

An application programming interface (API) specifies how some software components should interact with each other. The portal's API provides a form of communication via XML or JSON coding language to push your fleet's data (e.g., odometer reading, location, engine hours, etc.) from our system to your back-end systems (e.g., ERP, Dispatch, CRM). By using our API you can effortlessly integrate your fleet's data into your back-end systems to provide transparency into payroll, fuel card transactions, additional documentation, asset management, and more.



**Note.** Please create an API user with the appropriate vehicle access and permissions to use for APIs only. Password changes break the API data from flowing. For tokens, using an API App offers the following benefits: It does not allow access to the Portal site, it can be changed or deactivated by admin users, and app usage can be better tracked.

Think of our API as a pantry full of ingredients. All you have to do is create the recipes!

More details about our **API v2**, including step-by-step instructions and example requests/responses, are available here: <http://gpsinsight.com/apidocs>.

**API Reference**

- Alert
- ApiApp
- Channel
- CustomerSite
- Device
- Dispatch
- Driver
- DriverGroup
- FuelCard
- Garmin
- Geocode
- Heartbeat
- Hierarchy
- HierarchyNode
- Landmark
- LandmarkGroup
- Route
- StopNote
- SmsMessaging
- User
- UserAuth
- Vehicle
- VehicleGroup
- VehicleReport
- VIN
- Webhook

**Vehicle**

Available Methods

Search: Looking for something?

[addMaintenanceAlert](#) Add a maintenance alert for a vehicle

**addMaintenanceAlert**  
Add a maintenance alert for a vehicle

Documentation Testbed

Parameter	Description
vehicle	vehicle vin, serial_number, name or partial name (first to match)
maint_label	
value_offset	(optional)
offset_hrs	(optional)
next_svc_value	(optional)
next_svc_hrs	(optional)
next_svc_date	(optional)

Example Request:  
`https://api.gpsinsight.com/v2/vehicle/addmaintenancealert?session_token=xxxx&vehicle=CA4531009036&maint_label=Inspection&next_svc_value=62429.3`

Example Response:

```
{
  "head": { ... },
  "data": "updated"
}
```



**Tip!** Did you know that our customers can contribute to our API documentation? If they have examples of interesting implementations, they can submit them via [GitHub](#).

## Specific Integrations

Some API integrations have already been defined. To extend the kitchen analogy, the recipe for these types of integrations has not only been created, but the dish is served up and fully baked for you.

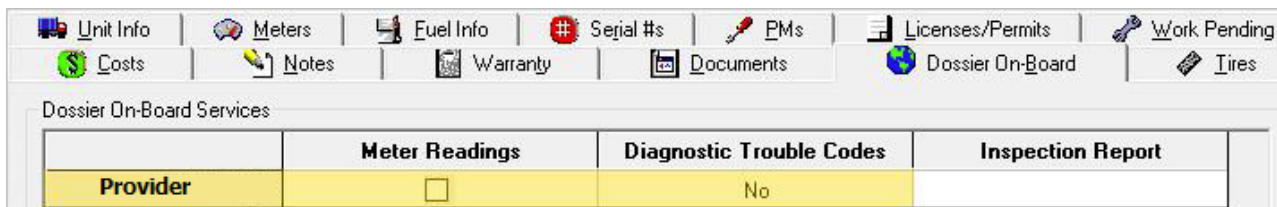
### Dossier DCloud

GPS Insight integration with Dossier provides Odometer and Engine Hours (automatic updating of meter data) and Diagnostic Trouble Codes (automatic import and processing of vehicle DTCs to create a work request).

► **To set up units and schedule them for data transmission:**

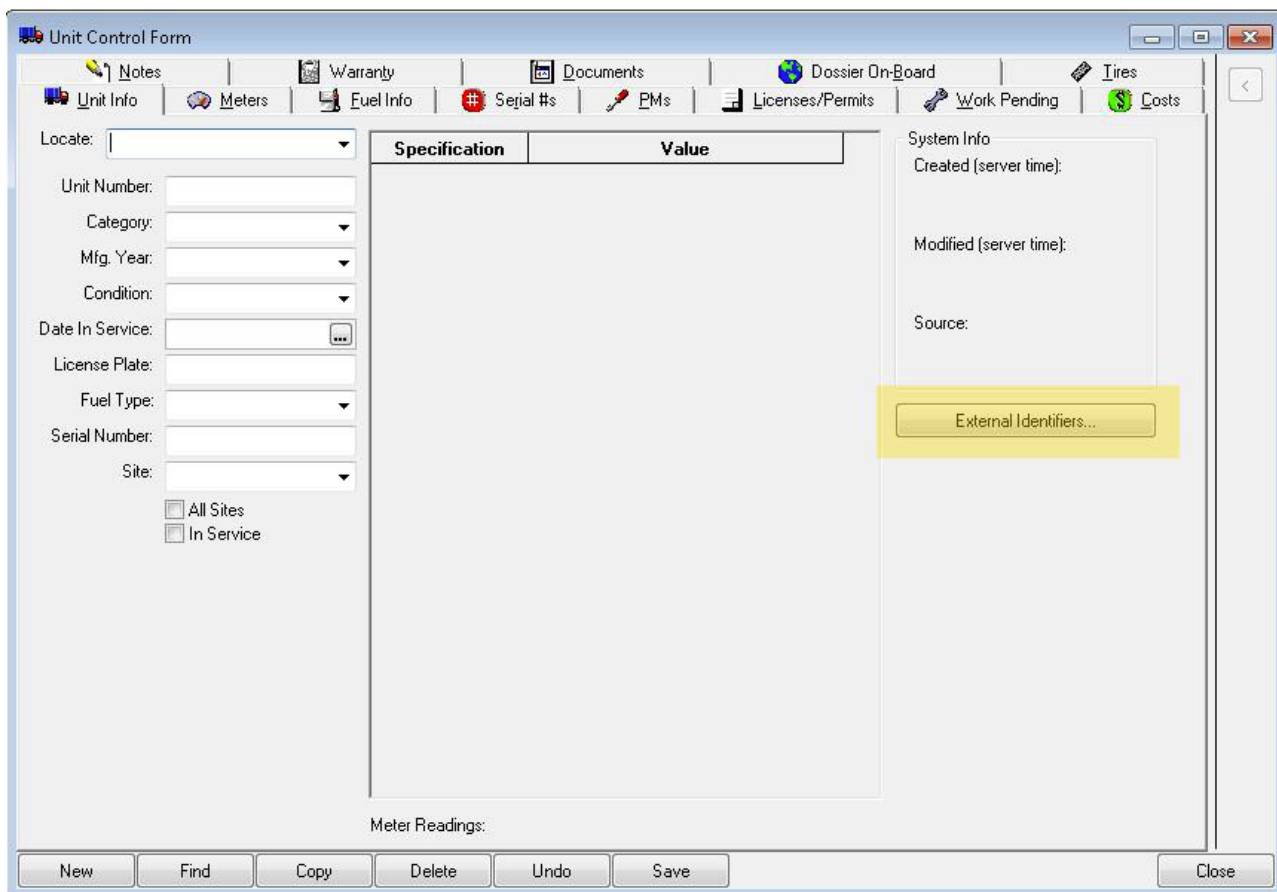
Link: <https://help.nuvo.solutions/docs/about-integration/apis/> Last Updated: June 11th, 2019

1. Configure each on-board unit that will receive GPS data by selecting the desired services on the **Dossier On-Board** tab of the Unit Control Form.



	Meter Readings	Diagnostic Trouble Codes	Inspection Report
Provider	<input type="checkbox"/>	No	

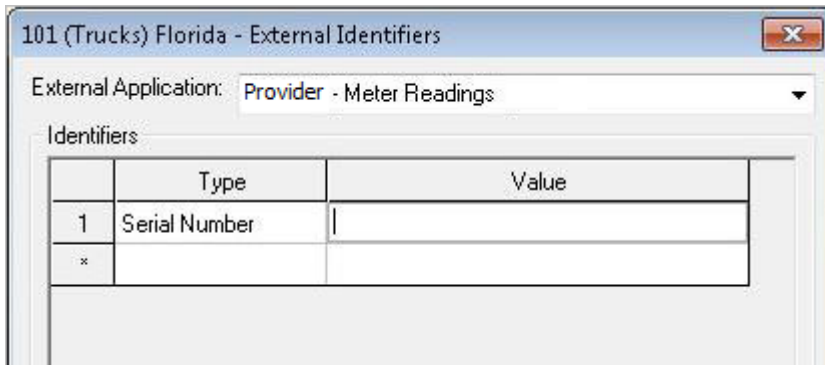
1. After a unit is configured to receive GPS data, click the **Unit Info** tab, and then click **External Identifiers**.



The screenshot shows the Unit Control Form with the Unit Info tab selected. On the left, there are input fields for Unit Number, Category, Mfg. Year, Condition, Date In Service, License Plate, Fuel Type, Serial Number, and Site. Below these are checkboxes for 'All Sites' and 'In Service'. The main area is a table with columns 'Specification' and 'Value'. On the right, there is a 'System Info' section with fields for 'Created (server time)', 'Modified (server time)', and 'Source'. A yellow button labeled 'External Identifiers...' is highlighted. At the bottom, there are buttons for 'New', 'Find', 'Copy', 'Delete', 'Undo', 'Save', and 'Close'.

1. In the External Identifier window, identify the external identifier to use (e.g., Meter Readings or DTCs), and then add the **Serial Number** from the GPS device. You can create an external identifier entry for each type; both use the same Serial Number for the unit.

Link: <https://help.nuvo.solutions/docs/about-integration/apis/> Last Updated: June 11th, 2019

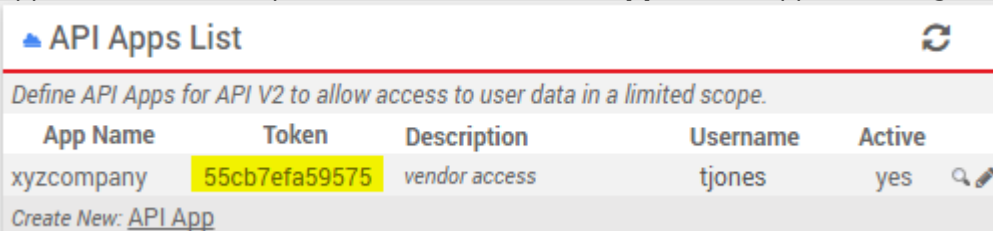


	Type	Value
1	Serial Number	
*		

1. Under the **Configuration** menu, open the Scheduler.
2. Under the Schedule Configuration tab, select the **[Provider] Enabled** check box.
3. Populate the **User Name** and **App Token** fields from the GPS portal.



**Note.** To obtain an App Token, log into the GPS portal. From the Menu Bar, hover over the **Account** menu, point to **Manage Users**, and click **Third-party access to api V2 functions**. In the API Apps List grid, click **API App** next to **Create New**. In the New API App window, enter an App Name and Description, and click **Save API App**. A new App Token is generated.



App Name	Token	Description	Username	Active
xyzcompany	55cb7efa59575	vendor access	tjones	yes

Create New: [API App](#)

For more

information about app tokens, refer to the [API documentation](#).

1. Click **Save**.

Scheduler

Conduit: DCloud Meter Reading Configure

Scheduled Job Name: Meter Reading Schedule

Description:

Schedule Details

Schedule Configuration

Dossier Lite Enabled: ☐

Zonar Enabled: ☐

Networkfleet Enabled: ☐

**[Provider] Enabled: ☒**

Batch Level: Errors And Warnings

Zonar User Name:  ☐

Zonar Password:  ☐

Zonar Customer Code:  ☐

Networkfleet Username:  ☐

Networkfleet Password:  ☐

Networkfleet Hour Reading Type:  ☐

**[Provider] User Name: tjones ☒**

**[Provider] App Token: 123c4567c890 ☒**

Save

New

Delete

Cancel

Open

Refresh

☐ Auto Refresh (30s)

Close