AN-028 Enabling Verkada LPR Camera with Radar Speed Reports

License Plate Recognition (LPR) cameras have become very useful for traffic monitoring. They become even more useful when they are combined with a radar sensor such as OPS9243-A to report speeding. The radar can alert the camera of excessive speed to capture the make, color, and license plate of the vehicle. This allows authorities to follow up with individuals with warnings or possibly tickets. San Francisco has started mounting speed reporting cameras and in one month issued nearly 100,000 notices of speeding.

Verkada provides LPR cameras (CB52-E and CB62-E) which are very simple to install and link into their network, while providing an easy to navigate dashboard with capabilities to filtering by vehicle type, color, and time. This application note explains how to set up a WiFi enabled OPS9243-A which delivers speed data into the Verkada Command dashboard.

Camera + Radar System Architecture

The system architecture includes a Verkada camera and WiFi enabled OPS9243-A (Figure *1*) mounted with the same roadway field of view. The Verkada camera is connected via Power-over-Ethernet (PoE) to send its video data to the Verkada Command dashboard. OPS9243-A uses PoE for power but sends the speed data over WiFi to the cloud and a Node Red server. The Node Red server runs simple routines to format the MQTT speed message in a manner that the Command dashboard can receive.

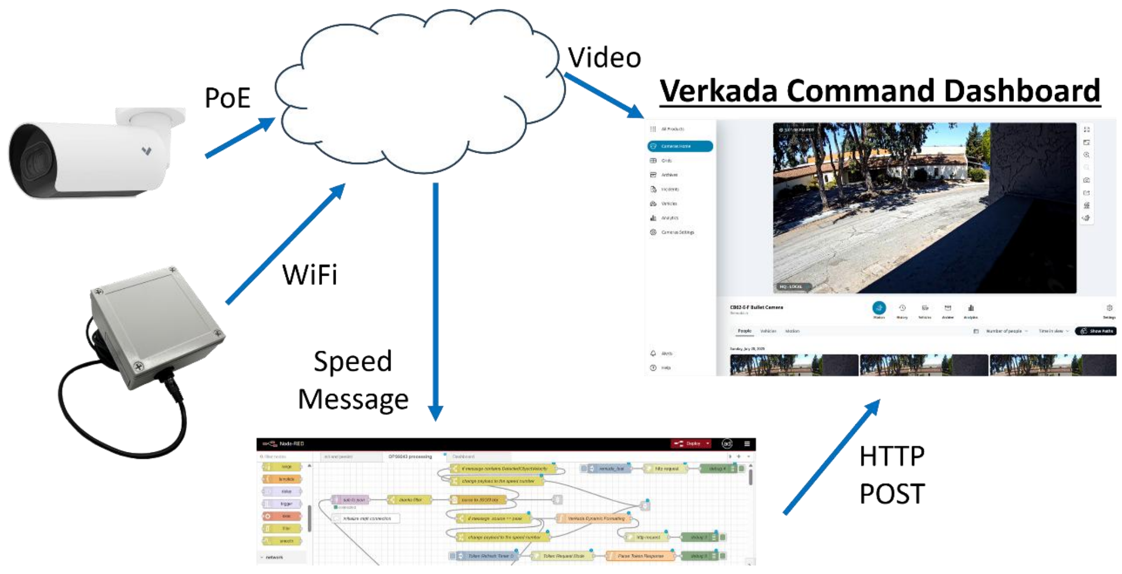


Figure 1. Verkada Camera and OPS9243-A Radar System Architecture

Part of the message includes unique ID's which designate the customer account, the message data being received, a unique token, and the camera ID.

- API Key – unique to the customer account
- UUID – unique to the Helix Event
- API Token – unique to the Command message, 30 minutes life
- Camera ID – unique to a camera

Mounting Camera and OPS9243-A

The camera and OPS9243-A should be mounted in a location where they view the same area of the roadway. OPS9243-A has a much narrower field of view (FoV) at 20° (horizontal) x 24° (vertical) and it should be focused on the center of the roadway monitored. If a two-lane road, use the middle of the two lanes as the target center point of the sensor FoV. On the OmniPreSense website is a FoV calculator which can help determine where the sensor is focused based on the mount height, angle looking into the roadway, and the down angle. The OPS9243-A has a built in down angle of 10°.

Verkada cameras have a much wider FoV at 107° (horizontal) x 62° (vertical). When zoomed in, the FoV is 43° (horizontal) x 24° (vertical). Make sure to match the camera FoV with the radar as much as possible to eliminate detecting vehicles outside of the radar FoV (ex. parking lot to left side in Figure 2).

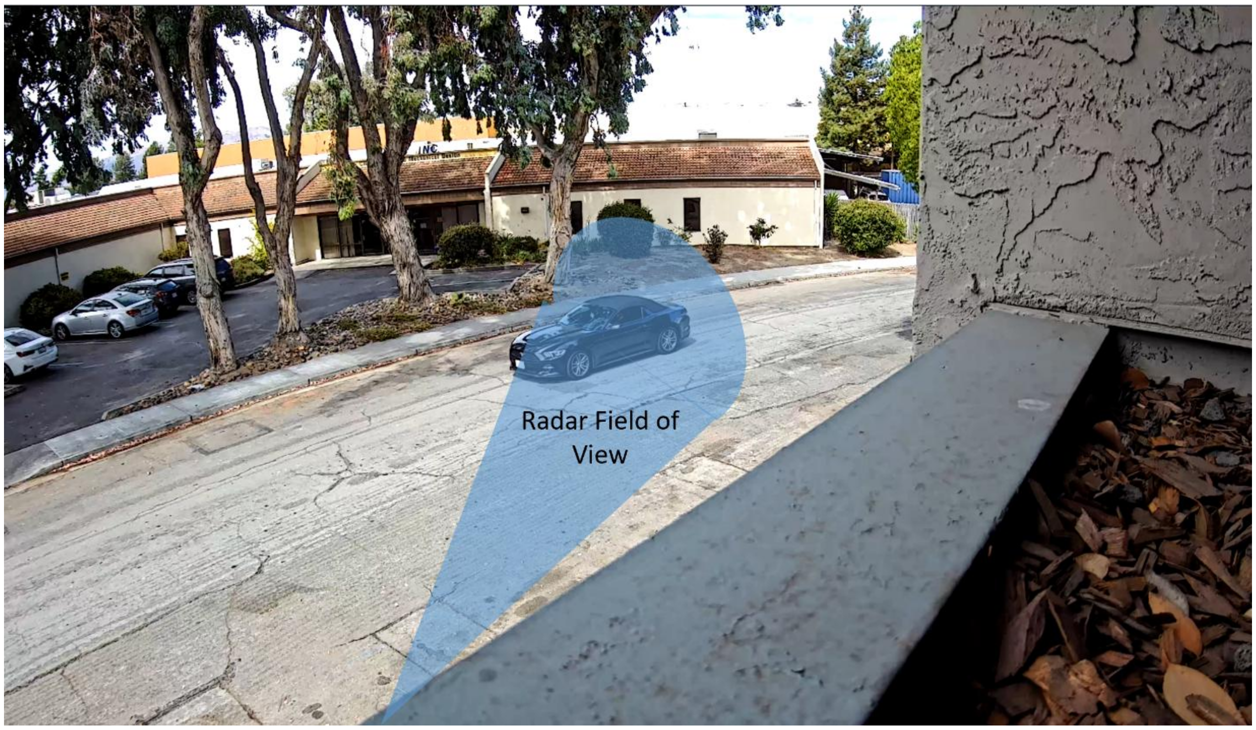


Figure 2. Camera and Radar Sensor Field of View

Speed Message Command Dashboard Configuration

Assuming you already have a Verkada camera installed and configured for the Command dashboard, you can next set up the speed message to send into Command. The process consists of creating a Helix Event and the appropriate IDs. The process is as follows:

1) Generate an API Key
2) Generate a Helix Event – defines the message format and data sent
3) Generate an API Token
4) Format Event message with ID – Node Red will be used and explained in detail below

Step 1: Generate API Key

The API Key is generated on the Command dashboard. Go to the square (nine dots) in the upper left of the dashboard or All Products. Selecting this brings up a pop-up window and select Admin (Figure 3).

Next, select API & Integration (Figure *4*) and the on the following window select API Keys (Figure 5). This brings up the main window for any existing API Keys. Press the blue + Add button (Figure 6) to bring up the Add New API Key configuration window (Figure 7).

In the New API Key configuration, provide a name for your key and select the Endpoints type. For sending speed data into Command, select Helix and make sure its Read/Write. Then select the Expiration time and remember it will eventually need to be renewed. Hitting Generate Key (Figure 8) will provide the API Key. Make sure to save this away in a secure location as it will be used later for the API Token generation.

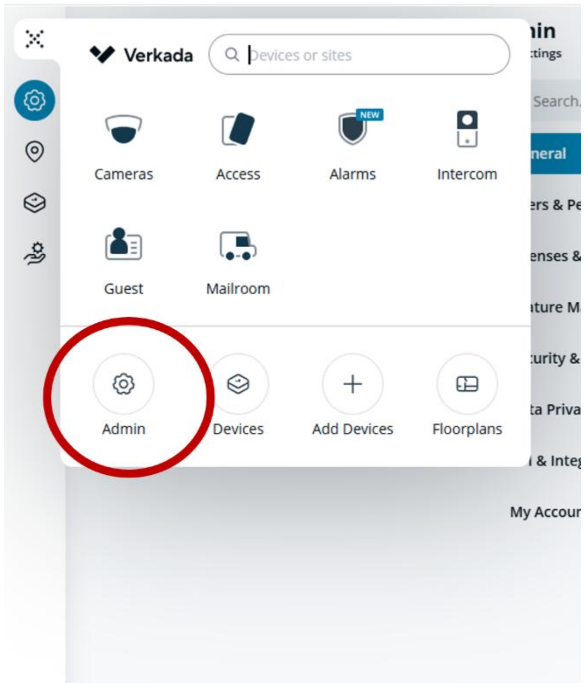
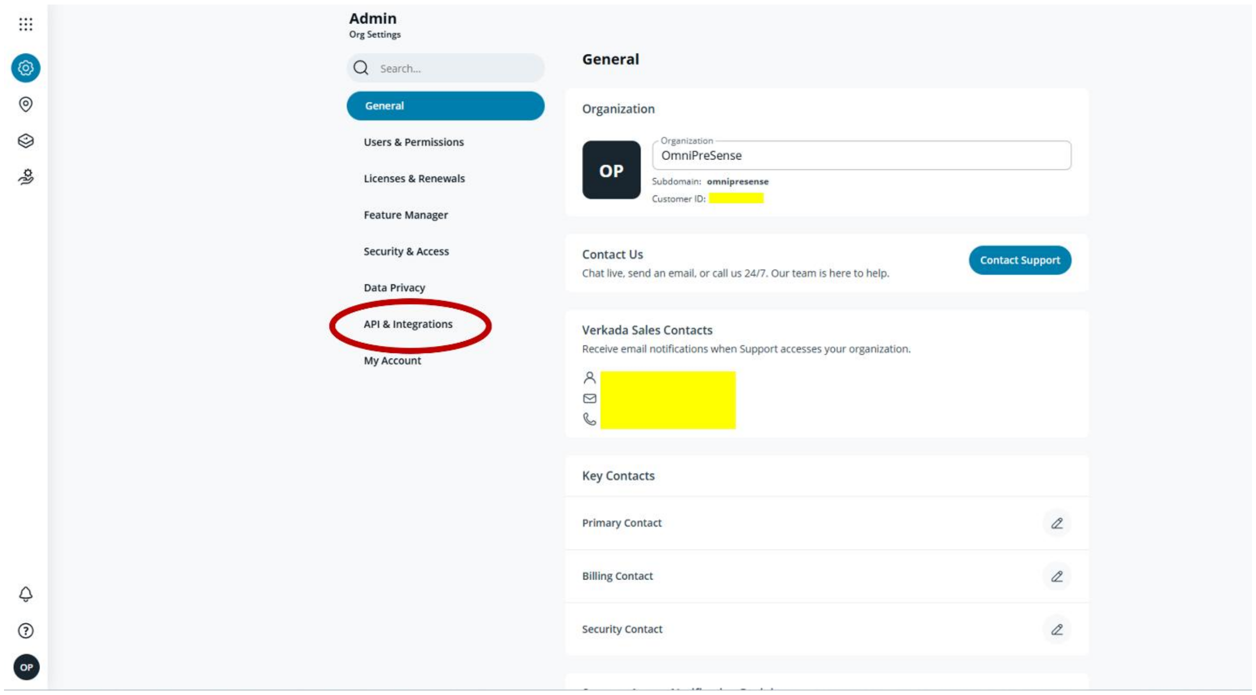


Figure 3. API Key Generation – Select Admin



Admin
Org Settings

Search...

General
Users & Permissions
Licenses & Renewals
Feature Manager
Security & Access
Data Privacy
API & Integrations
My Account

General

Organization

OP

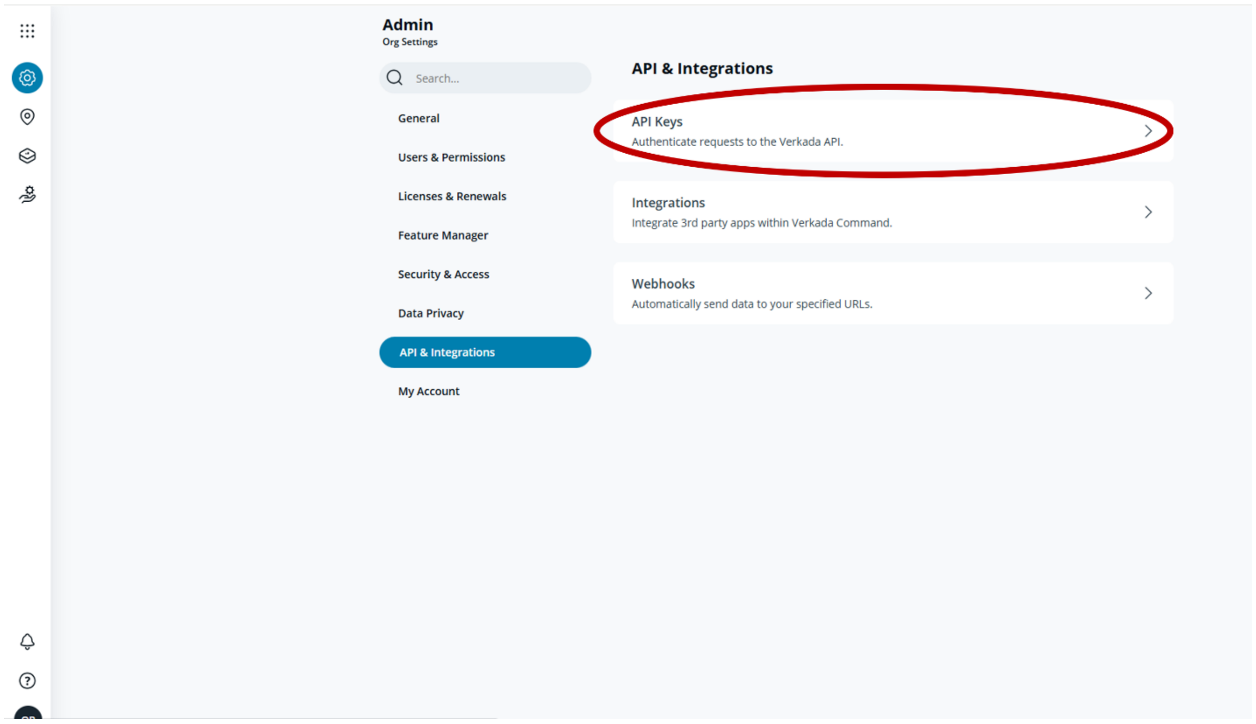
Organization: OmniPreSense
Subdomain: omnipresense
Customer ID: [redacted]

Contact Us
Chat live, send an email, or call us 24/7. Our team is here to help.
Contact Support

Verkada Sales Contacts
Receive email notifications when Support accesses your organization.
[redacted]

Key Contacts
Primary Contact
Billing Contact
Security Contact

Figure 4. API Key Generation – Select API & Integration



Admin
Org Settings

Search...

General
Users & Permissions
Licenses & Renewals
Feature Manager
Security & Access
Data Privacy
API & Integrations
My Account

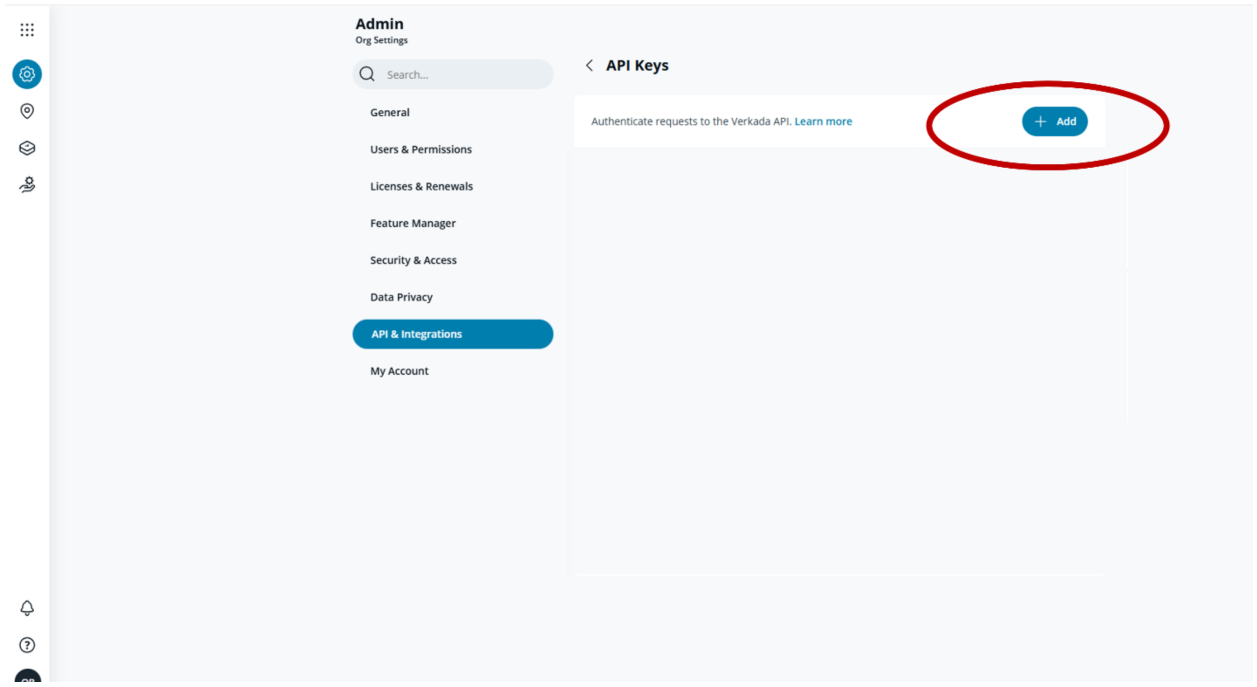
API & Integrations

API Keys
Authenticate requests to the Verkada API.

Integrations
Integrate 3rd party apps within Verkada Command.

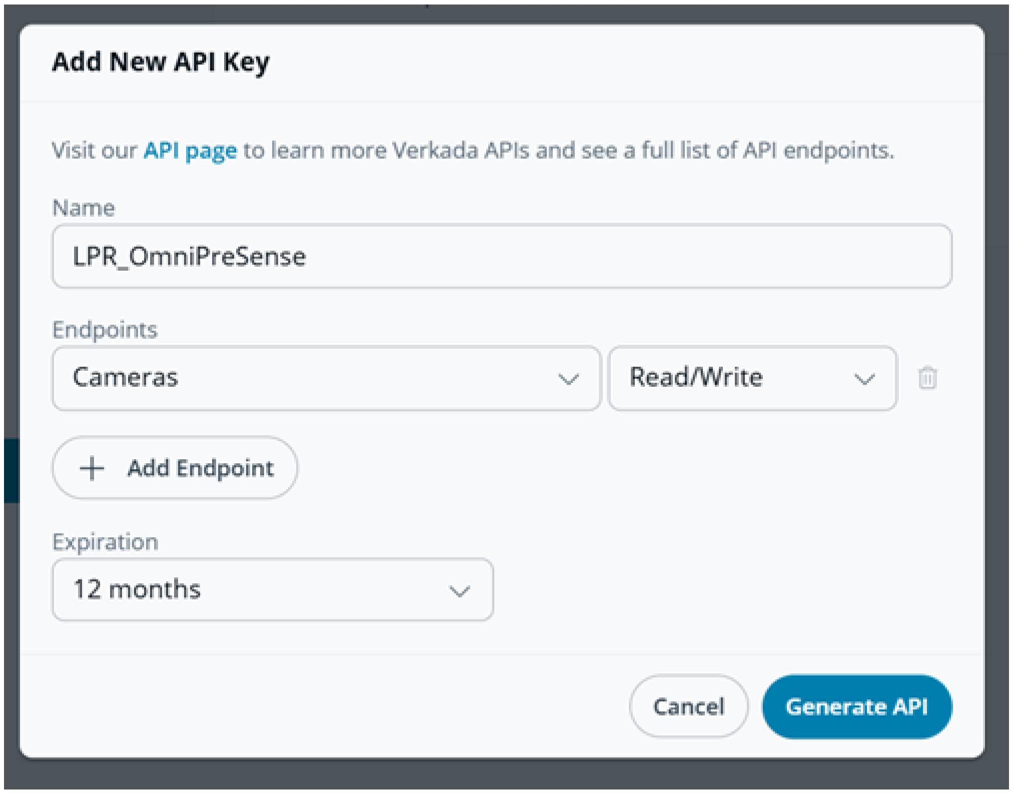
Webhooks
Automatically send data to your specified URLs.

Figure 5. API Key Generation – Select API Keys



Authenticate requests to the Verkada API. Learn more

Figure 6. API Key Generation – Select + Add



Add New API Key

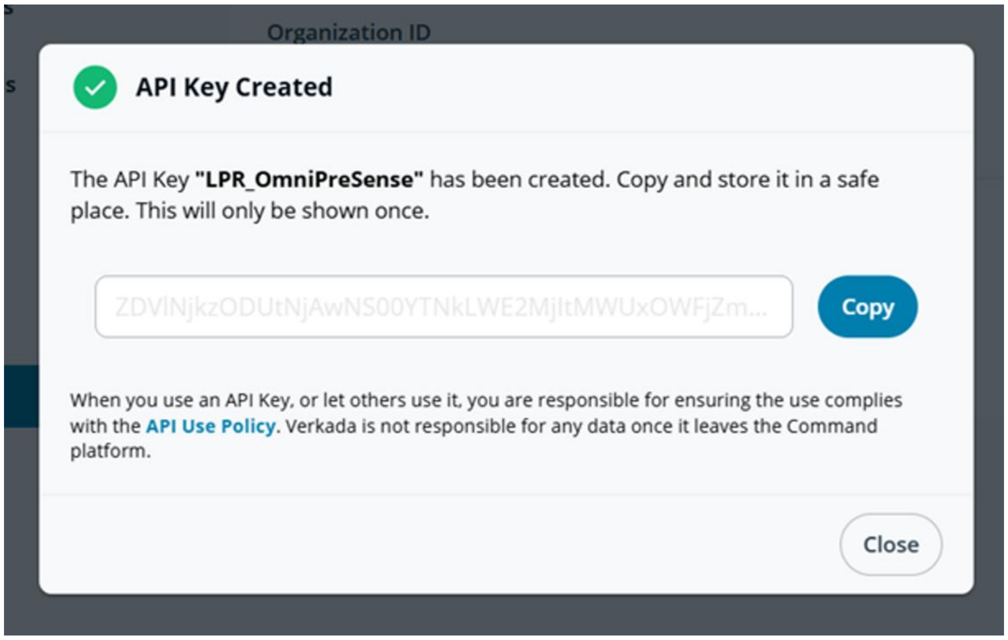
Visit our API page to learn more Verkada APIs and see a full list of API endpoints.

Name: LPR_OmniPreSense

Endpoints: Cameras, Read/Write

Expiration: 12 months

Figure 7. API Key Generation – New API Key Generation


Organization ID

API Key Created

The API Key **"LPR_OmniPreSense"** has been created. Copy and store it in a safe place. This will only be shown once.

ZDVlNjkzODUtNjAwNS00YTNkLWE2MjItMWUxOWFjZm...

When you use an API Key, or let others use it, you are responsible for ensuring the use complies with the **API Use Policy**. Verkada is not responsible for any data once it leaves the Command platform.

Figure 8. API Key Generation – New API Key

Step 2: Generate Helix Event

The Helix Event configures Command with the type of information you will be passing into Command from the OPS9243-A. Upon generating the format of the Event, a UUID is generated which is used in each speed message sent to Command.

To generate a Helix Event, navigate to and select Alerts at the lower left of the Command dashboard followed by the Helix Events + button (Figure 9). Following this, select the Generate Helix Event button (Figure 10).

In the following pop window (Figure 11), provide a name for the Event and an attribute. Speed is the primary data that will be passed to Command so define it as an attribute. Speed is an integer coming from OPS9243-A. Assign speed as an integer. Additional attributes may be defined such as direction, timestamp, and possibly lane and vehicle classification for future use. Select the appropriate format (string, float, etc.) for each attribute. When this is done, press Save.

The resulting window (Figure 12) provides the format of the JSON message that is sent to Command with each speed event reported by OPS9243-A. Save this information to a secure location as there is information in it which is required for Command to recognize the data as valid and accept it. Within the message is the URL that the messages will be sent by HTTP POST and the UUID that identifies the Helix Event.

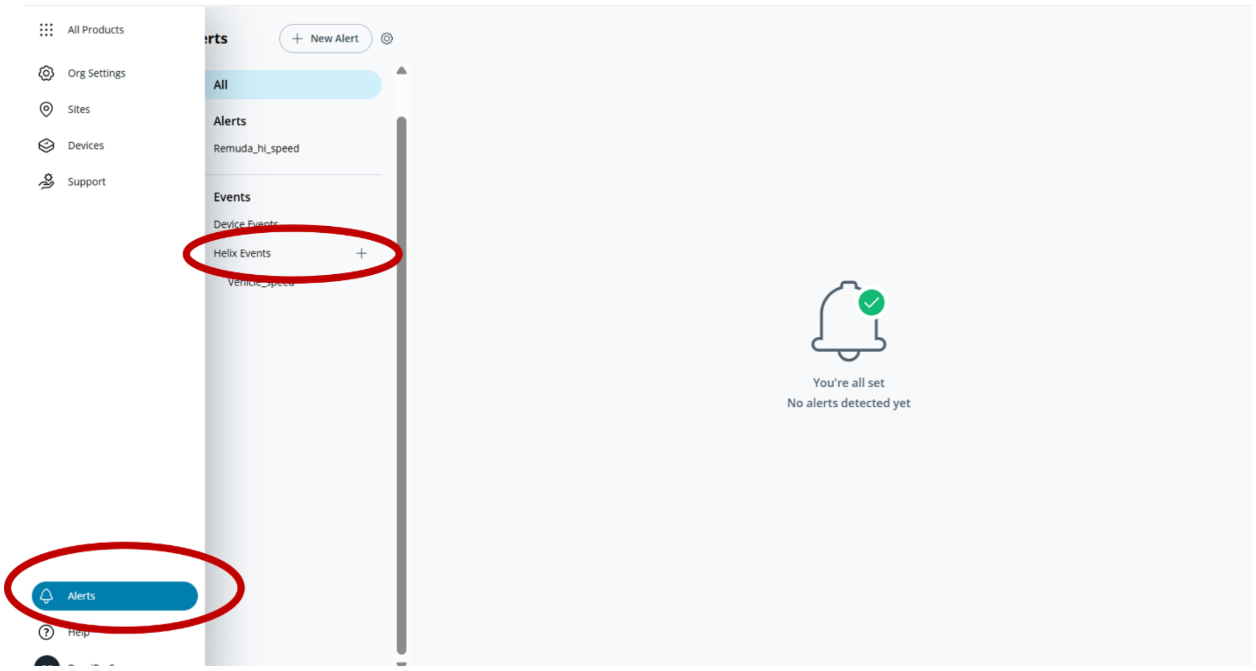
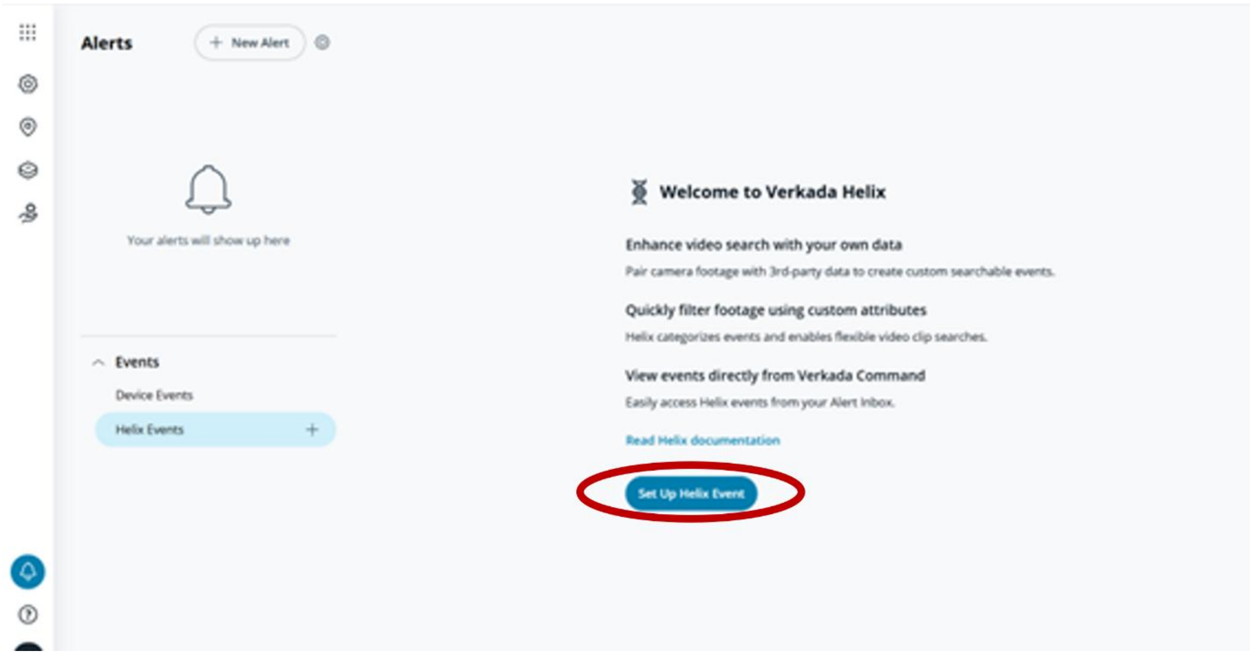


Figure 9. Helix Event Generation – Alerts and Helix Event + Button



Enhance video search with your own data

Pair camera footage with 3rd-party data to create custom searchable events.

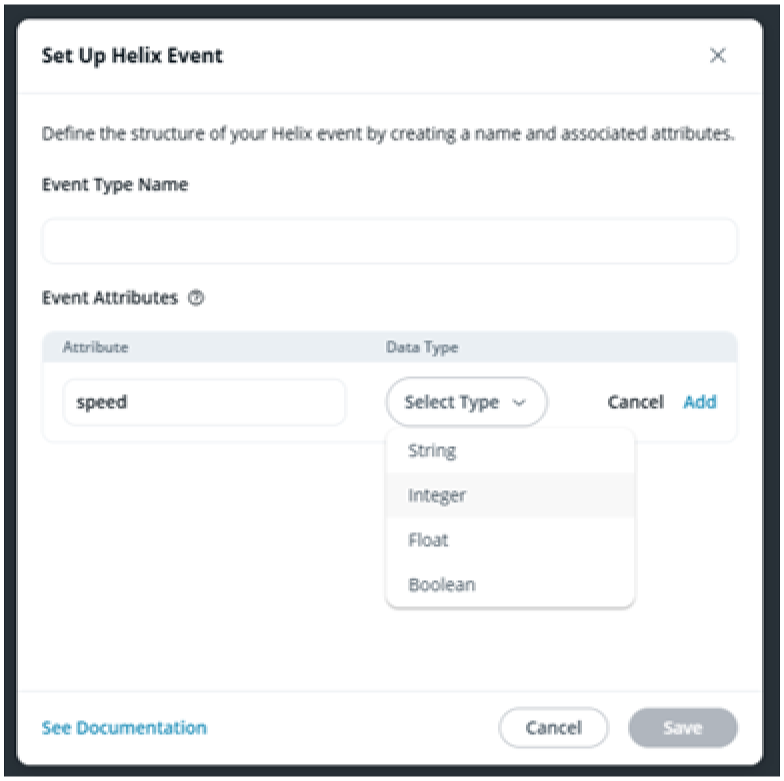
Quickly filter footage using custom attributes

Helix categorizes events and enables flexible video clip searches.

View events directly from Verkada Command

Easily access Helix events from your Alert Inbox.

Figure 10. Helix Event Generation – Set Up Helix Event



Set Up Helix Event

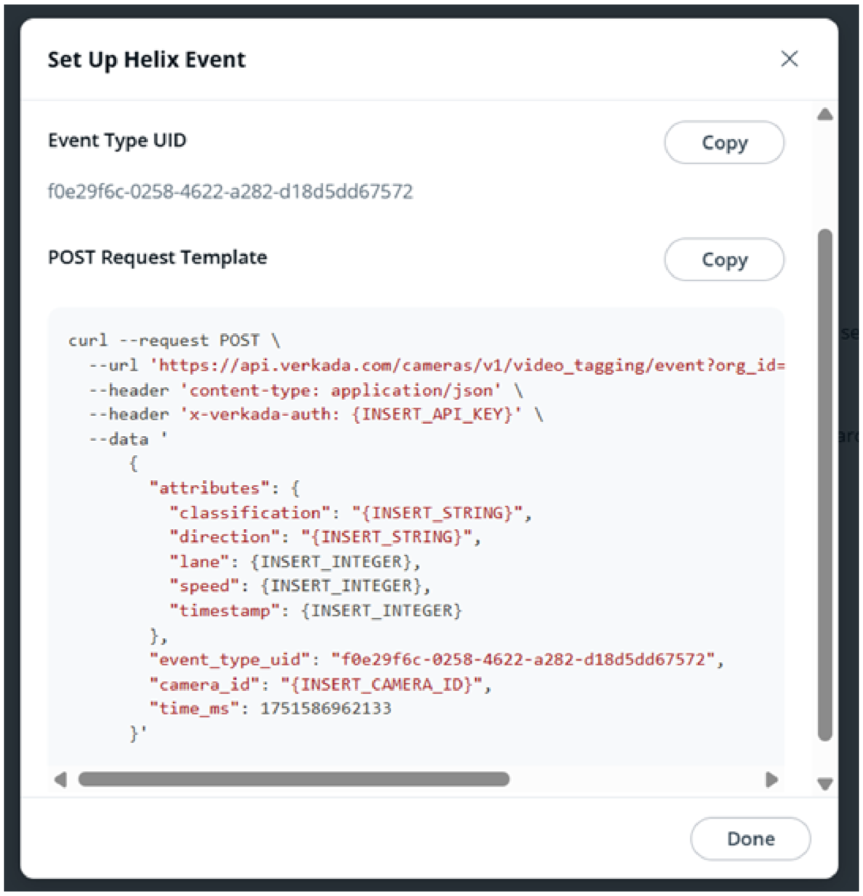
Define the structure of your Helix event by creating a name and associated attributes.

Event Type Name

Event Attributes

Attribute	Data Type
speed	Select Type

Figure 11. Helix Event Generation – Attributes Configuration



f0e29f6c-0258-4622-a282-d18d5dd67572

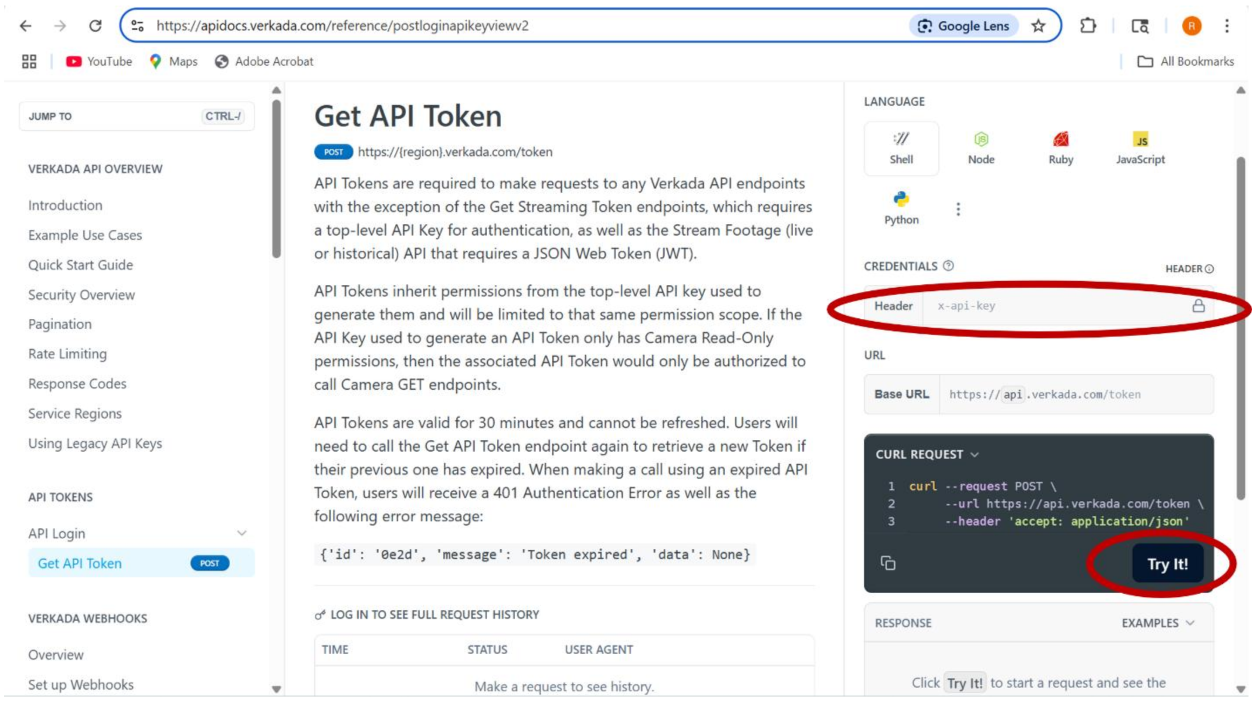
```
curl --request POST \
  --url 'https://api.verkada.com/cameras/v1/video_tagging/event?org_id=
  --header 'content-type: application/json' \
  --header 'x-verkada-auth: {INSERT_API_KEY}' \
  --data '
    {
      "attributes": {
        "classification": "{INSERT_STRING}",
        "direction": "{INSERT_STRING}",
        "lane": {INSERT_INTEGER},
        "speed": {INSERT_INTEGER},
        "timestamp": {INSERT_INTEGER}
      },
      "event_type_uid": "f0e29f6c-0258-4622-a282-d18d5dd67572",
      "camera_id": "{INSERT_CAMERA_ID}",
      "time_ms": 1751586962133
    }'
```

Figure 12. Helix Event Generation – Event JSON Message Format

Step 3: Generate API Token

The JSON message format lists {INSERT_API_KEY}. This is misleading as it is not the API Key that was generated in Step 1 but an API Token that needs to be generated. The API Token can be generated manually on the Verkada website in the Get API Token section of their documentation (Figure 13).

On the Verkada website, insert the API Key previously generated into the box on the right side under Credentials – Header and press the Try It button. The result will provide an API Token with a notification that it is **valid for only 30 minutes**. After this time, Command will no longer accept messages with that API Token. To get around this, a means of regenerating the token is required. In this implementation, regenerating the token is handled in Node Red and described below. You may save this API Token for testing purposes.



Get API Token

POST https://{region}.verkada.com/token

API Tokens are required to make requests to any Verkada API endpoints with the exception of the Get Streaming Token endpoints, which requires a top-level API Key for authentication, as well as the Stream Footage (live or historical) API that requires a JSON Web Token (JWT).

API Tokens inherit permissions from the top-level API key used to generate them and will be limited to that same permission scope. If the API Key used to generate an API Token only has Camera Read-Only permissions, then the associated API Token would only be authorized to call Camera GET endpoints.

API Tokens are valid for 30 minutes and cannot be refreshed. Users will need to call the Get API Token endpoint again to retrieve a new Token if their previous one has expired. When making a call using an expired API Token, users will receive a 401 Authentication Error as well as the following error message:

```
{'id': '0e2d', 'message': 'Token expired', 'data': None}
```

LOG IN TO SEE FULL REQUEST HISTORY

TIME	STATUS	USER AGENT
Make a request to see history.		

Base URL https://api.verkada.com/token

```
curl --request POST \
     --url https://api.verkada.com/token \
     --header 'accept: application/json'
```

Figure 13. API Token Manual Generation

Node Red Message Formatting

Node Red is used to receive the MQTT speed message from OPS9243-A and format it into a message Command expects, including the pertinent IDs. Details on how to set up a Node Red server are not covered in this application note but a good resource is Steve's Internet Guide. The details of the node configurations are provided on the OmniPreSense GitHub site here and available for copying. Messages received from OPS9243-A have a JSON format as shown below:

```
{"label" : "RemudaLn", "time" : "1753023749.901", "direction" : "outbound", "source" : "peak", "speed" : "-18"}
```

The Node Red flow is shown in Figure 14 and consists of two main parts, the input speed messaging formatting for Command and the periodic regeneration of the API Token.

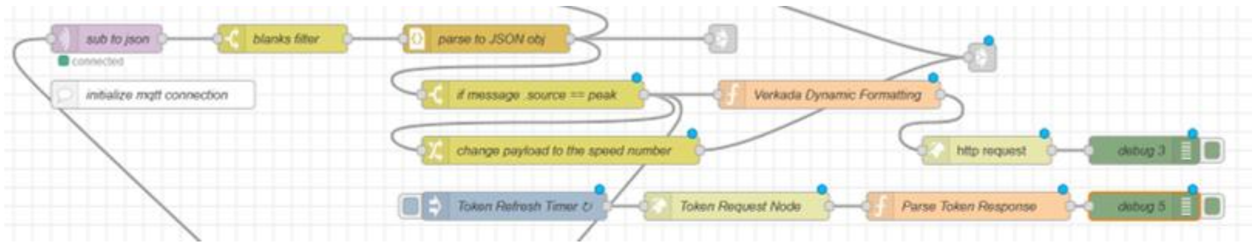


Figure 14. Node Red Flow

The main message flow through the nodes by order of processing are:

1) MQTT in Node: provides configuration to subscribe to MQTT messages published by OPS9243-A
2) Switch Node: validates the message received can be used
3) JSON Node: parse the incoming JSON message for its payload
4) Switch Node: confirms the key word "peak" is part of the message payload
5) Function Node: main processing to convert OPS9243-A into format with IDs required by Command
6) HTTP Request Node: formats message as a POST to Command with the target URL
7) Debug Node: used to confirm message was sent and received correctly

The main function node takes in the payload of the OPS9243-A message and formats it as shown previously in Figure 12. Javscript for this node handles the formatting and insertion of the IDs.

The URL provided by Command is inserted into the message and headers for the header type (application/token) and API Token (**not API Key**) are inserted.

The main payload or attributes are then populated into the message. The speed value is pulled from the payload and converted to its absolute value for passing to Command. Direction and timestamp, likewise, are pulled from the payload with the timestamp being Unix format. Currently, lane and classification are hard coded to a fixed value. With this complete, the message is passed to the HTTP Request node.

The HTTP Request node is configured as a POST with the URL provided earlier by Command. Return is configured to UTF-8 which provides confirmation of a successful data transfer.

A Debug node is provided to confirm messages are accepted by Command and received properly. A return message with statusCode 200 means a successful transfer. A statusCode 400 or similar is an unsuccessful transfer and you should check all the correct IDs are used.

The second part of the Node Red flow is the periodic API Token generation. The nodes utilized in order of processing are:

- Inject Node: used to trigger a new API Token request
- HTTP Request Node: send message to request new API Token
- Function Node: parse the response message from Command for the new API Token
- Debut Node: confirm proper message receipt

The Inject node is configured to trigger every 25 minutes or a little before the current token expires. The Inject node input button is available to immediately generate an API Token and should be used once on initial deployment.

The HTTP Request node is configured as a POST to the URL noted in the Verkada documentation (https://api.verkada.com/token). Within the headers are defined for the API Key passed format of the message returned (application/json).

The Function node contains Javascript to receive the message from Command with the new API Token and make the value globally available. This allows the token to be inserted into the message by the other Function node javascript every time a new API Token is generated.

Once your Node Red flows are configured, deploy them by hitting the red Deploy button in the upper right. With this, every time a speed message comes from the OPS9243-A, it will be reformatted and passed to Command.

Command Speed Results

Once Node Red is up and running, validate that the messages are received in Command. In Command, select the Alerts button again and under Helix Events you should see your Event defined earlier (Figure 15). In this case, we've called it vehicle_speed. Select the Event to see a new screen where the Events and speed data will be populated (Figure 16).

With speed capture now configured you can now set up Alerts for high-speed events and enable LPR mode on the capture to link the speeds with the license plates.

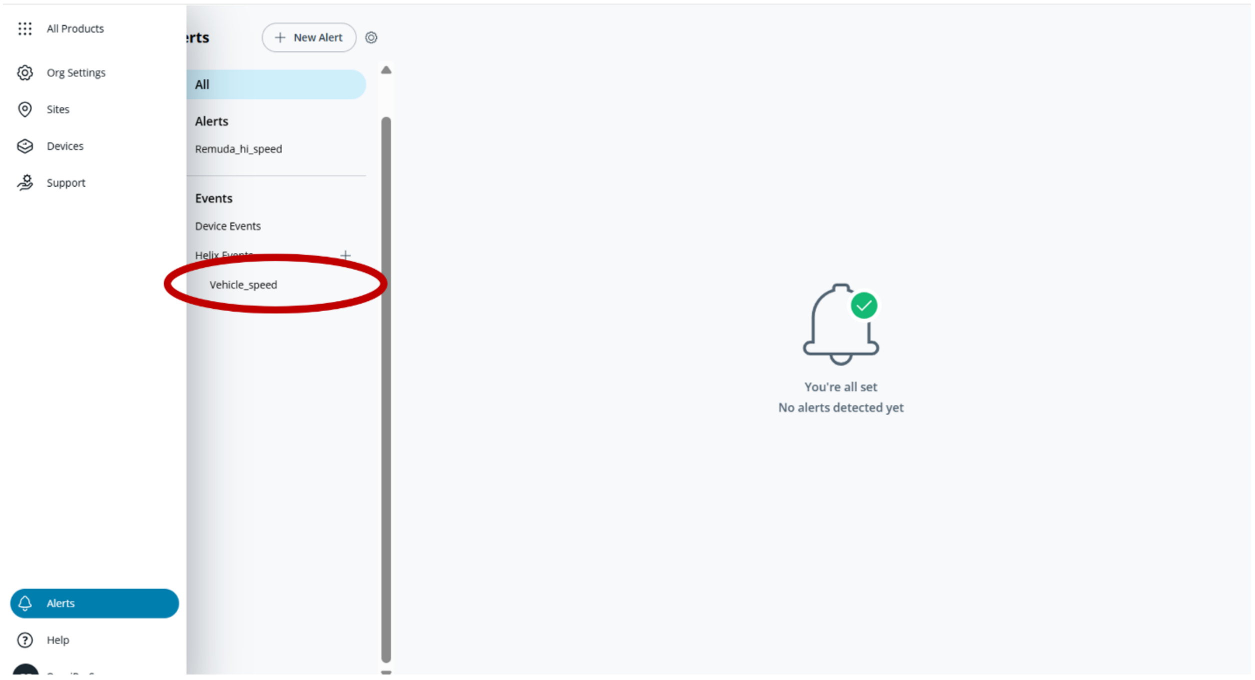
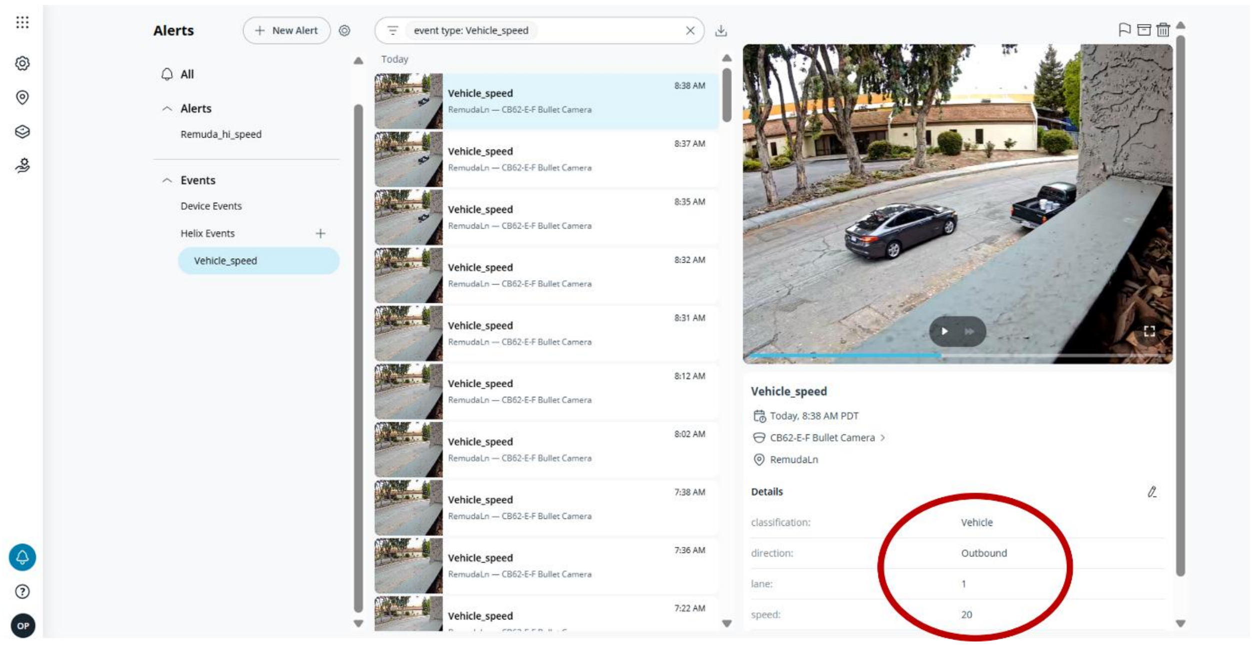


Figure 15. Helix Event – vehicle_speed



event type: Vehicle_speed

Today

Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	8:38 AM
Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	8:37 AM
Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	8:35 AM
Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	8:32 AM
Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	8:31 AM
Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	8:12 AM
Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	8:02 AM
Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	7:38 AM
Vehicle_speed RemudaLn — CB62-E-F Bullet Camera	7:36 AM
Vehicle_speed	7:22 AM

Vehicle_speed

Today, 8:38 AM PDT

CB62-E-F Bullet Camera >

RemudaLn

Details

classification:	Vehicle
direction:	Outbound
lane:	1
speed:	20

Figure 16. Helix Event – Speed Report with Image

Revision History

Version	Date	Description
A	July 24, 2025	Initial release.