

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.

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:

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

addMaintenanceAlert

Add a maintenance alert for a vehicle

Documentation Testbed

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

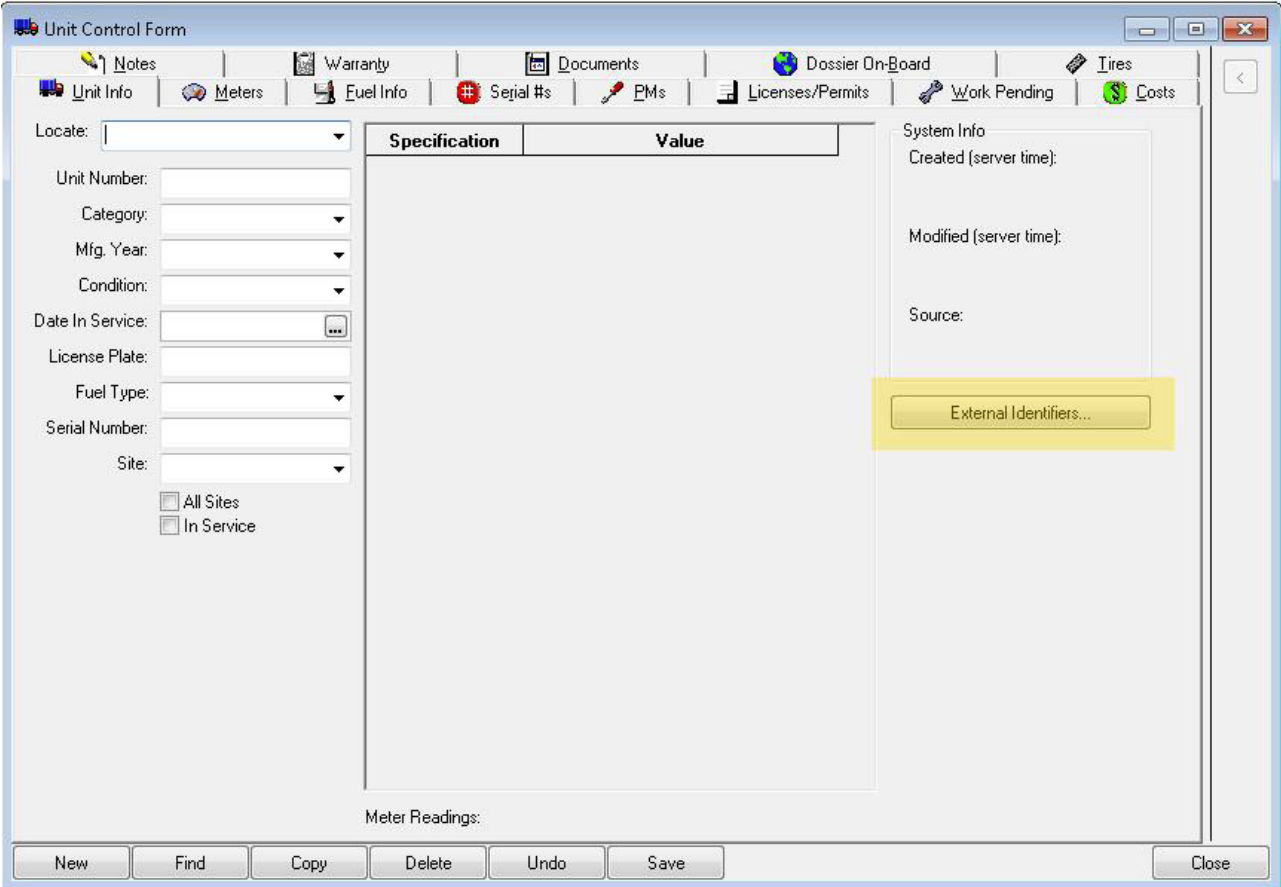
GPS Insight integration with Dossier provides Odometer and Engine Hours (automatic updating of meter data) and Diagnostic Trouble Codes (automatic import and processing of vehicle DTCs to create a work request).

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

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.

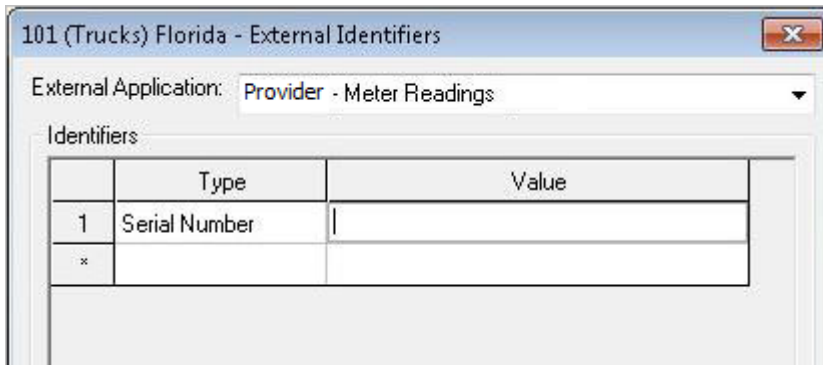
Unit Info	Meters	Fuel Info	Serial #s	PMs	Licenses/Permits	Work Pending
Costs	Notes	Warranty	Documents	Dossier On-Board	Tires	
Dossier On-Board Services						
	Meter Readings	Diagnostic Trouble Codes	Inspection Report			
Provider	<input type="checkbox"/>	No				

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' window. The 'Unit Info' tab is active. 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 'Specification' and 'Value' columns. On the right, there is a 'System Info' section with fields for 'Created (server time)', 'Modified (server time)', and 'Source'. The 'External Identifiers...' button is highlighted in yellow.

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.



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.



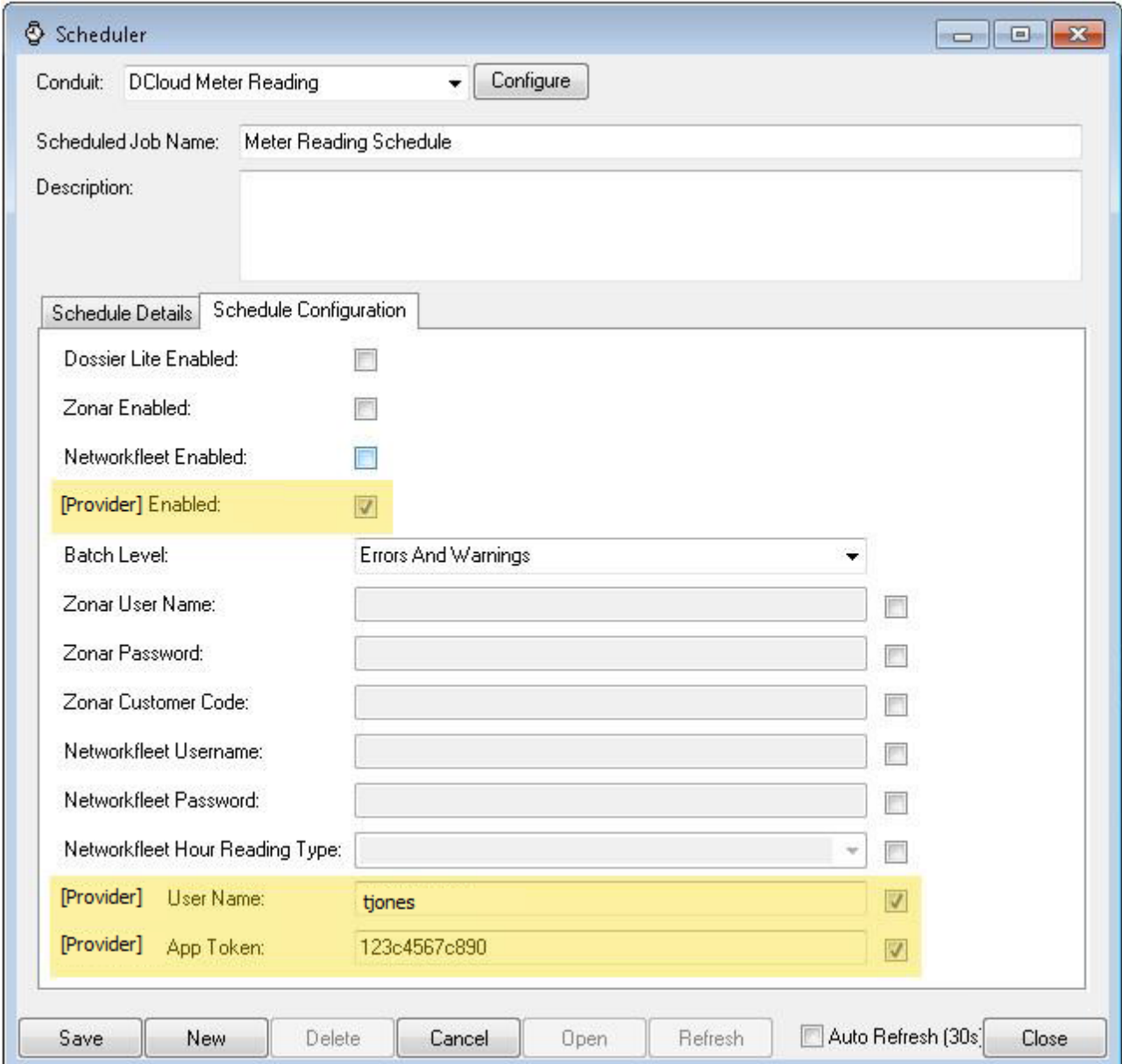
Note. To obtain an App Token, log into the GPS portal. From the Menu Bar, hover over the **Account** menu, point to **Manage Users**, and click **Third-party access to api V2 functions**. In the API Apps List grid, click **API App** next to **Create New**. In the New API App window, enter an App Name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

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

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

For more

1. Click **Save**.



The screenshot shows a 'Scheduler' window with the following configuration:

- Conduit: DCloud Meter Reading (with a 'Configure' button)
- Scheduled Job Name: Meter Reading Schedule
- Description: (empty text area)
- Active tabs: Schedule Details, Schedule Configuration
- Configuration options:
 - Dossier Lite Enabled:
 - Zonar Enabled:
 - Networkfleet Enabled:
 - [Provider] Enabled: (highlighted)
 - Batch Level: Errors And Warnings (dropdown)
 - Zonar User Name: (text input)
 - Zonar Password: (text input)
 - Zonar Customer Code: (text input)
 - Networkfleet Username: (text input)
 - Networkfleet Password: (text input)
 - Networkfleet Hour Reading Type: (dropdown)
 - [Provider] User Name: tjones (highlighted)
 - [Provider] App Token: 123c4567c890 (highlighted)
- Buttons at the bottom: Save, New, Delete, Cancel, Open, Refresh, Auto Refresh (30s), Close