

 [swri-robotics](#) / [novatel\\_gps\\_driver](#) Public

ROS driver for NovAtel GPS / GNSS receivers

 BSD-3-Clause license

 174 stars  137 forks  Branches  Tags  Activity

 Star

 Notifications

 **Code**  Issues **30**  Pull requests  Actions  Projects  Security  Insights



 3 Branches



 28 Tags



 Go to file

Go to file

 Code ▾



**danthony06** 4.2.1 

05fd27d · 7 months ago 



.github/workflows

Removing Iron since it is EOL

11 months ago



novatel\_gps\_driver

4.2.1

7 months ago



novatel\_gps\_msgs

4.2.1

7 months ago



.gitignore

Merge repos on dismount an...

9 years ago



LICENSE

Fix warnings and general cod...

6 years ago



README.md

Added support for CORRIMUS a...

7 months ago

 **README**

 BSD-3-Clause license



# NovAtel GPS Driver

## Overview

This is a C++ [ROS](#) driver for [NovAtel](#) GPS / GNSS Receivers.

Features:

- Fast and robust
- Supports serial, TCP, and UDP connections

- Can play back PCAP capture logs to test behavior
- Supports a variety of common NovAtel logs
- Easy to add support for more log types
- Supports ASCII and binary-format NovAtel logs
- Can synchronize `BESTPOS` , `BESTVEL` , and `PSRDOP2` logs together in order to produce [gps\\_common/GPSFix](#) messages
- Tested with OEM4, OEM6, and OEM7 receivers
- Can produce IMU data from receives with SPAN support

It has been tested primarily on NovAtel OEM628 receivers, but it has been used with various OEM4, OEM6, and OEM7 devices. Please let [the maintainers](#) know of your success or failure in using this with other devices so we can update this page appropriately.

## Usage

The driver should function on ROS 2 Dashing, and binary packages are available for both of them. To install them, first install ROS, then just run:

```
sudo apt-get install ros-dashing-novatel-gps-driver
```



If you'd like to build it from source:

```
mkdir -p novatel/src
cd novatel/src
git clone https://github.com/swri-robotics/novatel_gps_driver
rosdep install . --from-paths -i
cd ../..
colcon build
```



Then create a `.launch.py` file and configure it as desired:

```
"""Launch an example driver that communicates using TCP"""

from launch import LaunchDescription
import launch_ros.actions

def generate_launch_description():
    container = launch_ros.actions.ComposableNodeContainer(
        name='novatel_gps_container',
        namespace='',
        package='rclcpp_components',
```



```

executable='component_container',
composable_node_descriptions=[
    launch_ros.descriptions.ComposableNode(
        package='novatel_gps_driver',
        plugin='novatel_gps_driver::NovatelGpsNode',
        name='novatel_gps',
        parameters=[{
            'connection_type': 'serial',
            'device': '/dev/ttyUSB0',
            'verbose': True,
            'publish_novatel_positions': True,
            'publish_novatel_velocity': True,
            'publish_novatel_psrddop2': True,
            'frame_id': '/gps',
            'loop': True,
        }]
    )
],
output='screen'
)

return LaunchDescription([container])

```

`gps_common/GPSFix` messages will always be published, but by default, other message types are not. See the config parameters for a list of which other message types can be enabled.

## Packages

### 1. `novatel_gps_msgs`

ROS messages that represent NovAtel message logs. Each supported log should have its own message type. A list of official log types can be found at [http://docs.novatel.com/OEM7/Content/Logs/Log\\_Reference.htm](http://docs.novatel.com/OEM7/Content/Logs/Log_Reference.htm).

### 2. `novatel_gps_driver`

A C++ library with an accompanying ROS node and node that can connect to a NovAtel device over a serial, TCP, or UDP connection and translate NovAtel logs into ROS messages.

## Nodelets

### 1. `novatel_gps_driver/novatel_gps_nodelet`

## i. ROS Parameters

- `connection_type` : Type of physical connection to the device
  - One of `serial` , `tcp` , `udp` , or `pcap`
  - Default: `serial`
- `device` : Location of device connection.
  - For `serial` connections, the device node; e. g., `/dev/ttyUSB0`
  - For `tcp` or `udp` connections, a `host:port` specification.
    - If the host is omitted, it will listen for connections on the specified port.
    - If the port is omitted, `3001` will be used as the default for TCP connections and `3002` as the default for UDP connections.
  - For `pcap` connections, the location of a `.pcap` capture file. Note that the node will exit automatically after finishing playback.
  - Default: Empty
- `frame_id` : ROS TF frame to place in the header of published messages.
  - Default: Empty
- `imu_frame_id` : TF frame id to use in IMU messages.
  - Default: Empty
- `imu_rate` : Desired logging rate in Hz for IMU messages.
  - This is set as the rate for `CORRIMUDATA` logs.
  - Default: `100`
- `imu_sample_rate` : Sample rate of the connected IMU. Normally this is automatically detected based on the IMU type.
- `polling_period` : Desired period between GPS messages.
  - This will be set as the period for `GPGGA` , `GPRMC` , `GPGSA` , `BESTPOS` , and `BESTVEL` logs.
  - Default: `0.05` (20 Hz)
- `expected_rate` : Expected publish rate of GPS messages.
  - If time between GPS message stamps is greater than 1.5 times the expected publish rate, diagnostic warning is published.
  - Default: Based on `polling_period` parameter: `20.0` (20Hz)
- `publish_clocksteering` : `true` to publish novatel\_gps\_msgs/ClockSteering messages.
  - Default: `false`
- `publish_diagnostics` : `true` to publish node diagnostics.
  - Default: `true`
- `publish_gpgsa` : `true` to publish novatel\_gps\_msgs/Gpgsa messages.
  - Default: `false`

- `publish_gpgsv : true` to publish `novatel_gps_msgs/Gpgsv` messages.
  - Default: `false`
- `publish_gphdt : true` to publish `novatel_gps_msgs/Gphdt` messages.
  - Default: `false`
- `publish_imu_messages : true` to publish `novatel_gps_msgs/NovatelCorrectedImuData`, `novatel_gps_msgs/Inspva`, `novatel_gps_msgs/Inspvax`, `novatel_gps_msgs/Insstdev`, and `sensor_msgs/Imu` messages.
  - Default: `false`
- `publish_nmea_messages : true` to publish `novatel_gps_msgs/Gpgga` and `novatel_gps_msgs/Gprmc` messages.
  - Default: `false`
- `publish_novatel_dual_antenna_heading : true` to publish `novatel_gps_msgs/NovatelDualAntennaHeading` messages.
  - Default: `false`
- `publish_novatel_heading2 : true` to publish `novatel_gps_msgs/Heading2` messages.
  - Default: `false`
- `publish_novatel_positions : true` to publish `novatel_gps_msgs/NovatelPosition` messages. Note that even if this is false, these logs will always be requested from the receiver in order to generate `gps_msgs/GPSFix` messages.
  - Default: `false`
- `publish_novatel_psrddop2 : true` to publish `novatel_gps_msgs/NovatelPsrddop2` messages. If set, the data from these messages will be used to fill in the DoP values in `gps_msgs/GPSFix` messages. Note that these messages are only published when the values change, not at the standard polling rate.
  - Default: `false`
- `publish_novatel_utm_positions : true` to publish `novatel_gps_msgs/NovatelUtmPosition` messages.
  - Default: `false`
- `publish_novatel_velocity : true` to publish `novatel_gps_msgs/NovatelVelocity` messages. If set, the data from these messages will be used to fill in the speed and track values in `gps_msgs/GPSFix` messages.
  - Default: `false`
- `publish_novatel_xyz_positions : true` to publish `novatel_gps_msgs/NovatelXYZ` messages.
  - Default: `false`
- `publish_range_messages : true` to publish `novatel_gps_msgs/Range` messages.
  - Default: `false`

- `publish_sync_diagnostic` : If true, publish a time Sync diagnostic.
  - Ignored if `publish_diagnostics` is false.
  - Default: `true`
- `publish_dual_antenna_diagnostic` : If true, publish diagnostics for the second antenna.
  - Ignored if `publish_diagnostics` is false.
  - Default: same as `publish_novatel_dual_antenna_heading`
- `publish_time_messages` : `true` to publish novatel\_gps\_msgs/Time messages.
  - Default: `false`
- `publish_trackstat` : `true` to publish novatel\_gps\_msgs/Trackstat messages.
  - Default: `false`
- `publish_invalid_gpsfix` : `true` to publish the `/gps` topic, even if the status of the GPS fix is `STATUS_NO_FIX` .
  - Default: `false`
- `reconnect_delay_s` : Second delay between reconnection attempts.
  - Default: `0.5`
- `serial_baud` : Select the serial baud rate to be used in a serial connection.
  - Default: `115200`
- `spam_frame_to_ros_frame` : Translate the SPAN coordinate frame to a ROS coordinate frame using the VEHICLEBODYROTATION and APPLYVEHICLEBODYROTATION commands.
  - Default: `false`
- `use_binary_messages` : `true` to request binary NovAtel logs, `false` to request ASCII.
  - Binary logs are much more efficient and effectively required for IMU data, but ASCII logs are easier to parse for a human.
  - Default: `false`
- `wait_for_sync` : `true` in order to wait for both BESTPOS and BESTVEL messages to arrive before publishing `gps_msgs/GPSFix` messages. If this is `false` , GPSFix messages will be published immediately when BESTPOS messages are received, but a side effect is that the driver will often be unable to fill in the speed & track fields. This has no effect if `publish_novatel_velocity` is `false` .
  - Default: `true`
- `loop` : `true` to keep replaying PCAP reply. Only effective when device is `pcap` .
  - Default: `false`

## ii. ROS Topic Subscriptions

- `/gps_sync` (*std\_msgs/Time*): (optional) Timestamped sync pulses from a DIO module. These are used to improve the accuracy of the time stamps of the

messages published.

### iii. ROS Topic Publications







- `/bestpos` (*novatel\_gps\_msgs/NovatelPosition*): [BESTPOS](#) logs
- `/bestutm` (*novatel\_gps\_msgs/NovatelUtmPosition*): [BESTUTM](#) logs
- `/bestvel` (*novatel\_gps\_msgs/NovatelVelocity*): [BESTVEL](#) logs
- `/bestxyz` (*novatel\_gps\_msgs/NovatelXYZ*): [BESTXYZ](#) logs
- `/clocksteering` (*novatel\_gps\_msgs/ClockSteering*): [CLOCKSTEERING](#) logs
- `/corrimudata` (*novatel\_gps\_msgs/NovatelCorrectedImuData*): [CORRIMUDATA](#) logs
- `/diagnostics` (*diagnostic\_msgs/DiagnosticArray*): ROS diagnostics
- `/dual_antenna_heading` (*novatel\_gps\_msgs/NovatelDualAntennaHeading*): [DUALANTENNAHEADING](#) logs
- `/fix` ([sensor\\_msgs/NavSatFix](#)): GPSFix messages converted to NavSatFix messages
- `/gpgga` (*novatel\_gps\_msgs/Gpgga*): [GPGGA](#) logs
- `/gpgsa` (*novatel\_gps\_msgs/Gpgsa*): [GPGSA](#) logs
- `/gpgsv` (*novatel\_gps\_msgs/Gpgsv*): [GPGSV](#) logs
- `/gprmc` (*novatel\_gps\_msgs/Gprmc*): [GPRMC](#) logs
- `/gps` ([gps\\_msgs/GPSFix](#)): Fixes produced by combining BESTVEL, PSRDOP2 and BESTPOS messages together
  - **Note:** GPSFix messages will always be published regardless of what other types are enabled.
- `/heading2` (*novatel\_gps\_msgs/NovatelHeadin2*): [HEADING2](#) logs
- `/imu` ([sensor\\_msgs/Imu](#)): CORRIMUDATA logs converted to Imu messages
- `/inspva` (*novatel\_gps\_msgs/Inspva*): [INSPVA](#) logs
- `/inspvax` (*novatel\_gps\_msgs/Inspvax*): [INSPVAX](#) logs
- `/insstdev` (*novatel\_gps\_msgs/Insstdev*): [INSSTDEV](#) logs
- `/psrdop2` (*novatel\_gps\_msgs/Psrdop2*): [PSRDOP2](#) logs
- `/range` (*novatel\_gps\_msgs/Range*): [RANGE](#) logs
- `/rosout` (*roscpp\_msgs/Log*): Console output
- `/time` (*novatel\_gps\_msgs/Time*): [TIME](#) logs
- `/trackstat` (*novatel\_gps\_msgs/Trackstat*): [TRACKSTAT](#) logs

## Adding New Logs

Do you need support for a new log type? Follow these steps:

1. Find the log reference in the [official documentation](#).
2. Add a new .msg file to the `novatel_gps_msgs` package.
3. Add a new class in the `novatel_gps_driver` package that extends the `novatel_gps_driver::MessageParser` class that can parse the log and return the appropriate ROS message.
  - i. Note that most MessageParsers produce `UniquePtr`s to messages because this is more efficient for intraprocess communications. Some MessageParsers have to produce `SharedPtr`s because multiple references to their messages are kept for synchronization or other purposes. Choose the appropriate pointer type based on your needs and look at other messages as examples.
4. Modify the `novatel_gps_driver::NovatelGps` class:
  - i. Add an instance of your parser, a buffer for storing parsed messages, and a method for retrieving them.
  - ii. Modify the `NovatelGps::ParseBinaryMessage`, `NovatelGps::ParseNovatelSentence`, or `NovatelGps::ParseNmeaSentence` methods to use your parser to parse the new message type and store it in the appropriate buffer.
5. Modify the `novatel_gps_driver::NovatelGpsNodelet` class:
  - i. Add a configuration parameter to enable the new message type.
  - ii. Add a publisher for publishing it.
  - iii. Modify `NovatelGpsNodelet::CheckDeviceForData` to retrieve messages from the appropriate buffer and publish them.
6. Add a new unit test to `novatel_gps_driver/tests/parser_tests.cpp` to test your parser.

## Build Status

ROS2 Distro	Branch	Build status	Released packages
Humble	<a href="#">humble</a>	 CI no status build  passing	<a href="#">novatel-gps-driver</a> <a href="#">novatel-gps-msgs</a>
Jazzy	<a href="#">jazzy</a>	 CI no status build  passing	<a href="#">novatel-gps-driver</a> <a href="#">novatel-gps-msgs</a>
		 CI no status build  passing	

## Releases 5


**4.2.1** Latest  
 on Apr 24



[+ 4 releases](#)

---

## Packages

No packages published

---

## Contributors 21



[+ 7 contributors](#)

---

## Languages

