data specific to the plugin is removed, however the record of messages published to the IVP message router is retained in the IVP message router log files.

The manifest file is a JavaScript Object

Notation (JSON) text file that defines the characteristics of the plugin that are known to the IVP platform, configuration parameters that are required by the IVP platform, and any configuration parameters specific to the particular plugin. The JSON representation for an "empty" plugin module is shown in Figure 5-1. The fields defined in the manifest file are:

- Name – The name of the plugin. This string is required to be unique within the plugins installed in the IVP platform, unless the user intends to replace (update) an existing plugin module. The name is limited to 50 characters, and may not be empty.

```
{
  "name": "EmptyPlugin",
  "description": "...",
  "version": "0.0.1",
  "exeLocation": "/Debug/EmptyPlugin",
  "coreIpAddr": "127.0.0.1",
  "corePort": 24601,
  "messageTypes": [
    {
      "type": "...",
      "subtype": "...",
      "description": "..."
    }
  ],
  "configuration": [
    {
      "key": "...",
      "default": "...",
      "description": "..."
    }
  ]
}
```

**Figure 5-1. An "Empty" manifest file**

- **Description** – A brief description of the plugin module. This value is represented in the IVP database as a 'TEXT' data type which may be as large as 65,535 characters in length. The description may not be empty.
- **Version** – The plugin's version number. No validity checking is performed on the version number.
- **Executable Location** – The location in the file system where the executable for the plugin can be found.
- **Core IP Address** – The IPV 4 address of the IVP core.
- **Core Port** – The port number for the IVP core.
- **Message Types** – The type definitions of the messages that this extension plugin will publish to the IVP platform.
- **Configuration parameters** – The parameters required to configure the plugin and their default values.

When a plugin is successfully installed on the IVP platform, the manifest file is read. The configuration parameters and their required default values are recorded in the IVP platform's Configuration and Status database. The values stored in the database can be edited through the Admin Web Portal by a user with appropriate access rights. However, default values for a plugin's configuration parameters cannot be changed through the Admin Web Portal. These may only be changed by edits to the manifest file itself, preferably as updates to the zip file contents prior to being uploaded into the IVP platform. Edits made to a manifest file once the plugin has been installed are not recognized by the IVP platform until the plugin is halted and re-enabled.

## Generic GPS Plugin

The Generic GPS plugin interacts with a connected GPS receiver to supply the IVP Communications Hub with real time positioning information in NEMA GPGGA format. The plugin does not provide any locally generated GPS corrections.

The GPS plugin receives location information from the external GPS receiver through a serial port on the IVP platform, if one is attached, or reads static location information from a local configuration file. The GPS plugin publishes location information as an NEMA GPGGA formatted payload to the IVP Message Router. The manifest file for the Location Plugin is shown in Figure 5-2 and Figure 5-3.

```

"name":"Generic Gps",
  "description":"Exposes GPS data from a device's
    NMEA stream. Also has the ability to spoof
    a NMEA stream using configurable lat/long values",
  "version":"0.0.4",
  "exeLocation":"/Debug/GenericGpsPlugin",
  "coreIpAddr":"127.0.0.1",
  "corePort":24601,
  "messageTypes":[
    {
      "type":"NMEA",
      "subtype":"GGA",
      "description":"Global Positioning System
        Fix Data. Time, position and fix related
        data for a GPS receiver."
    },
    {
      "type":"NMEA",
      "subtype":"GGL",
      "description":"Geographic position,
        latitude / longitude"
    },
    {
      "type":"NMEA",
      "subtype":"GSA",
      "description":"GPS DOP and active satellites"
    },
    {
      "type":"NMEA",
      "subtype":"GSV",
      "description":"GPS Satellites in view"
    },
    {
      "type":"NMEA",
      "subtype":"RMC",
      "description":"Recommended minimum specific
        GPS/Transit data"
    }
  ],

```

**Figure 5-2. The Manifest File for the Generic GPS Plugin**

```

"configuration":[
  {
    "key":"Device Stream File",
    "default":"/dev/ttyACM0",
    "description":"The device stream file
                  of the GPS device."
  },
  {
    "key":"Mode",
    "default":"SPOOFED",
    "description":"Options are: 'SPOOFED' or 'LIVE'.
                  The field is NOT case sensitive."
  },
  {
    "key":"Spoofed Latitude",
    "default":"0.0000",
    "description":"+/- decimal degrees"
  },
  {
    "key":"Spoofed Longitude",
    "default":"0.0000",
    "description":"+/- decimal degrees"
  },
  {
    "key":"Spoofed Elevation",
    "default":"0.0000",
    "description":"+/- meters"
  },
  {
    "key":"Spoofed Speed",
    "default":"0.0000",
    "description":"m/s"
  },
  {
    "key":"Spoofed Course",
    "default":"0.0000",
    "description":"degrees"
  }
]

```

**Figure 5-3. The Manifest File for the Generic GPS Plugin (cont.)**

The fields defined in the Generic GPS Plugin file show the five (5) message subtypes published by the plugin. All of the messages published by the GPS Plugin conform to NMEA 0183 Standard definitions. They are:

- NMEA GAA strings – Global Positioning System Fix Data.
- NMEA GLL strings – Geographic Position, Latitude and Longitude
- NMEA GSA strings – GPS DOP and Active Satellites
- NMEA GSV strings – GPS Satellites in View
- NMEA RCM strings – Recommended Minimum Specific GPS/Transit data

The configuration parameters required by the Generic GPS plugin are:

- The Device Stream File – This parameter defines the serial port from which the plugin can read the GPS data.

- Mode – Determines whether the plugin is to read actual data from a GPS receiver over the serial port, or use a “spoofed” location defined by other parameters
- Spoofed Latitude, Longitude, Elevation, Speed and Course – These five parameters define the data required to construct the NMEA strings necessary for a predefined location.

Figure 5-4 shows the presentation of the status values and configuration parameter values for the Generic GPS Plugin after it has been installed and enabled. This screen is visible to any logged in user with application or system administration access.



Logged in as: **\_battelle** (System Administrator) [Log Out](#)

## Generic Gps Status

**Main Menu**

- + [Home](#)
- + [Change Password](#)
- + [View System Status](#)
- + [View Event Log](#)
- + [View Message Activity](#)

**System Admin Menu**

- + [Manage Users](#)
- + [Installed Plugins](#)

**Status Values**

Key	Value
Device Stream State	Stream Closed
Latitude	0.0000000
Longitude	0.0000000
Course (true)	0
Speed (m/s)	0.00
Signal	Fix
Fix	3D

**Configuration Parameters**

Key	Value	Default Value	Description
Device Stream File	/dev/ttyACM0	/dev/ttyACM0	The device stream file of the gps device.
Mode	SPOOFED	SPOOFED	Options are: 'SPOOFED' or 'LIVE'. The field is NOT case sensitive.
Spoofed Course	0.0000	0.0000	degrees
Spoofed Elevation	0.0000	0.0000	+ - meters
Spoofed Latitude	0.0000	0.0000	+ - decimal degrees
Spoofed Longitude	0.0000	0.0000	+ - decimal degrees
Spoofed Speed	0.0000	0.0000	m/s

Copyright (c) 2014 [Battelle Memorial Institute](#). All rights reserved.

Figure 5-4. The Generic GPS Status Page

## Position Correction Plugin

The Position Correction Provider supplies the IVP Communication Hub with real time differential GPS correction data, using an Ntrip (Networked Transport of RTCM via Internet Protocol) client to receive differential GPS data over the internet in RTCM format.

The Position Correction Plugin receives correction information over an internet connection after supplying the Position Correction Provider its current location information. The position correction information is converted into a J2735 RTCM message and published to the IVP Message Router. The Position Correction Plugin may also receive J2735 RTCM Messages from the IVP Message Router, and if the source of those messages is not itself, the Position Correction Plugin will stop asking for position correction data from the Position Correction Provider until the RTCM messages stop. This strategy allows the IVP platform to act either as a publisher or subscriber of position correction data as dictated by the local environment.

As shown in the manifest file of Figure 5-5, The Position Correction Plugin publishes RTCM 2.3 messages.

The configuration parameters required by the position correction plugin are:

- Endpoint IP Address – This is the IVP-4 address of the source of the RTCM stream
- Endpoint Port – The port address corresponding to the RTCM source.
- Username – The subscriber account username for the RTCM service
- Password – The unencrypted password of the subscriber account
- Mount Point – This identifies the desired type of correction data provided by the position correction provider.

```
{
  "name": "NTRIP",
  "description": "",
  "version": "0.0.1",
  "exeLocation": "/Debug/NtripPlugin",
  "coreIpAddr": "127.0.0.1",
  "corePort": 24601,
  "messageTypes": [
    {
      "type": "RTCM",
      "subtype": "2.3"
    }
  ],
  "configuration": [
    {
      "key": "Endpoint IP",
      "default": "156.63.133.118",
      "description": ""
    },
    {
      "key": "Endpoint Port",
      "default": "2101",
      "description": ""
    },
    {
      "key": "Username",
      "default": "battelle2",
      "description": ""
    },
    {
      "key": "Password",
      "default": "rtkPass",
      "description": ""
    },
    {
      "key": "Mountpoint",
      "default": "ODOT_RTCM23",
      "description": ""
    }
  ]
}
```

**Figure 5-5. The Manifest File for the Position Correction Plugin**

## UTC Plugin

The UTC plugin is the source of Universal Coordinated Time for plugin modules installed on the IVP platform. Depending on the configuration of the UTC plugin, the time source for the system's local time is synchronized with either timestamps from an installed GPS receiver, or from a Network Time Protocol (NTP) server, in the case that no GPS receiver is available and the IVP platform has a reliable source of internet connectivity. The manifest file for the UTC plugin is shown in Figure 5-6.

```
{
  "name": "UTC Time",
  "description": "Provides the current UTC time.",
  "version": "0.0.1",
  "exeLocation": "Debug/UtcPlugin",
  "coreIpAddr": "127.0.0.1",
  "corePort": 24601,
  "messageTypes": [
    {
      "type": "Time",
      "subtype": "MS_UTC",
      "description": "Current UTC time since Epoch in milliseconds."
    },
    {
      "type": "Time",
      "subtype": "Formatted_UTC",
      "description": "Current UTC time formatted into month, day, year, hour, minute, second."
    }
  ],
  "configuration": [
    {
      "key": "Time Server",
      "default": "localhost",
      "description": "The time server or localhost to get time from the IVP host."
    },
    {
      "key": "Frequency",
      "default": "1000",
      "description": "Millisecond frequency to send the time."
    }
  ]
}
```

**Figure 5-6. The Manifest File for the UTC Plugin**

Regardless of the source of the system's local time, the UTC plugin extracts the local server time and formats it into several messages that are published to the IVP Message Router.

## MAP Plugin

The MAP Plugin publishes a J2735 MAP Message containing data to describe complex intersections, curve outlines, and roadway segments which are used by a variety of other J2735 messages. An input to the MAP Plugin is an XML file containing the intersection geometry. Procedures for provisioning of an XML file are contained in the IVP User Document. The manifest file for the MAP plugin is shown in Figure 5-7.

```
{
  "name": "MAP",
  "description": "Provides the intersection geometry.",
  "version": "0.0.1",
  "exeLocation": "/Debug/MapPlugin",
  "coreIpAddr": "127.0.0.1",
  "corePort": 24601,
  "messageTypes": [
    {
      "type": "J2735",
      "subtype": "MAP",
      "description": "Intersection geometry in J2735 format."
    }
  ],
  "configuration": [
  ]
}
```

**Figure 5-7. The Manifest File for the MAP Plugin**

## SPAT Generator Plugin

J2735 SPAT message is published from the SPAT Generator Plugin using the information obtained by the Traffic Signal Controller. The manifest file for the SPAT Generator plugin is shown in Figure 5-8.

```

{
  "name": "SPAT Generator",
  "description": "",
  "version": "0.0.1",
  "exeLocation": "/Debug/SpatGeneratorPlugin",
  "coreIpAddr": "127.0.0.1",
  "corePort": 24601,
  "messageTypes": [
    {
      "type": "J2735",
      "subtype": "SPAT"
    }
  ],
  "configuration": []
}

```

**Figure 5-8. The Manifest File for the SPAT Generator Plugin**

## DSRC Message Manager Plugin

The DSRC Message Manager Plugin is responsible for taking internal messages flagged for transmission and ensuring they are sent out via the DSRC radio. Communications to the RSU will be UDP messages as defined in the RSU 3.1 specification. A sample message to immediately send a MAP message is shown below in Figure 5-9.

```

Version=0.5
Type=MAP
PSID=0xBFF0
Priority=7
TxMode=CONT
TxChannel=172
TxInterval=0
DeliveryStart=
DeliveryStop=
Signature=True
Encryption=False
Payload=3081DE800110810900000000000000001000830101A481C63081C3800102A11BA119A01080
0418054A3B8104CE3585DF82020D0681020040820102820207DB830306162184027D00850102A61
080041804FD888104CE35C39E82020CF68702016E880100A93C303A80020040A234A032A3300404
1C6BCDB304040420EC2B0404FAC8EC280404EF79F1210404EBC4FD660404E65310690404F9621
AA50404095B3F31AA3AA0383006A004800235293006A0048002010C3006A004800231383006A00
4800222113006A0048002010C3006A004800231483006A0048002221185021001

```

Source: Locomate USersGuid\_V1.26.pdf, section D2

**Figure 5-9. Sample UDP DSRC Message**

## INFLO Plugin

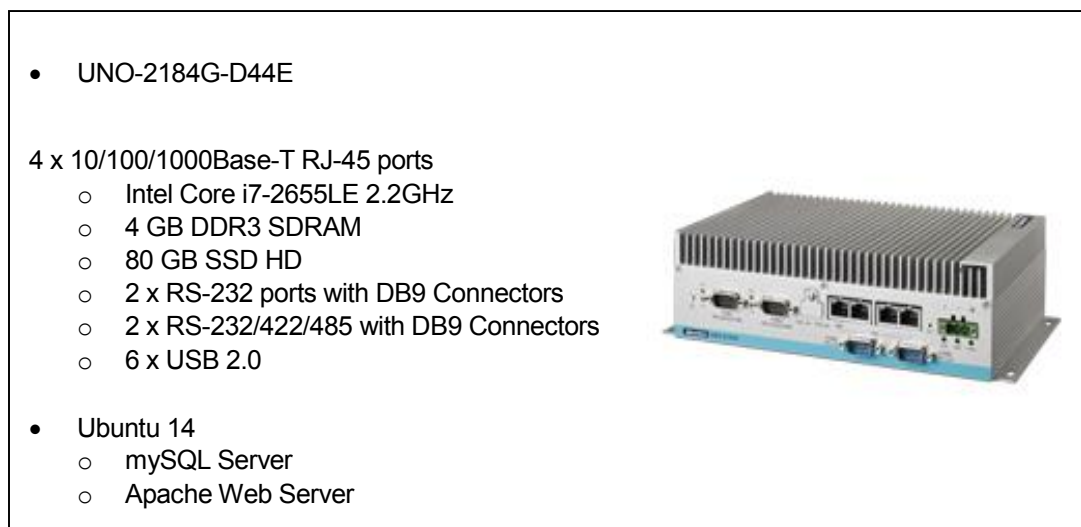
The INFLO Plugin retrieves Queue Warnings and Speed Harm TIM messages from the cloud computing system and sends the TIM messages out the DSRC radio. It also supplies BSM messages to the cloud computing system for processing. The manifest file for the INFO Plugin is shown in Figure 5-10.

```
{
  "name": "INFLO",
  "description": "The INFLO Plugin retrieves Queue Warnings and Speed Harm TIM messages from the cloud computing system and sends the TIM messages out the DSRC radio. The INFLO Plugin also supplies BSM messages to the cloud computing system for processing.",
  "version": "1.0.0",
  "exeLocation": "/Debug/InfloPlugin",
  "coreIpAddr": "127.0.0.1",
  "corePort": 24601,
  "messageTypes": [
  ],
  "configuration": [
    {
      "key": "BSM Web API",
      "default": "http://inflatst-cloud-roles.cloudapp.net/api/bsm",
      "description": "URL of the BSM Web API"
    },
    {
      "key": "TIM Web API",
      "default": "http://inflatst-cloud-roles.cloudapp.net/api/tim",
      "description": "URL of the TIM Web API"
    },
    {
      "key": "Roadway ID",
      "default": "0",
      "description": "ID of the INFLO Roadway"
    },
    {
      "key": "Roadway MM",
      "default": "0.0",
      "description": "Milemarker location on INFLO Roadway"
    }
  ]
}
```

**Figure 5-10. The Manifest File for the INFLO Plugin**

# Chapter 6 Hardware

The hardware selected to support the initial IVP platform development and deployment was the Advantech UNO-2184G. This unit is an example of an industrial grade, commercial off the shelf (COTS) automation computer, providing extensive I/O functionality as well as ample processing power to handle the needs of the targeted V2I applications as well as the message routing and administrative needs of the core IVP platform functionality. Key specifications of the hardware and software are listed in Figure 6-1.



Source: Advantech

**Figure 6-1. Platform Hardware and Specifications**

The DSRC (Dedicated short-range communications) radio will be used to relay messages from the infrastructure (RSU) to connected vehicles (OBU). The specific hardware selected for this function will vary with deployment. For our setup we are using Arada radios for both the RSU and OBU. The Arada RSU will be an Arada System's LocoMate™ Commando. The DSRC radio used as a proxy for deployed hardware during the development of the IVP platform is an Arada System's LocoMate™ GO OBU battery powered unit, shown in Figure 6-2. The Arada Systems' LocoMate™ GO OBU unit will receive messaging from the DSRC radio. Refer to Arada Systems' documentation for recommended antenna configurations.

The solution integrates GPS, Bluetooth and high-power 802.11p radios. It is fully compliant with Omni-Air's certification and used in worldwide deployments including the U.S. DOT's Safety Pilot in Ann Arbor, Michigan.

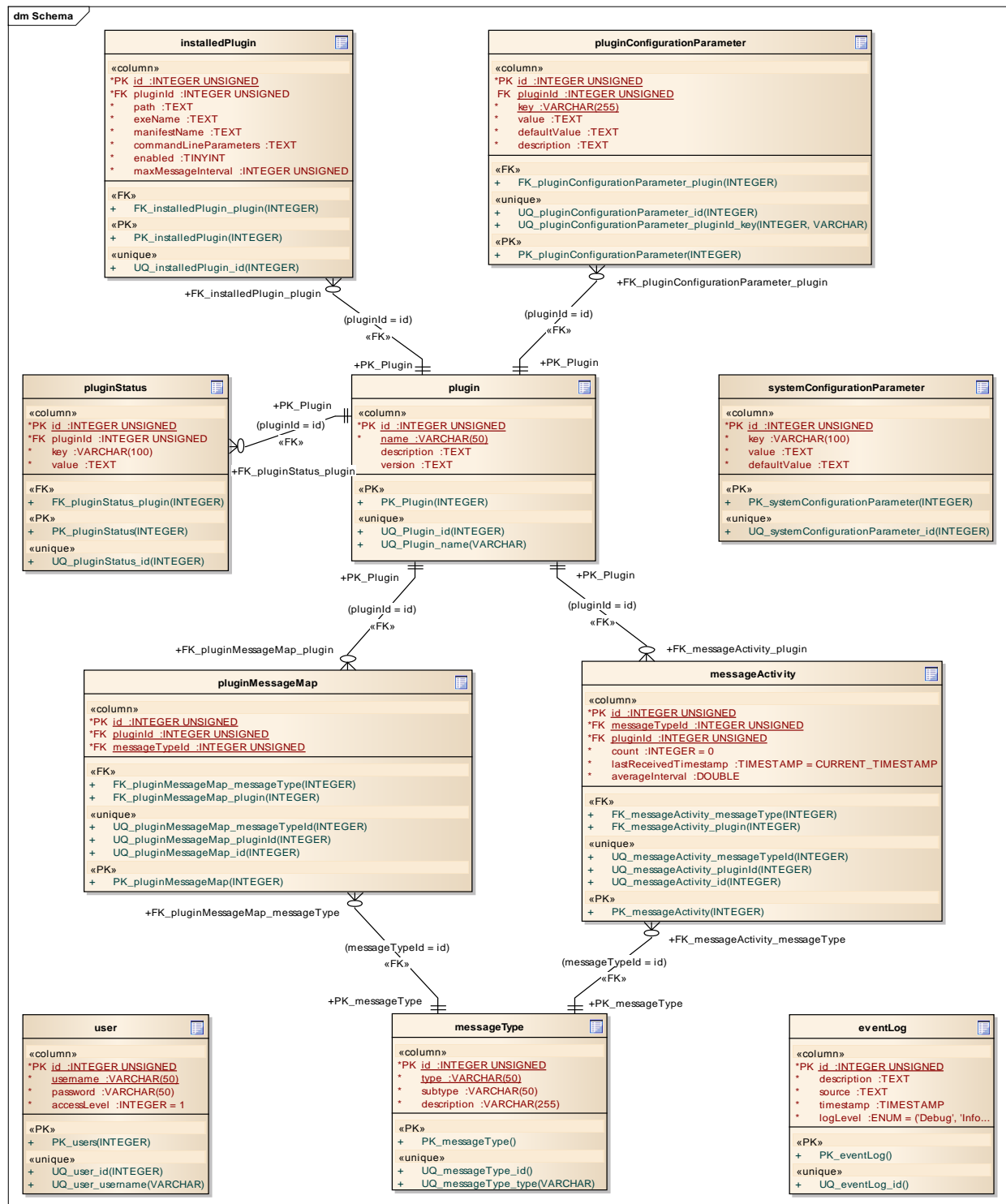


Source: Arada Systems

**Figure 6-2. The Arada Systems' LocoMate™ GO OBU**



# Chapter 7 Database Schema



### Figure 7-1. Database Schema

## installedPlugin

**Table 7-1. installedPlugin Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True		Primary key
False	pluginId	INTEGER	True	False		Foreign key into plugin table.
False	path	TEXT	True	False		The location of the plugin executable in the file system.
False	exeName	TEXT	True	False		The name of the plugin's executable.
False	manifestName	TEXT	True	False		The name of the manifest file describing this plugin's characteristics
False	commandLineParameters	TEXT	True	False		
False	enabled	TINYINT	True	False		
False	maxMessageInterval	INTEGER	True	False		

**Table 7-2. installedPlugin Table Column Constraints**

Name	Type	Columns
FK_installedPlugin_plugin	Public	pluginId
PK_installedPlugin	Public	id
UQ_installedPlugin_id	Public	id

**Table 7-3. installedPlugin Table Relationships**

Columns	Association	
(pluginId = id)	0..*	installedPlugin.FK_installedPlugin_plugin
	1	plugin.PK_Plugin

## pluginStatus

**Table 7-4. pluginStatus Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True		Primary key
False	pluginId	INTEGER	True	False		Foreign key into plugin table
False	key	VARCHAR	True	False	100	
False	value	TEXT	True	False		

**Table 7-5. pluginStatus Table Column Constraints**

Name	Type	Columns
FK_pluginStatus_plugin	Public	pluginId
PK_pluginStatus	Public	id
UQ_pluginStatus_id	Public	id

**Table 7-6. pluginStatus Table Relationships**

Columns	Association
(pluginId = id)	0..* pluginStatus.FK_pluginStatus_plugin 1 plugin.PK_Plugin

## systemConfigurationParameter

This table lists the IVP system configuration parameters used by both core components and plugins to control the behavior of the system.

**Table 7-7. systemConfigurationParameter Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True		Primary key
False	key	VARCHAR	True	False	100	The name of a configuration parameter.
False	value	TEXT	True	False		The value of a configuration parameter
False	defaultValue	TEXT	True	False		

**Table 7-8. systemConfigurationParameter Table Column Constraints**

Name	Type	Columns
PK_systemConfigurationParameter	Public	id
UQ_systemConfigurationParameter_id	Public	id

This table does not have any foreign keys, therefore no relationship table exists.

## pluginConfigurationParameter

This table lists the IVP system configuration parameters used by both core components and plugins to control the behavior of the system.

**Table 7-9. pluginConfigurationParameter Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True		Primary key
False	pluginId	INTEGER	False	True		Foreign key into plugin table.
False	key	VARCHAR	True	True	255	The name of a configuration parameter.
False	value	TEXT	True	False		The value of a configuration parameter
False	defaultValue	TEXT	True	False		Default setting of a configuration parameter.
False	description	TEXT	True	False		A brief description of the parameter's purpose.

**Table 7-10. pluginConfigurationParameter Table Column Constraints**

Name	Type	Columns
FK_pluginConfigurationParameter_plugin	Public	pluginId
UQ_pluginConfigurationParameter_id	Public	id
UQ_pluginConfigurationParameter_pluginId_key	Public	pluginId key
PK_pluginConfigurationParameter	Public	id

**Table 7-11. pluginConfigurationParameter Table Relationships**

Columns	Association	
(pluginId = id)	<b>0..*</b>	pluginConfigurationParameter.FK_pluginConfigurationParameter_plugin
	<b>1</b>	plugin.PK_Plugin

## plugin

This table lists the plugins loaded and available to run on the IVP platform.

**Table 7-12. plugin Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True		Primary key
False	name	VARCHAR	True	True	50	A unique plugin name
False	description	TEXT	False	False		
False	version	TEXT	False	False		

**Table 7-13. plugin Table Column Constraints**

Name	Type	Columns
PK_Plugin	Public	id
UQ_Plugin_id	Public	id
UQ_Plugin_name	Public	name

**Table 7-14. plugin Table Relationships**

Columns	Association
(pluginId = id)	0..* pluginMessageMap.FK_pluginMessageMap_plugin 1 plugin.PK_Plugin
(pluginId = id)	0..* messageActivity.FK_messageActivity_plugin 1 plugin.PK_Plugin
(pluginId = id)	0..* pluginConfigurationParameter.FK_pluginConfigurationParameter_plugin 1 plugin.PK_Plugin
(pluginId = id)	0..* installedPlugin.FK_installedPlugin_plugin 1 plugin.PK_Plugin
(pluginId = id)	0..* pluginStatus.FK_pluginStatus_plugin 1 plugin.PK_Plugin

## messageType

This table lists the valid message types of every plugin loaded on the IVP platform.

**Table 7-15. messageType Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True	0	Primary key
False	type	VARCHAR	True	True	50	A unique message type name
False	subtype	VARCHAR	True	False	50	
False	description	VARCHAR	True	False	255	A description of the message type

**Table 7-16. messageType Table Column Constraints**

Name	Type	Columns
PK_messageType	Public	id
UQ_messageType_id	Public	id
UQ_messageType_type	Public	type

**Table 7-17. messageType Table Relationships**

Columns	Association	
(messageTypeId = id)	<b>0..*</b>	messageActivity.FK_messageActivity_messageType
	<b>1</b>	messageType.PK_messageType
(messageTypeId = id)	<b>0..*</b>	pluginMessageMap.FK_pluginMessageMap_messageType
	<b>1</b>	messageType.PK_messageType

## pluginMessageMap

This table identifies the types of messages generated by each plugin.

**Table 7-18. pluginMessageMap Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True		Primary key
False	pluginId	INTEGER	True	True		Foreign key into the plugin table
False	messageTypeId	INTEGER	True	True		Foreign key into the messageType table.

**Table 7-19. pluginMessageMap Table Column Constraints**

Name	Type	Columns
FK_pluginMessageMap_messageType	Public	messageTypeId
UQ_pluginMessageMap_messageTypeId	Public	messageTypeId
UQ_pluginMessageMap_pluginId	Public	pluginId
PK_pluginMessageMap	Public	id
UQ_pluginMessageMap_id	Public	id
FK_pluginMessageMap_plugin	Public	pluginId

**Table 7-20. pluginMessageMap Table Relationships**



Columns	Association	
(pluginId = id)	<b>0..*</b>	pluginMessageMap.FK_pluginMessageMap_plugin
	<b>1</b>	plugin.PK_Plugin
(messageTypeId = id)	<b>0..*</b>	pluginMessageMap.FK_pluginMessageMap_messageType
	<b>1</b>	messageType.PK_messageType

## eventLog

This table records events generated by every IVP core component and plugin in the IVP platform.

**Table 7-21. eventLog Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True	0	Primary key
False	description	TEXT	True	False		The log message content
False	source	TEXT	True	False		The name of the plugin or other agent that logged the event
False	timestamp	TIMESTAMP	True	False		The date and time of the event in UTC
False	logLevel	ENUM	True	False		The type of event being logged, one of <ul style="list-style-type: none"> <li>• Debug</li> <li>• Info</li> <li>• Warning</li> <li>• Error</li> <li>• Fatal</li> </ul>

**Table 7-22. eventLog Table Column Constraints**

Name	Type	Columns
PK_eventLog	Public	id
UQ_eventLog_id	Public	id

## messageActivity

This table records the most recent message activity of each active plugin in the IVP system. The data in this table is updated by the IVP plugin monitor core component for every message the plugin monitor receives.

**Table 7-23. messageActivity Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True		Primary key
False	messageTypeId	INTEGER	True	True		Foreign key into the messageType table
False	pluginId	INTEGER	True	True		Foreign key into the plugin table
False	count	INTEGER	True	False		Records the total number of messages of a single message type emitted by a plugin.
False	lastReceivedTimestamp	TIMESTAMP	True	False		The date and time of the most recent message of a type in UTC.
False	averageInterval	DOUBLE	True	False		The rolling average of the interval between most recent set of messages of a single message type emitted by a plugin, measured in milliseconds.

**Table 7-24. messageActivity Table Column Constraints**

Name	Type	Columns
FK_messageActivity_messageType	Public	messageTypeId
UQ_messageActivity_messageTypeId	Public	messageTypeId
UQ_messageActivity_pluginId	Public	pluginId
PK_messageActivity	Public	id
UQ_messageActivity_id	Public	id
FK_messageActivity_plugin	Public	pluginId

**Table 7-25. messageActivity Table Relationships**

Columns	Association	
(pluginId = id)	<b>0..*</b>	messageActivity.FK_messageActivity_plugin
	<b>1</b>	plugin.PK_Plugin
(messageTypeId = id)	<b>0..*</b>	messageActivity.FK_messageActivity_messageType
	<b>1</b>	messageType.PK_messageType

## user

The list of accounts that can access the IVP platform via the administrative portal is held in the users table.

**Table 7-26. user Relational Database Table Columns**

PK	Name	Type	Not Null	Unique	Len	Notes
True	id	INTEGER	True	True		Primary key
False	username	VARCHAR	True	True	50	The account name for the user, typically an email address
False	password	VARCHAR	True	False	50	An encrypted password
False	accessLevel	INTEGER	True	False		The access level permitted for this user, one of: 1. read-only access to portal 2. application administrator access 3. system administrator, all access

**Table 7-27. user Table Column Constraints**

Name	Type	Columns
PK_users	Public	id
UQ_user_id	Public	id
UQ_user_username	Public	username

This table does not have any foreign keys, therefore no relationship table exists.

U.S. Department of Transportation  
ITS Joint Program Office-HOIT  
1200 New Jersey Avenue, SE  
Washington, DC 20590

Toll-Free "Help Line" 866-367-7487  
[www.its.dot.gov](http://www.its.dot.gov)

FHWA-JPO-14-TBD



U.S. Department of Transportation