WADGNSS Client User Guide

Wang Hu, Daniel Vyenielo, Farzana Rahman, and Jay A. Farrell

August 2020

Abstract

This document describes software that enables Wide Area Differential Global Navigation System (WADGNSS) implementation for users on a global basis, without the user needing access to a local GNSS reference station. The software functions in a client-server architecture. Typical users will only use the client portion of the software locally on their computer. That client will connect to a remote server. The remote server accesses State Space Representation (SSR) model parameters for ionosphere, satellite position, satellite clock, and satellite hardware biases via the internet that it uses to construct artificial GNSS satellite measurements for a virtual reference station at a user-specified location $\mathbf{P}_b$. The client software runs on the users computer to perform the following tasks:

1. Establishes a connection to the user GNSS receiver. Any type of physical connection (e.g., serial, USP, ethernet) is feasible. The distributed version of the software supports a serial connection implemented via USB. Any type of GNSS receiver with NMEA protocol and RTCM version 3 protocol can be supported. The distributed version of the software supports u-Blox ZED-F9P and u-Blox M8P.
2. Configures the user GNSS receiver for RTCM differential operations.
3. Obtains an initial user position.
4. Establishes an Ethernet connection to the WADGNSS server.
5. Communicates the desired virtual reference station location $\mathbf{P}_b$ to the server. The location of $\mathbf{P}_b$ should be within approximately 20km of the rover location and can be updated if the rover location changes significantly.
6. Redirects the virtual base station measurements from its Ethernet connection with the server to its physical connection with the rover receiver.

This client/server implementation delivers Observation Space Representation (OSR) corrections to the rover in the form of synthetic measurements coputed for the virtual reference station location $\mathbf{P}_b$. Those measurements are distributed in RTCM format through message types 1004 and 1005 using the NTRIP protocol.

At present, the method works only for the GPS constellation. The theoretical approach behind the software extends to other GNSS systems. At present, the SSR data required to compute the OSR corrections is only available in real-time for GPS.

It is available in source code and executable formats at `https://github.com/jaffarrell/WADGNSS`. The client software is released under the MIT license.

Contents

Acknowledgements

This software was developed under funding from the California Department of Transportation under the project "Network Differential GNSS Corrections for Connected and Autonomous Vehicles." (Contract 65A0767). The server software utilizes:

- BKG Ntrip (https://igs.bkg.bund.de/ntrip)
- RTKLIB (https://github.com/tomojitakasu/RTKLIB).

The server software is not currently availble for public release. Neither the BKG Ntrip or RTKLIB software is included in the client.

1 Introduction

The DGNSS computes synthetic measurements applicable to a virtual reference station in the vicinity of the user. The architecture of the system is illustrated in Figure 1. The client software is hosted on a user computer with a direct connection to the users receiver. The purpose of the client software is to facilitate communications between the server and user receiver. At start up, the client establishes a connection with the user receiver, configures the receiver for differential operation, establishes communications with the server, and communicates the users IP address and virtual base location $\mathbf{P}_b$ to the server. After that point, the client receives the virtual base station measurements from the server via Ethernet and sends them to the user receiver over the local connection (e.g. serial port or USB).

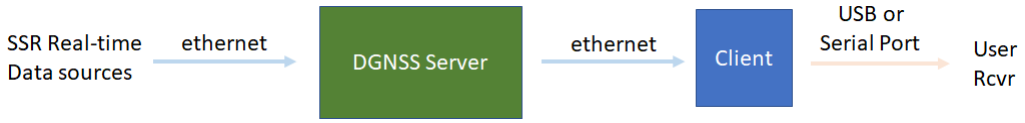

Figure 1: Client Server DGNSS Architecture.

2 Client

In the following sections we outline compilation, installation, GPS receiver setup, and execution of the `WADGNSS_Client` application. We assume most users will read the guide sequentially once and then use it only as a reference document.

We recommend that most users begin from the **Installation** subsection, which describes the procedure for installing a pre-compiled executable application. We also distribute the source code. Users who wish to start from the source code to compile the `WADGNSS_Client` application should begin in the **Compilation** subsection.

2.1 Client Software Compilation

The following procedure outlines compilation on the Ubuntu 18.04 operating system. The procedure may be compatible with other Linux distributions, but is untested as we are actively developing on Ubuntu.

1. Download the necessary prerequisite software to compile `WADGNSS_Client` source. You will need the `cmake` build tool and a `C++` compiler such as `g++`. If not already installed on your local system, you can run the following commands to download the most recent releases.

   ```
   $ sudo apt update && sudo apt upgrade
   $ sudo apt clean && sudo apt auto-remove
   $ sudo apt install make cmake g++
   ```

2. Compile the client software on your local machine.

   (a) Navigate to the `WADGNSS_Client` root directory. The root directory contains a `CMakeLists.txt` file.

   (b) Execute the following to make a build directory and navigate into the newly created build directory.

   ```
   $ mkdir build && cd build
   ```

   (c) Compile the `WADGNSS_Client` by executing the following commands. The first command will configure the `cmake` build system and the second will invoke `make` to start the compilation process.

   ```
   $ cmake .. && make
   ```

   (d) If compilation is successful an executable named `WADGNSS_Client` will be visible in the build directory. The client software is now successfully compiled.

You can now proceed to Section 2.3.

2.2 Client Software Installation

This section describes the recommended installation procedure for the `WADGNSS_Client` executable. At present, we support installation on Ubuntu 18.04 (Ubuntu 18.04.5 LTS). However, we are working to provide support for other operating systems and are focusing our efforts on releasing a Windows compatible client soon.

1. In any directory on your machine create a new directory and navigate into the new directory with the following command.

```
$ mkdir WADGNSS && cd WADGNSS
```

2. Copy the `WADGNSS_Client` executable into the `WADGNSS` directory. The `WADGNSS_Client` executable is now successfully installed.

You can now proceed to Section 2.3.

2.3 GPS Receiver Setup

Before running the `WADGNSS_Client` executable, the user must initialize and configure their receiver using its specific software or other interface. At present, `WADGNSS_Client` software supports automated configuration for u-Blox ZED-F9P and u-Blox M8P. All other receivers need to be configured manually. Please note that `WADGNSS_Client` currently only supports GPS receivers capable of serial input and output communications. Serial messages broadcast from a GPS receiver are used to serve corrections.

The following example on manual configuration is for a u-blox receiver using U-center. Manual configuration of other receivers can be accomplished by configuring the same parameters using the device specific configuration software:

- Enable GPS and disable all non-GPS navigation systems.
- Enable GxGNS message in NMEA protocol.
- Enable message output from the serial port of the GNSS receiver.
- Enable RTCM version 3 message input from the serial port of the GNSS receiver.

An example for u-blox receivers using U-center software is shown below.
Step 1: Enable GPS and set GPS only.
Go to *View* → *Messages View* → *UBX* → *GNSS*. Select enable GPS and disable others. Then click "*Send*"

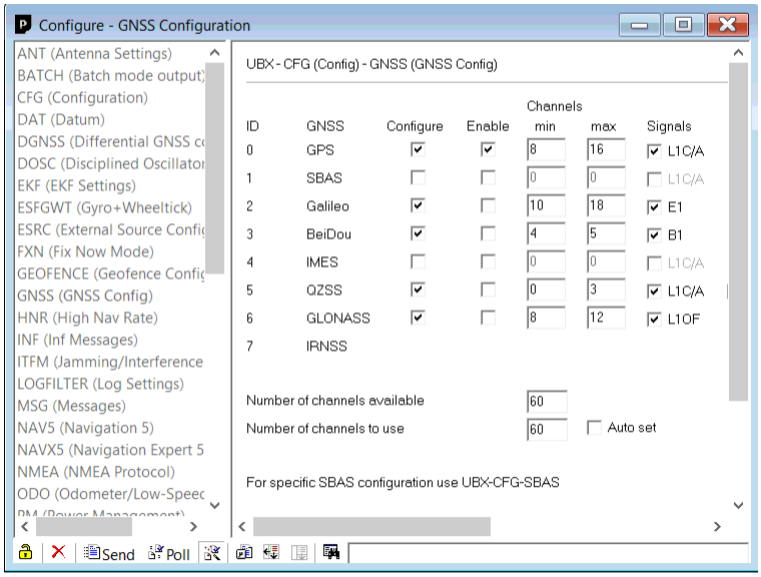

Figure 2: Step 1

Step 2: Enable PUBX 00 via NMEA protocol.
Go to *View* → *Messages View* → *NMEA* → *GxGNS*. Right click "*GxGNS*" then select "*Enable Message*".

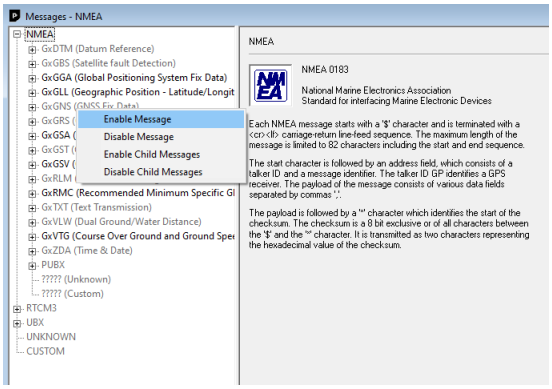

Figure 3: Step 2

Step 3: Enable a receiver USB port for RTCM version 3 input.
Go to *View* → *Messages View* → *UBX* → *CFG* → *CFG* → *PRT*. Select the options based on Fig. 4. Then click "*Send*".

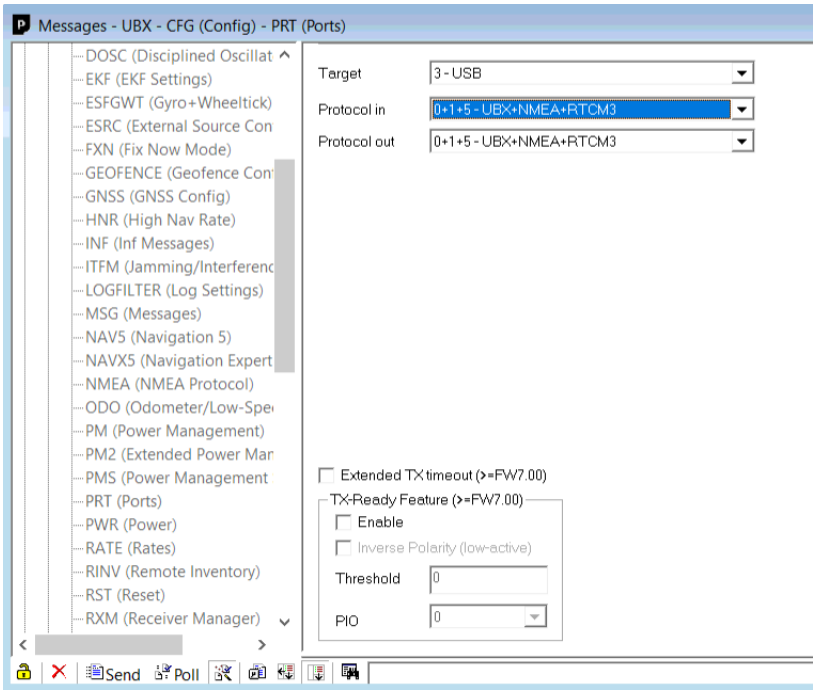

Figure 4: Step 3

We recommend saving the configuration to the gps receiver. If configurations are saved the receiver should remain configured the next time it starts following power off. In U-center, configurations can be saved to the gps receiver by navigating to the *CFG* menu item on the *Configuration View*. Then, select *Save current configuration*. Finally, press the *Send* button. The configurations will save to the receiver. Unfortunately, steps for saving configurations in other gps receiver configuration software will differ.

2.4 `WADGNSS_Client` Execution

The `WADGNSS_Client` executable expects four command line arguments: com port, configuration option, server IP and server port. An example usage template is shown below.

```
$ ./WADGNSS_Client [com port] [configuration] [server IP] [server port]
```

The following sections discuss these command line parameters and how to find the correct values for your system.

2.4.1 Parameter: `[com port]`

The `[com port]` option designates on which of the computers `USB` communication ports the `WADGNSS_Client` software receive messages from the GPS receiver. Determining port number to which the GPS receiver is broadcasting is operating system specific. It may take a couple attempts to discover the correct filepath.

In Ubuntu, serial data from the GPS receiver is available at a filepath matching the pattern \dev\ttyACMX where X is a number in the range [0-9]. For example, data from our receiver might be available at \dev\ttyACM0 so we provide the filepath \dev\ttyACM0 in place of the [com port] argument.

In Windows, the com port can be determined by opening Device Manager and expanding Ports (COM & LPT) option. For example, suppose your GPS receiver is visible as device COM6. Then, the filepath COM6 is provided in place of the [com port] argument.

2.4.2 Parameter: [configuration]

The [configuration] parameter indicates the desired configuration of the supported GPS receiver. If supported, the WADGNSS_Client will initialize the connected GPS receiver to the corresponding configuration. We currently support these automated configuration options:

- 0: Receiver configured by its software,
- 1: u-blox M8P module
- 2: u-blox ZED-F9P module

2.4.3 Parameters: [server IP] and [server port]

The [server IP] and [server port] parameters indicate the network location of the WADGNSS_Server to which the WADGNSS_Client will connect. We recommed connecting to the official WADGNSS server hosted at IP pppdgnss.engr.ucr.edu on port 2101. Substitute these values for [server IP] and [server port] parameters when executing the WADGNSS_Client, respectively.

2.4.4 Example Execution

Bringing all this together, an example execution command for the WADGNSS_Client where USB data is available at \dev\ttyACM0, the connected GNSS receiver is u-blox M8P, and the client will connect to a WADGNSS_Server at IP pppdgnss.engr.ucr.edu on port 2101 is as follows:

```
$ ./WADGNSS_Client \dev\ttyACM0  1  pppdgnss.engr.ucr.edu  2101
```

Congratulations you have successfully launched the WADGNSS_Client. To stop the WADGNSS_Client press "ctrl + c" keys, simultaneously.

2.5 Client Output Files

The client produces two output files.

ROVERLOG_20201022.log: This file contains the position information output by the user receiver (i.e., the rover). For the default configuration (as distributed), the rover outputs the $GxGNS messages in NMEA protocol. The user may edit the client source code to change the receiver output messages or to redirect such receiver output information through computer ports to other programs or devices.

running_log.log: This file contains client connection status messages. For example, a typical sequence of start-up messages is as follows:

```
[2020-09-23 03:31:02]: Start connect receiver.
[2020-09-23 03:31:02]: Succeed.
[2020-09-23 03:31:03]: Initialize u-blox type: 2 (1 for M8P, 2 for ZED-F9P).
[2020-09-23 03:31:03]: Build connection with server succeed.
[2020-09-23 03:31:03]: Start read pos info ($GxGNS) from rover.
[2020-09-23 03:31:04]: Send to server: $POSECEF -2427457.4792 -4709287.4204 3540190.5200
```

The first three messages relate to establishing communications with and initializing the receiver. The last three messages related to establishing communications with the server and sending it the position $\mathbf{P}_b$.

