A NovAtel Precise Positioning Product

Inertial Explorer®
User Guide

OM-20000106 Rev 10 November 2014

Inertial Explorer User Guide

Publication Number: OM-20000106

Revision Level: 10

Revision Date: November 2014

This manual reflects Inertial Explorer software version 8.60.

Warranty

NovAtel Inc. warrants that its GNSS products are free from defects in materials and workmanship, subject to the conditions set forth on our web site: www.novatel.com/products/warranty/ and for the following time periods:

Software Warranty	One (1) Year
Computer Discs	Ninety (90) Days

Return instructions

To return products, refer to the instructions on the Returning to NovAtel tab of the warranty page: www.novatel.com/products/warranty/.

Proprietary Notice

Table of Contents

Software License

BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE PRODUCT, YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE WITH THESE TERMS OF USE, DO NOT INSTALL, COPY OR USE THIS ELECTRONIC PRODUCT (SOFTWARE, FIRMWARE, SCRIPT FILES, OR OTHER ELECTRONIC PRODUCT WHETHER EMBEDDED IN THE HARDWARE, ON A CD OR AVAILABLE ON THE COMPANY WEB SITE) (hereinafter referred to as "Software").

1. **License:** NovAtel Inc. ("NovAtel") grants you a non-exclusive, non-transferable license (not a sale) to use the software subject to the limitations below. You agree not to use the Software for any purpose other than the due exercise of the rights and licences hereby agreed to be granted to you.

2. **Copyright:** NovAtel owns, or has the right to sublicense, all copyright, trade secret, patent and other proprietary rights in the Software and the Software is protected by national copyright laws, international treaty provisions and all other applicable national laws. You must treat the Software like any other copyrighted material and the Software may only be used on one computer at a time. No right is conveyed by this Agreement for the use, directly, indirectly, by implication or otherwise by Licensee of the name of NovAtel, or of any trade names or nomenclature used by NovAtel, or any other words or combinations of words proprietary to NovAtel, in connection with this Agreement, without the prior written consent of NovAtel.

3. **Patent Infringement:** NovAtel shall not be liable to indemnify the Licensee against any loss sustained by it as the result of any claim made or action brought by any third party for infringement of any letters patent, registered design or like instrument of privilege by reason of the use or application of the Software by the Licensee or any other information supplied or to be supplied to the Licensee pursuant to the terms of this Agreement. NovAtel shall not be bound to take legal proceedings against any third party in respect of any infringement of letters patent, registered design or like instrument of privilege which may now or at any future time be owned by it. However, should NovAtel elect to take such legal proceedings, at NovAtel's request, Licensee shall co-operate reasonably with NovAtel in all legal actions concerning this license of the Software under this Agreement taken against any third party by NovAtel to protect its rights in the Software. NovAtel shall bear all reasonable costs and expenses incurred by Licensee in the course of co-operating with NovAtel in such legal action.

4. **Restrictions:**

You may not:

(a)use the software on more than one computer simultaneously;

(b)distribute, transfer, rent, lease, lend, sell or sublicense all or any portion of the Software without the written permission of NovAtel;

(c)alter, break or modify the hardware protection key (dongle) thus disabling the software copy protection;

(d)modify or prepare derivative works of the Software;

(e)use the Software in connection with computer-based services business or publicly display visual output of the Software;

(f)implement DLLs and libraries in a manner that permits automated internet based post-processing (contact NovAtel for special pricing);

(g)transmit the Software over a network, by telephone or electronically using any means (except when downloading a purchased upgrade from the NovAtel web site); or

(h)reverse engineer, decompile or disassemble the Software.

NovAtel retains the right to track Software usage for detection of product usage outside of the license terms.

You agree to keep confidential and use your best efforts to prevent and protect the contents of the Software from unauthorized disclosure or use.

5. **Term and Termination:** This Agreement and the rights and licences hereby granted shall continue in force in perpetuity unless terminated by NovAtel or Licensee in accordance herewith. In the event that the Licensee shall at any time during the term of this Agreement: i) be in breach of its obligations hereunder where such breach is irremediable or if capable of remedy is not remedied within 30 days of notice from NovAtel requiring its remedy; then and in any event NovAtel may forthwith by notice in writing terminate this Agreement together with the rights and licences hereby granted by NovAtel. Licensee may terminate this Agreement by providing written notice to NovAtel. Upon termination, for any reasons, the Licensee shall promptly, on NovAtel's request, return to NovAtel or at the election of NovAtel destroy all copies of any documents and extracts comprising or containing the Software. The Licensee shall also erase any copies of the Software residing on Licensee's computer equipment. Termination shall be without prejudice to the accrued rights of either party, including payments due to NovAtel. This provision shall survive termination of this Agreement howsoever arising.

6. **Warranty:** NovAtel does not warrant the contents of the Software or that it will be error free. The Software is furnished "AS IS" and without warranty as to the performance or results you may obtain by using the Software. The entire risk as to the results and performance of the Software is assumed by you. See product enclosure, if any for any additional warranty.

7. **Indemnification:** NovAtel shall be under no obligation or liability of any kind (in contract, tort or otherwise and whether directly or indirectly or by way of indemnity contribution or otherwise howsoever) to the Licensee and the Licensee will indemnify and hold NovAtel harmless against all or any loss, damage, actions, costs, claims, demands and other liabilities or any kind whatsoever (direct, consequential, special or otherwise) arising directly or indirectly out of or by reason of the use by the Licensee of the Software whether the same shall arise in consequence of any such infringement, deficiency, inaccuracy, error or other defect therein and whether or not involving negligence on the part of any person.

8. **Disclaimer and Limitation of Liability:**

(a)**THE WARRANTIES IN THIS AGREEMENT REPLACE ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. NovAtel DISCLAIMS AND EXCLUDES ALL OTHER WARRANTIES. IN NO EVENT WILL NovAtel's LIABILITY OF ANY KIND INCLUDE ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, EVEN IF NOVATEL HAS KNOWLEDGE OF THE POTENTIAL LOSS OR DAMAGE.**

(b)NovAtel will not be liable for any loss or damage caused by delay in furnishing the Software or any other performance under this Agreement.

(c)NovAtel's entire liability and your exclusive remedies for our liability of any kind (including liability for negligence) for the Software covered by this Agreement and all other performance or non-performance by NovAtel under or related to this Agreement are to the remedies specified by this Agreement.

9. **Governing Law:** This Agreement is governed by the laws of the Province of Alberta, Canada. Each of the parties hereto irrevocably attorns to the jurisdiction of the courts of the Province of Alberta.

10. **Customer Support:** For Software UPDATES and UPGRADES, and regular customer support, see *Customer Service* on page 8.

Foreword

Congratulations on purchasing Waypoint® Products Group's Inertial Explorer®.

Inertial Explorer is a Windows-based suite of programs that provide GNSS (Global Navigation Satellite System) and inertial data post-processing. This manual will help you install and navigate your software.

Scope

This manual contains information on the installation and operation of Inertial Explorer. It allows you to effectively navigate and post-process GNSS, IMU (Inertial Measurement Unit) and wheel sensor data. It is beyond the scope of this manual to provide details on service or repair, see the *Customer Service* on page 8 for customer support.

How to use this manual

This manual is based on the menus in the interface of Inertial Explorer. It is intended to be used in conjunction with the most recent revision of the *GrafNav/GrafNet® User Guide* found on the NovAtel web site and the corresponding version of Waypoint's Inertial Explorer software.

Prerequisites

To run Waypoint software packages, your personal computer must meet or exceed this minimum configuration:

Operating System

Windows Vista, 7, 8 or 8.1.

Hard Drive Space

90 MB of available space on the hard disk.

Processor

A Pentium or Xeon processor is required. Simultaneous forward/reverse processing is possible on dual CPU and Xeon systems. At least 256 MB of RAM is also required.

Although previous experience with Windows is not necessary to use Waypoint software packages, familiarity with certain actions that are customary in Windows will assist in using the program. This manual has been written with the expectation that you already have a basic familiarity with Windows.

Conventions

This manual covers the full performance capabilities of Inertial Explorer 8.60 data post processing software. The following conventions are used in this manual:

✉ This is a note box that contains important information before you use a command or log, or to give additional information afterwards.

This manual contains shaded boxes on the outside of the pages. These boxes contain procedures, screen shots, tables and quick references.

Customer Service

If the software was purchased through a vendor, please contact them for support. Otherwise, for software updates and customer service, contact NovAtel's Waypoint Products Group using the following methods:

Call: U.S. & Canada 1-800-NovAtel (1-800-668-2835)
China 0086-21-54452990-8011
Europe 44-1993-848-736
SE Asia and Australia 61-400-883-601

Email: support@novatel.com

Web: www.novatel.com

Write: NovAtel Inc.
Customer Service Department
1120-68 Avenue NE
Calgary AB
Canada, T2E 8S5

Chapter 1 Inertial Explorer

1.1 Overview of Inertial Explorer

Inertial Explorer builds upon NovAtel's GNSS-only processor, GrafNav. Inertial Explorer shares a similar interface to GrafNav but also includes IMU processing capabilities. Both loosely coupled and tightly coupled are supported for both differential and Precise Point Positioning (PPP).

Inertial Explorer is well integrated with NovAtel SPAN products, however support is also available for processing third party IMU data. Inertial Explorer comes pre-configured with aerial, ground vehicle and marine processing profiles as well as a New Project Wizard that helps new customers get started quickly.

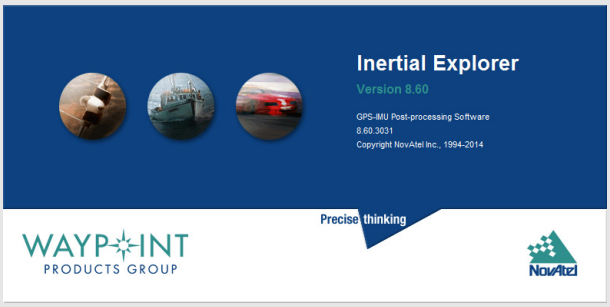
✉ This manual assumes the use of the *GrafNav/GrafNet 8.60 User Guide* which is available on our website at www.novatel.com/support/info/documents/572

1.2 Getting Started with Inertial Explorer

This section provides step-by step procedures on how to process data in Inertial Explorer.

Installation

Verify that the installation was successful by ensuring that you have a *Waypoint Inertial Explorer 8.60* program group on your computer. If this program group is not there, refer to the *GrafNav/GrafNet 8.60 User Guide* for installation instructions.



How to start Inertial Explorer

1. Verify installation.
2. Click on *Inertial Explorer* to start the program.

How to convert SPAN IMU data

1. Open the *Convert Raw GNSS to GPB* utility through *File* | *Convert* | *Raw GNSS to GPB*.
2. Click the *Get Folder* button and navigate to the directory containing the raw data.
3. Click the *Auto Add All* button to automatically add any detected raw GNSS data files to the *Convert Files* list.
4. Select *Convert*. All available GNSS, INS, DMI and dual antenna heading data will be converted.

How to convert third party IMU data

✉ Before using the *Waypoint IMU Data Conversion* tool, a generic IMU file must be formed as this is used as input. See *Table 2, Binary Structure of Raw Data* on page 41 for the format of this file.

1. Open the conversion utility via *File* | *Convert* | *Raw IMU Data to Waypoint Generic (IMR)*.
2. Click the *Browse* button to locate the raw IMU data file.
3. Under the *IMU Profiles* box, select the appropriate conversion profile.
4. Click *Convert* to create the IMR file. See *Chapter 2, Conversion Utilities* on page 37 for more information.
5. Add the file to the project via *File* | *Add IMU File*.

Convert and Process GNSS Data

Refer to the *GrafNav/GrafNet 8.60 User Guide* to process GNSS data. The only exception is that the new project is created in Inertial Explorer, not GrafNav.

✉ For NovAtel SPAN users, all available GNSS, INS, DMI and dual antenna heading data will automatically be extracted by the GNSS converter.

Convert IMU Data

IMU data must be converted to Waypoint's generic IMR format for processing. To do this, follow the steps in the shaded box.

✉ NovAtel SPAN users do not have to follow these steps because this is done automatically when converting the raw GNSS data.

Process SPAN IMU Data

The steps for processing SPAN IMU data are in the shaded box.

How to process SPAN IMU data

1. Click the *Process* menu and then select *Process LC (Loosely Coupled)* or *Process TC (Tightly Coupled)*.

✉ If you are processing in loosely-coupled mode, make sure that you have processed the GNSS data first.

2. Ensure that an appropriate processing profile for your application (aerial, ground vehicle or marine) and your SPAN IMU type have been automatically selected. If this needs to be changed, use the profile pull down menu.
3. Ensure the IMU to GNSS lever arm has been correctly loaded. If the IMU to GNSS lever arm was not saved during data collection, enter it.
4. Ensure the body to IMU rotation has been correctly loaded. If the body to IMU rotation was not saved during data collection, enter it.
5. Click *Process.*

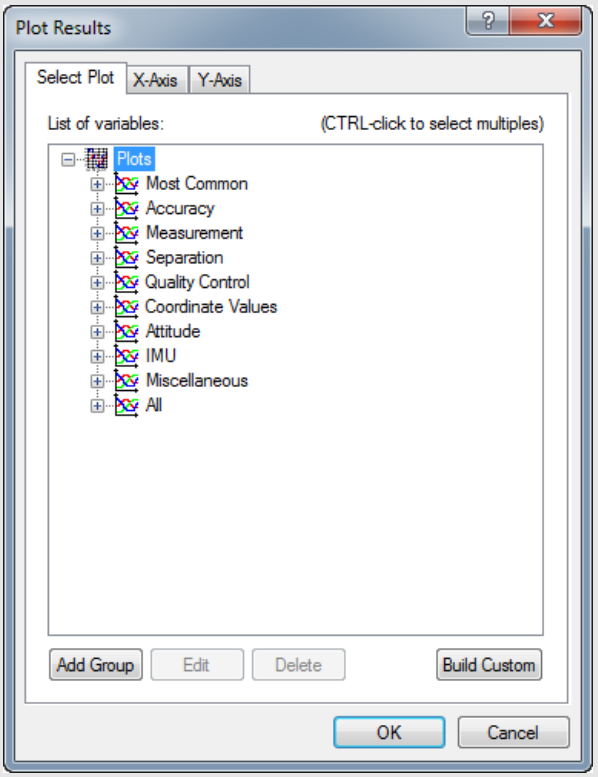
✉ The *Process* option on the LC and TC catalog features two additional options:

Process without pre-processing:
Using this option skips Inertial Explorer's pre-processing checks prior to processing. It may be necessary to use this option if you would like to ignore a critical pre-processing warning which disables processing.

Solve Lever Arm:
Using this option adds the X, Y and Z IMU to GNSS lever arm states to Inertial Explorer's Kalman filter. After processing and smoothing (if automated smoothing has been enabled within the *Solution* tab of *Settings* | *Preferences*), the best converged estimate of the lever arm is automatically reported after processing.

If processing *both* or *multi-pass* directions, both forward and reverse estimates are reported. Note that several iterations may be required for the lever arm to converge and Inertial Explorer's ability to estimate lever arm values is dependent on the amount of data collected and vehicle dynamics.

✉ If processing third party IMU data, an IMU error model must first be developed. It is also recommended to create a custom processing profile in order to automatically load all preferred processing settings.



Plotting and Quality Control

Once processing is complete, view the quality of the results by analyzing the IMU plots. Under the *Output* menu, choose *Results* to access the following IMU plots:

Attitude (Roll and Pitch)
This plot shows the roll and pitch profile of the processed IMU data.

Attitude (Azimuth/Heading)
This plot shows the heading/azimuth of the IMU and the GNSS course-over-ground (COG). They should be in reasonable agreement when the vehicle is moving forward.

If the GNSS COG and the IMU azimuth are biased by a large and constant amount (i.e. +/- 90 degrees or 180 degrees) it indicates the IMU sensor frame has not been rotated to the vehicle body frame (Y-forward, X-right and Z-up). If this is not intentional, a body to IMU rotation should be applied in the LC or TC processing dialog boxes. This will ensure Inertial Explorer's output roll, pitch and heading values are referenced to the vehicle frame.

Attitude Separation
This plot requires that forward and reverse have both been processed. It shows the difference between their attitude values. Ideally, they should agree to a reasonable level considering the quality of the IMU and the dynamics of the survey.

IMU-GNSS Position Misclosure
This plot shows the difference between the GNSS solution projected to the IMU center of navigation and the mechanized INS positions obtained from GNSS/INS processing. They should agree to a reasonable level, largely dependent on the quality of the GNSS trajectory (i.e. severity of the GNSS signal conditions).

✉ Use the *Build Custom* button to add some of the these plots to a customized list.

✉ Consider adding your preferred Q/C plots to a group using the *Add Group* button. Your custom plot group will appear under the *Grouped Plots* list. When plotting a group, all plots within that group are simultaneously plotted.

Smooth Solution
By default, Inertial Explorer's backward smoother is automatically run after processing. This is needed to produce the best possible solution and also to generate all necessary high rate binary files that are required by the Export Wizard.

Automatic smoothing can be controlled through an option within the *Solution* tab of *Settings* | *Preferences*. If this has been disabled, it is required to run the Smoother (*Process* | *Smooth Solutions*) prior to using the Export Wizard.

Export Final Coordinates

The steps for exporting final coordinates are in the shaded box.

How to export final coordinates

1. Select *Output* | *Export Wizard*.
2. Specify the source for the solution.
   Epochs outputs the trajectory, while *Features/ Stations* exports positions only for loaded features, such as camera marks.
3. Select a profile.
4. Click *Next*.
   The Export Wizard will prompt you for all necessary information depending on the contents of your export profile.

✉ If the Export Wizard prompts you for a geoid, the geoid can be downloaded from the NovAtel website at: www.novatel.com/support/waypoint-support/waypoint-geoids/.

Project Wizard Steps

1. Create and name the project
2. Add the rover data to the project.

✉ The rover data can be in Waypoint's GPB format or in the receiver's raw format. If the data is in the receiver's raw format, the Wizard converts it to GPB for you.

If you are a NovAtel SPAN user and you add a raw data file, the Wizard automatically detects the IMU model for conversion to IMR format.

3. Add the base station data to the project.

✉ You can add your own local base station data (in raw or GPB format) or you can have the Wizard download free service data from the Internet.

If you plan to process with PPP, you can skip the previous step and download the precise satellite clock and orbit files from the Internet.

1.3 File Menu

Refer to the *GrafNav/GrafNet 8.60 User Guide* for information on the features available via this menu. The points relevant to Inertial Explorer are discussed in this section of the manual.

1.3.1 New Project

Project Wizard

The Project Wizard offers you a guided step-by-step way of creating a project. The *Project Wizard* steps are listed in the shaded box.

1.3.2 Add Master File(s)

Up to eight base stations can be added to a single Inertial Explorer project. We recommend adding additional base station data only if each base station is in a distinctly different project area and is at some point the closest in the trajectory.

All data must be converted to GPB prior to adding as a base station. When adding a base station, take care to verify base station coordinates and datum as this is critical to absolute position accuracy.

1.3.3 Add Remote File

Only one remote file can be added to an Inertial Explorer project. The file must be converted to GPB prior to adding it to the project. When selecting the remote GPB file, Inertial Explorer will automatically check for any associated IMU data (*.imr file), DMI data (*.dmr file), heading data (HMR file) and mount data (*.mmr) and prompt you whether you would like to add this data to the project as well.

When adding a remote GPB file, ensure the *Measured height* of the antenna is set to zero. Inertial Explorer uses the entered IMU to GNSS lever arm in order to transfer the GNSS position updates to the IMU center of navigation during processing. A vector can be entered from the IMU to any other sensor or point of interest on the vehicle during Export to transfer position data.

1.3.4 Add IMU File

Only one IMR file can be added to an Inertial Explorer project. This menu item is not often needed as IMR data will be automatically added to the project when adding the remote GPB file, provided it is in the same directory and has the same name as the remote GPB file.

1.3.5 Load

LC Solution (Loosely Coupled)
Loads the loosely coupled solution.

TC Solution (Tightly Coupled)
Loads the tightly coupled solution.

1.3.6 Convert

Raw GNSS to GPB
Raw GNSS data must be converted to GPB format for processing. Refer to the *GrafNav/GrafNet 8.60 User Guide*.

Raw IMU Data to Waypoint Generic (IMR)
IMU data must be converted to IMR format in order to be processed by Inertial Explorer. Use this utility to perform this conversion. See *Convert IMU Data* on page 10 for more information.

NovAtel SPAN customers do not need to use this option. All NovAtel data (including all raw GNSS and IMU data) is automatically converted within the Raw GNSS Data Converter.

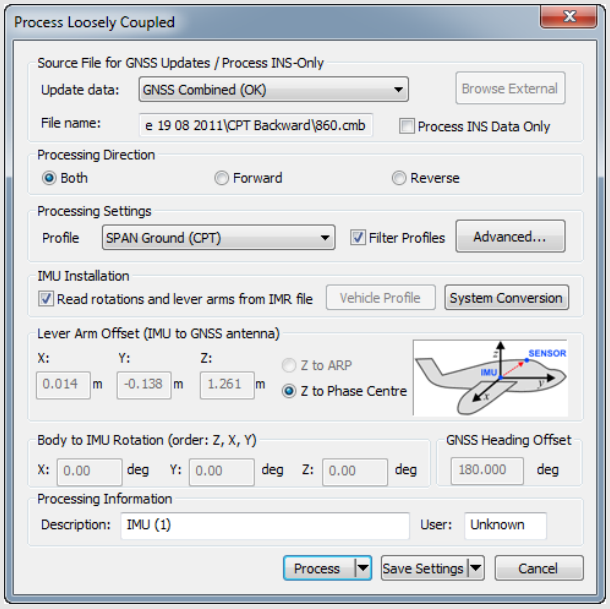
GPB to RINEX
This option converts GPB files to a RINEX file. It supports the creation of Version 2.0 and 2.11 of the RINEX format. For additional information, refer to the *GrafNav/GrafNet 8.60 User Guide*.

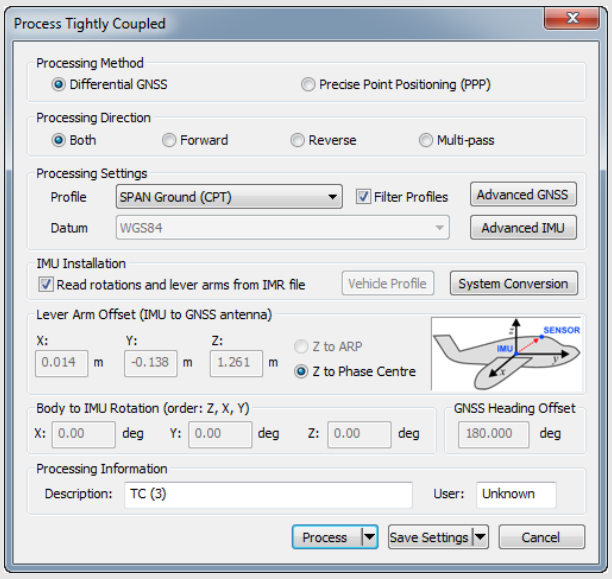
1.3.7 Removing Processing Files

This utility removes all the files associated with any given project. Refer to the *GrafNav/GrafNet 8.60 User Guide* for details of this utility. The Inertial Explorer capabilities are discussed here.

Files to Remove
Selects files to remove from the project or folder.

Inertial Processing (LC and TC)
Removes all ASCII and binary files created during IMU processing, including message logs and trajectories.





1.4 View Menu

Refer to the *GrafNav/GrafNet 8.60 User Guide* for a description of all the features available in this menu

✉ In Inertial Explorer, view IMU message logs and trajectory files under *View | Forward Solution* and *View | Reverse Solution*. See *Section 3.2, File Formats* on page 42 for information on file formats.

1.5 Process Menu

Refer to the *GrafNav/GrafNet 8.60 User Guide* for information regarding all of the features available from this menu. Only those features that are exclusive to Inertial Explorer are discussed here.

1.5.1 Process LC (Loosely Coupled) and TC (Tightly Coupled)

This window provides access to most settings related to IMU processing.

Source File for GNSS Updates (LC Processing)

Update Data

Use this option to select the GNSS file from which Inertial Explorer obtains updates. In most cases, the combined solution is suggested. However, you may specify an alternate file by selecting *External trajectory* from the drop-down menu and clicking the *Browse External* button.

File Name

Displays the selected file that will be used for updates.

Process INS Data Only

This option disables the use of GNSS data during INS processing. Updates will only be performed with user-entered coordinate updates.

✉ This mode of processing is not recommended. It is only used for special applications, such as pipeline pigging.

Process Settings

Profile

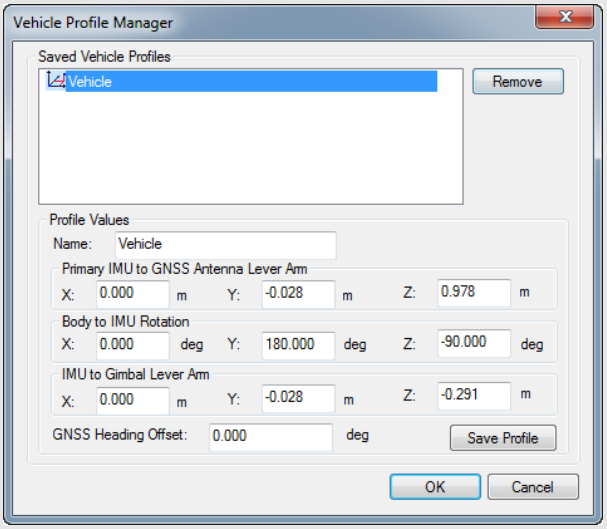
A processing profile is automatically loaded based on the detected processing environment (airborne, marine, ground vehicle) when converting the raw GNSS data to GPB format and the type of SPAN system used. If the automatically detected processing profile is incorrect, it can be changed by accessing the pull down menu.

Filter Profiles
This option, enabled by default, will only show profiles associated with the SPAN IMU in use. If using third party IMU data, disable this option in order to gain access to custom processing profiles.

Advanced...(LC Processing)
This button provides access to all IMU processing settings.

Advanced GNSS (TC Processing)
This button provides access to all GNSS processing settings. Refer to the *GrafNav/GrafNet 8.60 User Guide* for information.

Advanced IMU (TC Processing)
This button provides access to all IMU processing settings.

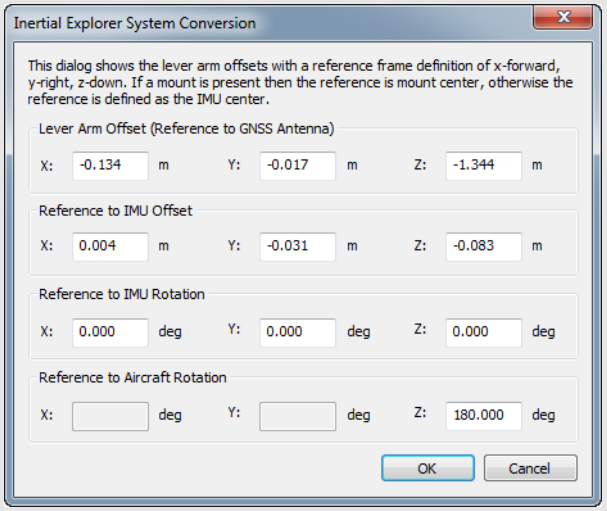


IMU Installation

Read rotations and lever arms from IMR file
If using a NovAtel SPAN system, the IMU to GNSS lever arm and vehicle body rotation may be set during data collection. If this has been set and the necessary logs (*IMUTOANTOFFSETSB*, *VEHICLEBODYROTATIONB and SETIMUORIENTATIONB*) are present in the raw data, this information is imported automatically to Inertial Explorer.

Vehicle Profile
This button accesses the *Vehicle Profile Manager*. It allows the primary lever arm, body to IMU rotation, IMU to gimbal lever arm and GNSS heading offset to be saved to a vehicle profile. This facilitates quick and easy loading of important project parameters that are specific to each survey vehicle.

It is not necessary to save vehicle profiles if using a NovAtel SPAN system as this information can be retrieved directly from the raw data and imported automatically. Vehicle profiles are intended to assist non-SPAN customer workflow.



System Conversion
Inertial Explorer uses a vehicle frame of Y-forward, X-right and Z-up. This is different from other GNSS/INS processing software packages which use X-forward, Y-right and Z-down. If you are used to working in the latter system and are transitioning to Inertial Explorer, the System Conversion tool allows you to input your project parameters within the X-forward, Y-right and Z-down frame and converts the values to the conventions used by Inertial Explorer.

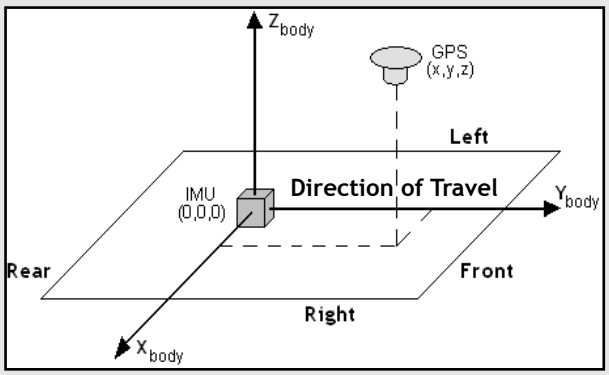


Figure 1: Body Frame Definition for Lever Arm Offset

The IMU is the local origin of the system and the measurements are defined as the following:

X: The measured lateral distance in the vehicle body frame from the IMU to the GNSS antenna.

Y: The measured distance along the longitudinal axis of the vehicle from the IMU to the GNSS antenna.

Z: The measured height change from the IMU to the GNSS antenna.

✉ All measurements are from the navigation center of the IMU to the GNSS antenna.

Lever Arm Offset (IMU to GNSS antenna)
To perform GNSS updates accurately, enter the 3-D offset, in metres, from the IMU sensor array's navigation center to the GNSS antenna. This offset vector must be entered with respect to the body-frame of the vehicle, as the image in the shaded box shows.

You must also specify whether the Z value applies to the antenna's reference point (ARP) or L1 phase center. To specify ARP, you must select an antenna model when you add the remote GPB file to the project. In this case, the antenna model's offset value is applied to the Z value to raise the Z value to the L1 phase center.

✉ Save lever arms for future access using the *Favorites* button.

Read from IMR file
If the lever arm values are written to the header of the IMR file, then use this option to extract them.

Body to IMU Rotations (Rotate Vehicle Frame into IMU Frame)
Many typical IMU installations have the surface of the IMU directly attached to the floor of the vehicle so the sensor frame of the IMU and the body frame of the vehicle are more or less aligned. In these installations, the roll, pitch and yaw of the vehicle are directly sensed by the IMU. Some IMUs are installed in a tilted position with respect to the body frame of the vehicle. If the tilt between the IMU frame and body frame is known, Inertial Explorer compensates so that the attitude information produced is with respect to vehicle body frame, not the IMU sensor frame.

The order of rotations employed is R_z, then R_x, followed by R_y, in decimal degree units.

GNSS Heading Offset
This value may also be referred to as the "Reference to Aircraft Rotation" and is meant for customers that have backwards-pointing LIDAR applications or other specialized applications where the IMU cannot be rotated to the vehicle frame through body to IMU rotations. This option applies a correction (as entered in degrees) to GNSS Course Over Ground (COG) values in order to allow kinematic alignments to succeed when the IMU is intentionally not aligned to the vehicle frame. This option has no effect if a static alignment is performed.

Advanced IMU

This button provides access to all IMU processing options.

Alignment

Method for Initial Alignment

In INS processing, small changes in velocity and orientation are integrated in order to derive position, velocity and attitude from a starting point. As such, it is a relative positioning method and the initial integration conditions must be known. Alignment is the process of solving these initial integration constants.

The initial position and velocity of the IMU are usually derived from Inertial Explorer's GNSS processor. Initial roll and pitch are derived from the accelerometer measurements and initial heading is derived from gyroscope measurements.

IMUs of tactical grade or higher are capable of static alignment. However MEMS IMUs, or any IMU with a gyro bias larger than the Earth rate (15 deg/hr at the equator), are not capable of deriving a reliable heading from gyro measurements alone. In these cases, the GNSS Course-Over-Ground (COG) must be used to help Inertial Explorer determine the initial azimuth of the IMU.

The following methods of alignment are available:

Automated Alignment (Recommended)

Automated alignment scans the raw IMU data in order to determine whether a static alignment can be attempted. If no usable static alignment is detected, a kinematic alignment is applied when the vehicle reaches the minimum speed set within the *Auto/ Kinematic Align Tolerance*. If static data is detected, a static coarse alignment is attempted for the duration of the detected session length.

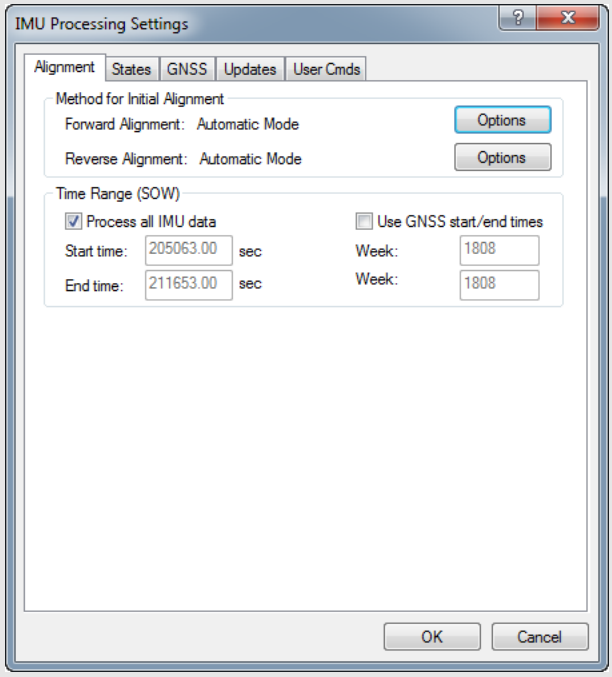
Static coarse alignment only

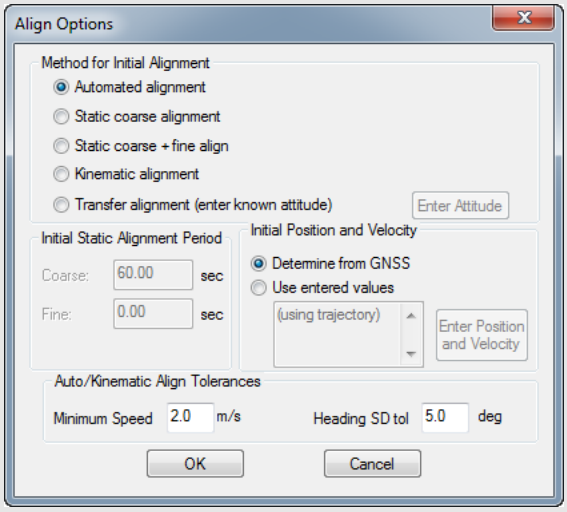
Coarse alignment uses the sensed gravity vector components to estimate roll and pitch. It uses sensed Earth-rotation rate to provide an initial estimate of the yaw of the IMU. As such, only IMUs with gyro biases much less than the Earth rate (15 deg/hr at the equator) are capable of reliable static alignment.

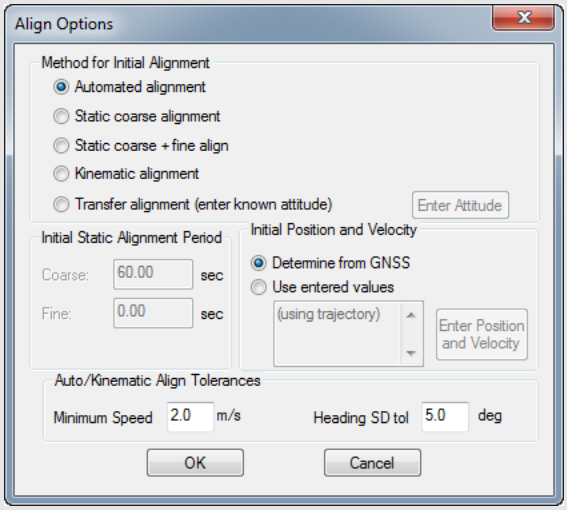
Static coarse + fine alignment

This option can be selected if using a navigation grade IMU and if collecting at least 10 minutes of static data. Typically, the first 2 minutes of data are used for coarse alignment followed by the remainder (8-10 minutes) of fine alignment.

Within 5 to 10 minutes, GNSS updates enable the IMU to provide attitude information consistent with the accuracy level achievable by the accelerometer/ gyro triad, with or without fine alignment. This depends on the type of IMU, and the application's







requirements. After roll, pitch and yaw are roughly estimated for coarse alignment, fine alignment refines them to a better level of precision.

Kinematic alignment
When no static data is detected, a kinematic alignment will be used. The GNSS Course-Over-Ground (COG) will be used as an initial approximation of the forward pointing IMU axis. As such, it is important when using a kinematic alignment that the IMU be mounted Y-forward, X-right and Z-up. If it is not possible to mount the IMU in this fashion, appropriate body to sensor rotations should be entered. If intentionally misaligning the sensor and vehicle frames, use the GNSS Heading Offset to correct the GNSS COG used in the alignment process. Kinematic alignment requires that the IMU be traveling relatively straight and relative level for at least 4 seconds.

Transfer alignment
If the initial roll, pitch and yaw values are known, these values can be entered as initial integration constants to allow navigation to proceed. Attitude angles can be provided by another IMU, in which case the misalignment between the IMUs must be applied, or they can be extracted from another trajectory, such as the opposite processing direction.

Click *Enter Attitude* to enter initial attitude information manually or select *Get from Trajectory* to scan at a specified time from a defined IMU trajectory.

Initial Static Alignment Period
The length of time assigned to static alignment depends on the method of alignment being used. In all cases, it is important that the values entered are in accordance with the *Begin* and *End* times specified under the *General* tab of the GNSS processing options menu.

To perform a static alignment, specify the length of time that the IMU was stationary. If this is unknown, the *Velocity Profile* plot obtained from the GNSS processing is useful.

✉ This field does not apply for transfer alignment or for kinematic alignment.

Initial Position and Velocity
The two options include the following:

Determine from GNSS (suggested)
This method is for collected GNSS data in addition to IMU data. The starting position and velocity is read in from the GNSS trajectory specified under the *Source of GNSS Updates* box in LC processing and in the *General* tab for TC processing.

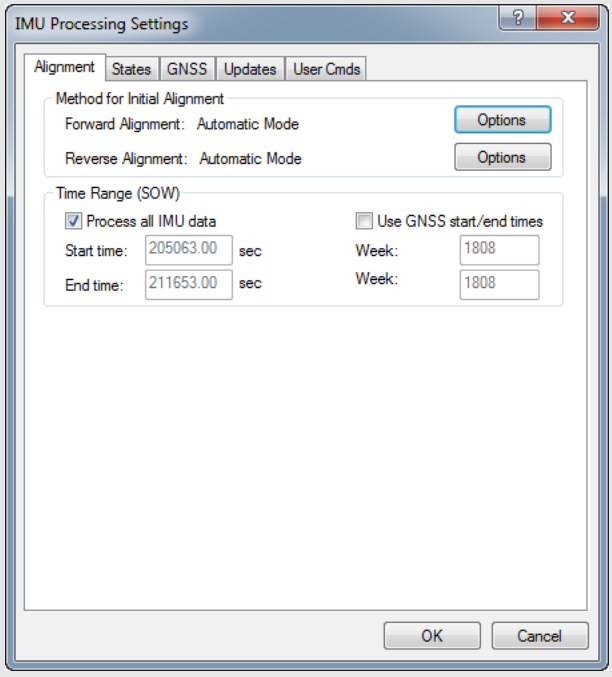
Use entered values (IMU only)
This option is for performing IMU-only processing. If GNSS data has been processed, load the position from a computed trajectory. Otherwise, enter it manually. In either case, click the *Enter Position and Velocity* button to access the input window.

Auto/Kinematic Align Tolerances

Minimum Speed
This parameter specifies the minimum speed that the system must be traveling before kinematic alignment is attempted. This should be lowered to 1 m/s for pedestrian or other very low dynamic applications.

Heading SD Tol
This parameter specifies the tolerance below which the heading standard deviation must fall before the alignment routine will move into navigation mode. Lower this value if the software is not achieving a good alignment. Raise this value if the software is not aligning at all. This value is tested both if a static alignment is attempted and if a kinematic alignment is tested.

Time Range (SOW)
These options are in the shaded box.



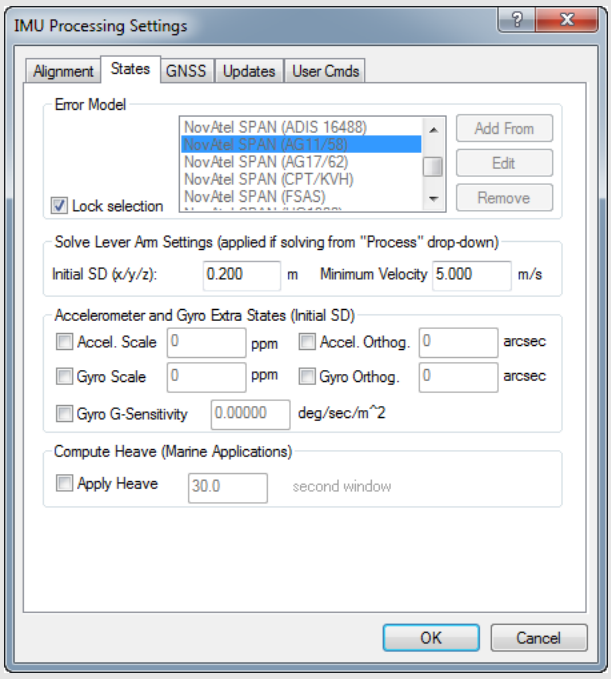
Time Range Options

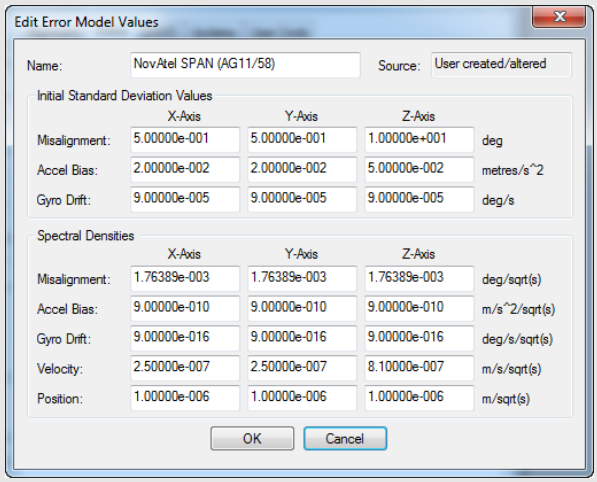
Process All IMU Data
If this option is enabled, the software obtains the beginning and end times from the raw binary IMU file. These times are in GPS seconds of the week.

Use GNSS start/end times
When selected, IMU processing will start and end based on a time range set under the *General* tab of the GNSS processing options menu.

Start Time
Forward alignment will begin at the entered time (GPS SOW). Reverse processing will end at this time.

End Time
Reverse alignment will begin at the entered time (GPS SOW). Forward processing will end at this time.





States

Error Model

Error models consist of initial standard deviation and spectral density values which are indicative of IMU sensor quality. They help control the extent to which Inertial Explorer weights GNSS and INS measurements during processing.

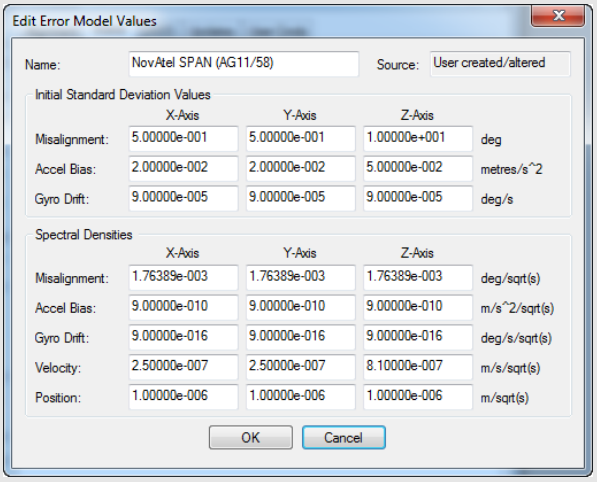
Inertial Explorer comes pre-configured with error models for all SPAN IMUs. These are automatically selected when loading an appropriate processing profile for the application (aerial, ground vehicle, marine) and SPAN system used. Inertial Explorer also comes pre-configured with a handful of error models developed for third party IMUs as well as generic error models that can be a useful starting point when attempting to develop your own new error model for a non-SPAN IMU.

Edit Error Model Values

Error model editing is only necessary when developing your own error model for a new IMU. Note that when working with MEMS sensors, it may also be necessary to enable accelerometer and gyro extra states in order to achieve a reasonable level of performance in addition to error model tuning.

Initial Standard Deviation Values

The following mathematical quantities are available:



Misalignment

These terms pertain to the difference between the computed direction cosine matrix and relate the IMU body frame to the computation frame (ECEF in Inertial Explorer) and an error-free idealized direction cosine matrix or attitude matrix.

These values represent the best estimate of the sensor's ability to compute roll, pitch and yaw during coarse alignment, assuming any is present. Roll and pitch are estimated from the sensed gravity components in the horizontal axes of the accelerometer triad, while yaw is estimated from the sensed Earth rate about the leveled gyro axes. Typically, while even MEMS sensors can make some estimation of roll and pitch, the Earth rate is often masked by noise for lower-grade IMUs. The values entered here for *x* and *y*, which represent pitch and roll, are often as much as an order of magnitude smaller than that for *z*, which represents yaw.

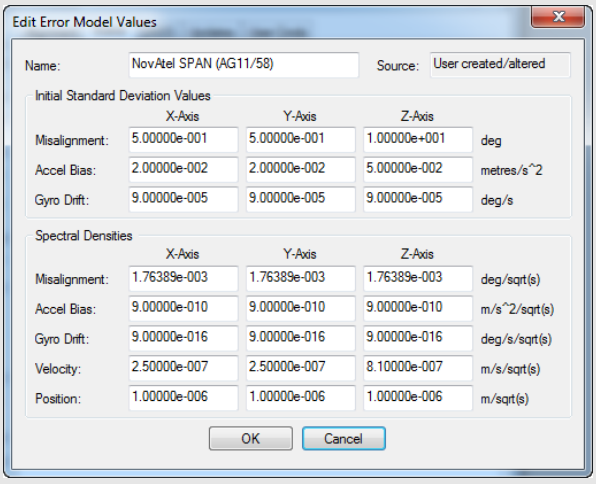
If in doubt, simply enter large values in the range of thousand of arc seconds and allow the IMU Kalman filter time to eventually compute more sensible estimates of the error in the computation of the attitude matrix. These values must be entered in degrees.

Accel Bias

These values represent the initial uncertainties in the a priori knowledge of the constant bias errors in the accelerometer triad. If these bias values were left at zero, meaning that they are unknown, then the standard deviation values entered here should reflect this uncertainty. The processor then computes the biases on-the-fly. These values should be entered in m/s^2.

Gyro Drift

These values refer to the initial uncertainty of the a priori knowledge of the sensor drift in the gyroscopes. If the biases are left at zero, then enter standard deviations values here that reflect this. The program attempts to compute reasonable values during processing. All values should be entered in degrees / sec.



Spectral Densities Values

Generally speaking, the lower the grade of the sensor, the larger the spectral densities that should be used for processing. As previously discussed, the spectral densities add noise to the covariance propagation process prior to filtering. Therefore, the higher the densities, the greater the weight that is placed on the GNSS updates during filtering. The following mathematical quantities are available:

Misalignment

A misalignment noise density, in degrees, becomes a covariance when multiplied by some time interval, δt. If the sensor triad is problematic in terms of providing an accurate attitude matrix, or if initial alignment is poor, then you may need to introduce large spectral density values here. These spectral components add noise to the computed Kalman covariances for misalignment, which, in turn, forces the processor to rely more heavily on the GNSS position and velocity updates. As a result, large errors in the direction cosine matrix are compensated for.

Accel Bias

Accelerometer bias densities, when multiplied by the prediction time interval, act as additive noise to the accelerometer bias states. As such, larger values here may help to compensate for large biases in the accelerometers.

Gyro Drift

Gyroscope drift densities act as additives to the covariances computed for the gyroscope drift states. In the case of inexpensive units, larger values here may be necessary.

Velocity

Velocity spectral densities are noise densities that account for unmodeled velocity effects during each Kalman prediction. Increasing this value permits more emphasis to be placed on the GNSS update data, but may also lead to an increase in error growth during outages. For this reason, these values should be determined as part of the tuning process. The default values are recommended unless dealing with a trajectory of unusually high dynamics, such as a race car, in which case these may need to be reduced by an order of magnitude.

Position

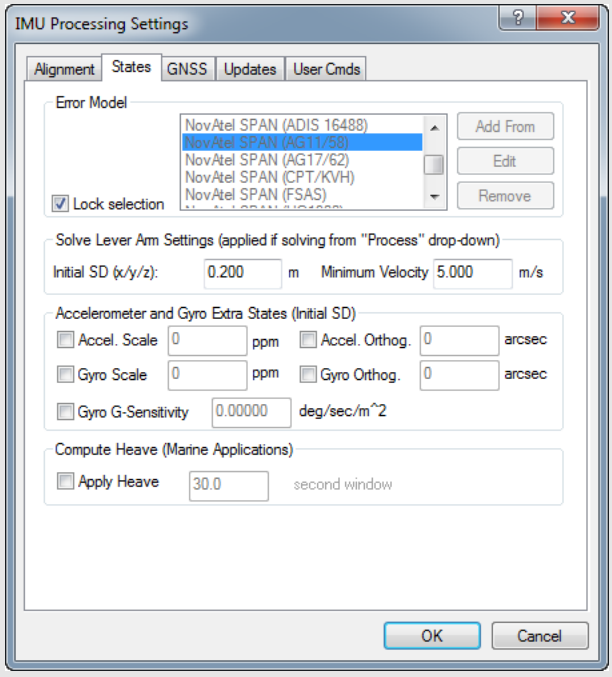
Position spectral densities are noise densities that account for unmodeled position effects during each Kalman prediction. Apply all of the considerations mentioned above for the velocity spectral densities.

Solve Lever Arm Settings
These values are applied if using the *Solve Lever Arm* feature under the *Process* pull-down menu on either the TC or LC processing dialog.

Inertial Explorer's ability to observe the IMU to GNSS lever arm is largely dependent on the length of data collection, quality of GNSS data and vehicle dynamics. This feature is not meant as a substitute for measuring the lever arm but rather for checking or troubleshooting purposes.

Initial SD
This value reflects the uncertainty in the lever arm measurement.

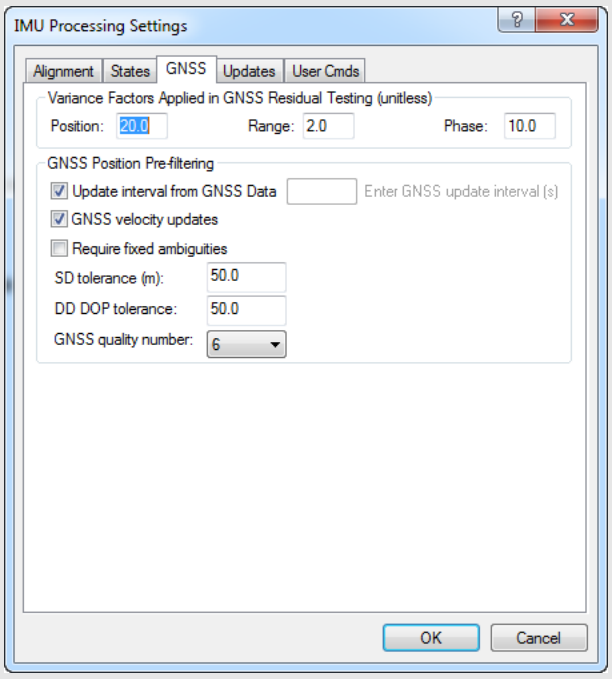
Minimum Velocity
The lever arm state will not be updated unless this minimum velocity has been reached. A minimum value of at least 1 m/s is suggested (but not required) to help avoid the possibility of the IMU to GNSS lever arm state diverging under very low dynamics.



Accelerometer and Gyro Extra States
These options add scale and/or non-orthogonality states to the Kalman filter for the accelerometer and gyroscope measurements. They are often needed by MEMS sensors to account for errors in the manufacturing process.

Compute Heave (Marine Applications)
For marine users who wish to apply heave compensation to the computed ellipsoidal height, use this option to engage Inertial Explorer's low-pass filter. The algorithm requires that a window size reflecting the period of the wave motion be entered. The smaller the window size, the more responsive the filter will be to the wave motion. If using this option, the computed ellipsoidal height and the heave compensated height can be viewed from the *Height Profile* plot. The computed heave value can also be viewed from the *Heave* plot.

After processing with *Apply Heave* enabled, you can access heave compensated ellipsoidal and orthometric heights from the Export Wizard. The *Height Profile* plot will display the processed ellipsoidal height together with the heave-compensated ellipsoidal height in order to show the effect of the heave window used. The longer the window used the smoother the marine heave compensated height will be (i.e., less responsive to changes).



GNSS

Variance Factors Applied in GNSS Residual Testing
Inertial Explorer performs residual testing using a standard least squares approach on every type of updated applied within our Kalman filter. Phase and range updates, applied when there are a minimum of two satellites, are only available in tightly-coupled processing.

Updates are accepted only if the computed residual is within the set tolerance. The larger the variance factor tolerance, the less likely an update is to be rejected by residual testing. For this reason, large values are typically applied in aerial processing profiles in order to reduce the chances of false rejection and lower thresholds are applied in ground vehicle profiles in order to lessen the likelihood that a biased update will be accepted. It is safe to use large values in clean GNSS environments where the quality of GNSS data is good, however lower values (1-3) are recommended when surveying in challenging GNSS environments.

GNSS Position Pre-filtering

Update interval from GNSS Data
This option is only available in tightly coupled processing. By default, Inertial Explorer will process using all available GNSS data. This results in position updates being available at the nominal GNSS logging rate. Use this option to limit the GNSS processing interval in tightly coupled processing. This can be useful if logging GNSS data at high rates (i.e. >1 Hz).

GNSS velocity updates
GNSS velocity updates are important, especially when a kinematic alignment is performed. As such, this option is normally engaged. However, for any special applications where GNSS velocity updates are to be rejected, they can be disabled here.

Require fixed ambiguities
If using a high precision IMU and when surveying in urban conditions with some challenging GNSS data, this option may be useful in achieving the best possible results. This option is not recommended for most systems (MEMS or tactical grade IMUs) as any GNSS updates (even one derived from float ambiguities) are generally beneficial in observing IMU sensor errors.

SD tolerance
If the position update returned by the GNSS processor is larger than this value, it will not be passed to the IMU filter. By default a large threshold is used as Inertial Explorer relies on variance factor testing to determine whether a GNSS position update should be applied or not.

DD DOP tolerance
Double Differenced DOP (DD DOP) is roughly equivalent to $PDOP^2$. GNSS position updates with large DOP values can be unreliable, however Inertial Explorer by default uses a very loose pre-filtering threshold and instead relies on GNSS variance factor testing to determine whether or not a GNSS position update should be applied. If it is desired to use a lower value, enter it here.

GNSS Quality Number
The GNSS processing engine assigns a quality number to each processed epoch between values of 1 to 6, 1 being the best. By default Inertial Explorer does not use the GNSS quality number in pre-filtering as instead it relies on GNSS variance factor testing in determining whether a GNSS position update should be accepted. If you wish to enable a lower pre-filtering tolerance, enter it here.

Updates

Automated ZUPT Detection Tolerances
These settings control the software's ability to detect periods of zero velocity.

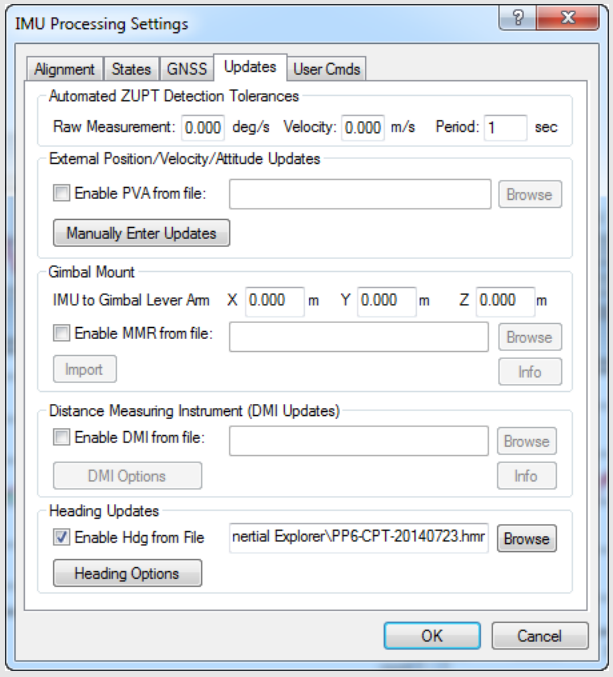
Raw Measurement
The raw gyro measurement threshold. This value may need to be raised for lower-grade sensors (i.e. MEMS) to accommodate the noisier measurements.

Velocity
The GPS velocity threshold. Potential ZUPTs are rejected if the GNSS-derived velocity exceeds this value.

Period
Length of time span over which measurements are averaged.

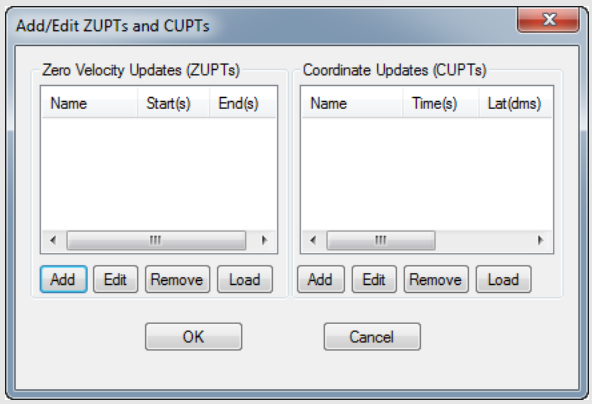
External Position/Velocity/Attitude Updates
A binary PVA file can be input to Inertial Explorer if external position, velocity and/or attitude updates are available. See *Section 3.2.4, PVA File* on page 47 for the format of this file. Input of this binary file is the recommended approach if large numbers of external updates are available. If using a PVA file, see the gray box for the supported user commands.

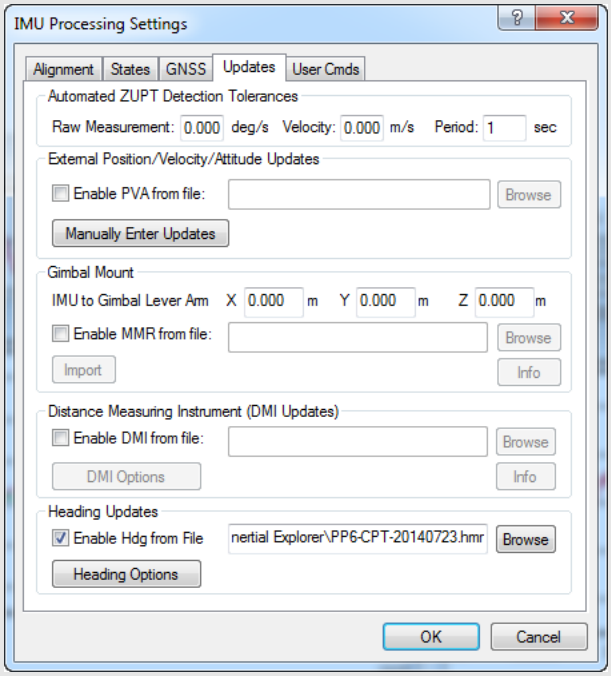
User commands can be input through the *User Cmds* tab of the *Advanced IMU* options.



User Commands Supported in PVA file

- PVA_ENABLE = “NEVER”/”ALWAYS”/ ”SPOSSD” [tol (m)] (default is “ALWAYS”)
  - NEVER = never use update ([tol] = 0)
  - ALWAYS = always use update ([tol] = 0)
  - SPOSSD = use after solution standard deviation exceeds [tol]
- PVA_FILE = “filename” [def=””]
  - filename = name of binary PVA file to use for updates
- PVA_LEVER = “ON”/”OFF” [x] [y] [z] [def=OFF]
  - Shift position of update to IMU using body-frame lever arm values. Sensor to IMU (m)
  - This command will override any offsets in the binary PVA file
- PVA_MEAS_SF = sfpos sfvel sfatt [def=1 1 1]
  - Scale input standard deviations by these scale factors
- PVA_SIGMA_TOL = ON/OFF pos vel att [def=OFF]
  - Unitless a-posteriori standard deviation ratio for PVA for blunder detection
- PVA_UPDATE = “REC/POS|VEL|ATT” [def=REC]
  - REC = use record type from record
  - POS|VEL|ATT = Space separated mask for update type in file





Manually Enter Updates

Inertial Explorer also accepts input of individual ZUPTs and CUPTs through the *Manually Enter Updates* button. This feature is more convenient for customers that want to manually enter a small number of such updates as opposed to having to format your own binary file. This feature also supports loading of ZUPTs and CUPTs from an ASCII file, however this feature is limited to supporting a maximum of 1000 updates. For this reason, using the binary PVA file is encouraged when large numbers of external updates exist.

Zero Velocity Updates (ZUPTs)

Inertial Explorer automatically detects ZUPTs by analyzing the GNSS, IMU and, if available, DMI data. This is true for both loosely and tightly coupled processing. As such, the manual entry of ZUPTs is generally not necessary, except in cases of poor data quality. Nonetheless, individual ZUPTs can be added here or loaded from an ASCII file.

Coordinate Updates (CUPTs)

External coordinate updates can be very beneficial to GNSS/INS post-processing in areas of denied GNSS signal reception if they can be properly time tagged. This dialog can be used to add individual time coordinate updates or load from an ASCII file.

Gimbal Mount

If using a gimbal mount, the IMU to gimbal center lever arm can be entered here. This should be entered in the vehicle frame (Y-forward, X-right, Z-up) with the origin at the IMU center of navigation. This causes Inertial Explorer to shift its output from the IMU to the Gimbal Center.

MMR files are automatically produced by the SPAN data converter and contain the rotations of the stabilized gimbal platform. This is required in order to properly compensate for the changing IMU to GNSS lever arm during operation of the gimbal unit.

Distance Measuring Instrument (DMI)

If logging DMI data, the NovAtel SPAN decoder will automatically write a DMR file which contains time stamped DMI measurements. If a DMR file is detected during project creation, it is automatically loaded into the project and thus does not have to be explicitly set here.

DMI Options

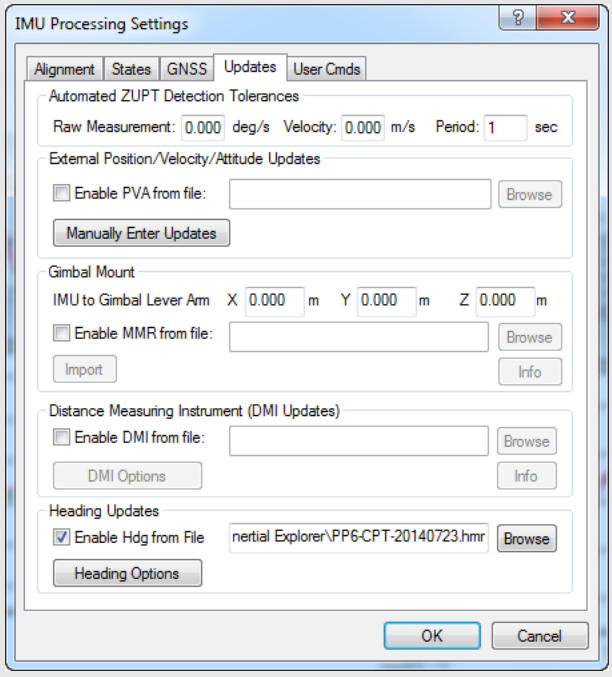
✉ A typical DMI will either output a tick count or a measured speed. If tick counts are recorded, Inertial Explorer converts them as a velocity update. If speed was recorded, then the software applies them as such.

Detect ZUPTs from DMI sensor
This option is off by default as Inertial Explorer already has two layers of ZUPT detection; analysis of the raw IMU measurements and using available GNSS data. This option therefore is generally not needed and if the DMI used does not function well at low velocity, it can actually be harmful.

If however a high resolution DMI is used which works well at low velocity and if ZUPTs will be observed during periods of extended GNSS outages, this option can be very beneficial in helping to observe ZUPTs.

Measurement standard deviations
The standard deviation associated with the DMI measurements depends on the DMI being used. As such, this value may need to be determined empirically.

Wheel circumference
The wheel circumference is extracted from raw SPAN data if the WHEELSIZEB log is present. The default value is 1.96 metres if no value is detected from the raw GNSS data or set during conversion. Inertial Explorer computes a DMI scale factor to account for varying wheel sizes during data collection, however the best estimate possible of the wheel circumference should be input.

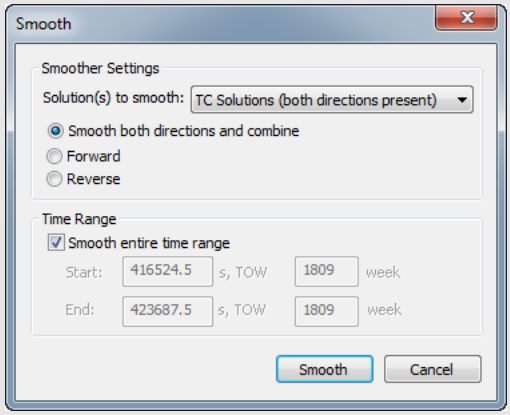


Heading Updates

If using the NovAtel dual antenna ALIGN system and requesting HEADINGB logs, an HMR file is automatically produced during data conversion which can be input to Inertial Explorer here. The secondary lever arm must also be set if using it in *Dual GPS Antenna System* mode.

Heading updates are most useful in assisting auto/kinematic alignment in low dynamic applications such as marine surveys. When a kinematic alignment is used and heading updates are available, Inertial Explorer will extract the initial heading of the vehicle from the HMR file.

The HMR data format is described in *Section 3.2.3, HMR File* on page 46.



1.5.2 Combine Solutions

Refer to the *GrafNav/GrafNet 8.60 User Guide* for information regarding these options. Only points relevant exclusively to Inertial Explorer are made here.

Smooth Solutions

Inertial Explorer is capable of combining processing directions and/or back-smoothing on an inertial trajectory.

Smoothing greatly improves position quality over GNSS data gaps. The benefits of smoothing are not limited to improving position quality however; velocity and attitude are also back-smoothed. Even if no GNSS data gaps are observed, smoothing will always generate the highest quality trajectory possible. Running the smoother automatically creates all the high rate binary files required by the Export Wizard.

Smoother Settings

Smoothing can be performed in just one direction, or both. Much like GNSS and GNSS-IMU processing, it is recommended that smoothing be performed in both directions.

Solution(s) to smooth

This option performs smoothing on the loosely coupled or tightly coupled trajectory.

Time Range

This setting controls the period of time for which to perform the combining and/or smoothing of the trajectories. Epochs outside of this time range are not considered and do not appear in the output files.

1.5.3 Solve Boresighting Angles

Show
This drop-down menu is linked to the window below it and gives viewing access to the values listed in the shaded box.

Settings
The following features are available:

Calibration name
Enter a name to distinguish calibration runs from one another. Inertial Explorer keeps a history of calibration runs, so a unique identifier is helpful when trying to recover previous results. This is useful for using multiple systems and/or tracking stability over time.

Boresight Angles
Upon successful completion of the calibration procedure, the final values for the computed boresight angles are displayed here.

Add results to list
When this option is enabled, the last values computed by the program are stored so that they are easily accessible by the *Export Wizard*.

View report after computation
Enabling this option forces the software to launch the boresighting report upon successful completion of a calibration. The contents of the report are discussed later on.

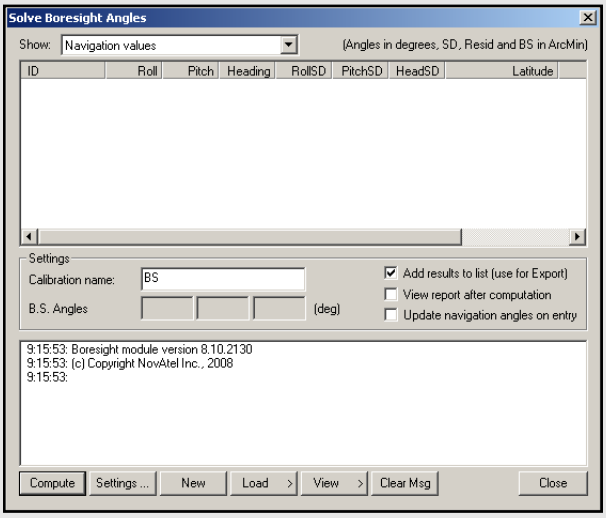
Update navigation angles on entry
When this option is enabled, Inertial Explorer loads the latest navigation values for the camera events into the boresighting module.

Message Window

This window provides valuable insight on the status of the current calibration. Whenever input data is being loaded, read the messages to ensure the expected number of camera events have been read in. After the calibration procedure is complete, the final boresighting values, as well as the number of iterations needed to arrive at them, are displayed.

The following options are available via the buttons along the bottom of the *Solve Boresight Angles* window:

Compute
Assuming all the required input data has been loaded, click this button to begin the iterative least squares procedure. The *Message Window* contains pertinent information regarding the success or failure of the procedure.

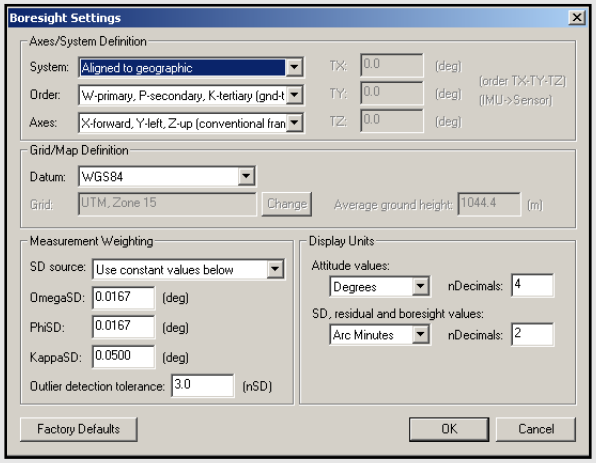


Values that are visible with the *Show* drop down menu:

Navigation values
The roll, pitch and heading values, along with their associated standard deviations, are displayed for each loaded camera event. The coordinates of the IMU at the time of the event are also displayed. These values are generally transferred from Inertial Explorer directly and correspond to the IMU values interpolated at camera event times.

Photo E/O values
The omega, phi and kappa values, along with their associated standard deviations, are displayed for each loaded camera event. These values are produced externally in a photogrammetric package.

Matches/residuals
Before the computations begin, choose whether or not to include the observations associated with a camera event in the least squares procedure by right-clicking on the event. After the least squares procedure has finished, the window is updated with the final residual values at each camera event. Additional information, such as quality indicators and computed omega, phi and kappa values are also displayed.

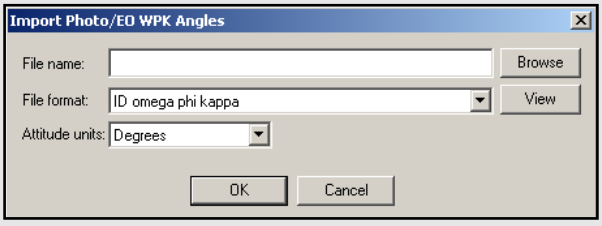


Axes/System Definition options

System
The selection made here defines the ground coordinate system to which the omega, phi and kappa values are oriented. Normally, they are referenced to a map projection which is defined in the *Grid/Map Definition* settings.

Order
This setting defines the order in which the omega, phi and kappa angles are to be applied during the transformation from the ground system to the image or IMU system. Only the omega-primary, phi-secondary and kappa-tertiary rotation order is supported.

Axes
Use this setting to define the orientation of the image system. The most commonly used system is the conventional frame, where the x-axis points forward, the y-axis points left, and the z-axis points upwards. The frame defined here determines the composition of the R_c matrix.



Settings…
This button gives access to the *Boresight Settings* window, which is useful for configuring many parameters used in the boresight calibration.

Axes/System Definition
These options are listed in the shaded box.

Grid/Map Definition
The options available here depend on the system definition chosen. If the input angle was provided with respect to a map grid, then the selection made here determines the convergence value, α, used to form the R_g matrix. In addition, grid users are given the opportunity to enter the average ground height in order to maximize accuracy.

Measurement Weighting
The selections made here determine the composition of the variance-covariance matrix used in the least squares procedure to derive the final boresighting values. Choose to enter a set of constant standard deviation values to apply to all measurements, or have the values derived from either the navigation SD values, the photo SD values (if provided), or a combination of both.

The other setting here pertains to the outlier tolerance. The value specified determines at which point a measurement is removed from the least squares procedure.

Display Units
These options pertain to the values displayed in the *Solve Boresight Angle* window and determine which units are used when writing to the *Boresight Report* file. These options also allow the number of decimal places to which all values are displayed or written to be modified.

New
This button clears any stored data from previous calibration runs in order to start a new one.

Load
Use this button to load the required navigation and exterior orientation input data.

The navigation data can be obtained either by loading the latest set of roll, pitch and heading values computed by Inertial Explorer, or by an external file which contains this information for each camera event. Alternatively, if such information is available, there is the ability to provide the module directly with the omega, phi and kappa angles required to rotate the ground system into the IMU frame. Obtaining the attitude angles directly from Inertial Explorer is by far the most common usage.

The exterior orientation parameters for each photo must be supplied by an external file. This file should contain the omega, phi and kappa angles required to rotate the ground system into the image system.

View
This button gives access to the post-calibration report. The report contains relevant boresight calibration information, as well as a list of all the input data provided for each camera event. The bottom of the report displays the boresight values and residuals from the final iteration.

✉ This report can be viewed through either NotePad or the internal Inertial Explorer ASCII viewer. This button also gives you access to the calibration history. For each calibration run, the final boresighting results are saved, assuming the *Add results to list* option is enabled.

Clear Msg
This button clears the *Message Window* of any messages currently displayed.

1.6 Settings Menu

Refer to the *GrafNav/GrafNet 8.60 User Guide* for information regarding all of the features available from this menu.

1.7 Output Menu

Refer to the *GrafNav/GrafNet 8.60 User Guide* for information regarding all of the features available from this menu. Only those features exclusive to Inertial Explorer are discussed here.

1.7.1 Plot Results

Refer to the *GrafNav/GrafNet 8.60 User Guide* for information regarding all of the GNSS plots available.

By default, the software generates all plots at the GPS update interval. You can raise the interval as high as the IMU data rate to get a denser plot, but generation takes longer. This setting is available under the X-axis tab. This option requires that you have generated a combined binary file for your trajectory (before or after smoothing) (see *Section 1.5.3, Solve Boresighting Angles* on page 31). The combined file must be re-generated after every processing run to ensure that the plot reflects the latest results.

Table 1, IMU Plots, in the shaded box on this page and the following pages, contains descriptions of the IMU plots available only through Inertial Explorer.

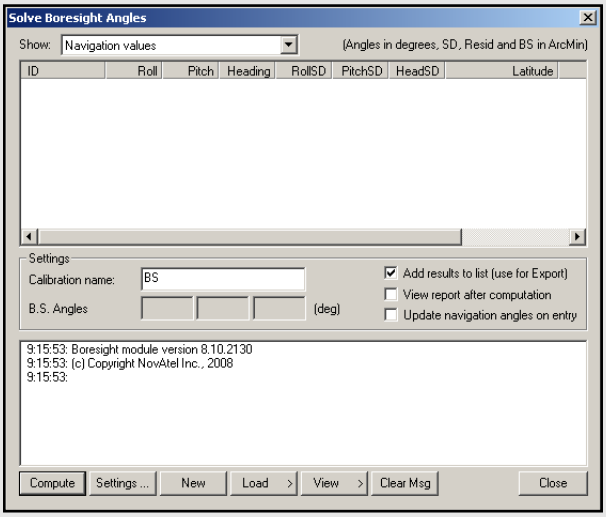


Table 1: IMU Plots

Plot	Description
Accelerometer Bias	This is the apparent output in acceleration when there is no input acceleration present. It is computed by the GNSS/INS Kalman filter and the effects may be sinusoidal or random. It is plotted in terms of the X (right direction), Y (forward direction), and Z (up direction) of the INS body. Generally, they should stabilize after the alignment period and agree when processed in both directions.
Attitude (Azimuth/ Heading)	Plots the heading and GNSS COG (course-over-ground) that was computed from the GNSS/INS processing. The effects of crabbing are visible in this plot if the GNSS COG bears a constant offset from INS heading. The *IMU Heading COG Difference* plot shows the difference between these two heading values. Note that any transitions between a heading of 359 degrees and 0 degrees shows up as a vertical line.
Attitude (Roll and Pitch)	Plots the roll and pitch values from GNSS/INS processing. In airborne data, it is common to see roll values between 30 degrees and pitch values of around 10 degrees, depending on the flight pattern of the aircraft itself.
Attitude Separation	This plot shows the difference between the forward and reverse solutions in terms of roll, pitch and heading. A zero separation is ideal, as it indicates matching solutions in the forward and reverse IMU processing. Spikes at the beginning and the end of the plot are common, as they indicate the periods of alignment.
Body Frame Acceleration	This plot shows the components of acceleration in the vehicle body frame.
Body Frame Velocity	This plot shows the components of velocity in the vehicle body frame.

(Sheet 1 of 3)

Table 1: IMU Plots (continued)

Plot	Description
DMI Scale Factor	This plot presents the DMI scale factor, as computed by the Kalman filter. It should be loaded separately for forward and reverse processing to ensure that the same scale factor is computed in both directions. Ideally, the plotted line should be horizontal, indicating a constant scale factor.
DMI Residual	This plot presents the difference between the computed displacement or velocity and that reported by the DMI.
DMI Analysis Tool	This tool allows DMI users to view the raw data measurements found in their DMR file. They can use the options available here to find an appropriate scale factor that will make the DMI data fit best with the values computed from the GNSS-IMU data.
Estimated Accelerometer Bias Accuracy	This plot shows the estimated standard deviation of the accelerometer bias. It is plotted in terms of the X (right direction), Y (forward direction), and Z (up direction) of the INS body.
Estimated Attitude Accuracy	This plot shows the standard deviation computed in the GNSS/INS Kalman filter in terms of roll, pitch and heading.
Estimated Gyro Drift Accuracy	This plot shows the estimated standard deviation of the gyro drift rate, which generally decreases with time. It is plotted in terms of the X (right direction), Y (forward direction), and Z (up direction) of the INS body.
Gimbal Data Values	Inertial Explorer supports data processing from IMUs mounted on a gyro-stabilized platform (gimbal). When doing so, it is required to import an MMR file in order to compensate for the changing lever arm between the IMU and GPS when the gimbal is engaged. This plot shows the decoded rotations within the MMR file and can thus be useful in troubleshooting if the values in the MMR file are suspect.
Gyro Drift Rate	This is the apparent change in angular rate over a period of time, as computed by the GNSS/INS Kalman filter. The effects are usually random. It is plotted in terms of the X (right direction), Y (forward direction), and Z (up direction) of the INS body. Generally, they should stabilize after the alignment period and agree when processed in both directions.
Gyro Attitude Misclosure	This plots shows the misclosure (residual) of gyroscope Kalman filter updates. Large values here could be an indication of attitude instability.
IMU Angular Rates	This plot shows the gyroscope rate of change of attitude in the X, Y and Z axes of the IMU body with the drift removed. This plot is used to check the gyros.
IMU Status Flag	Shows the status of IMU processing. Specifically, this plot provides indication of the type of update, if any, being applied at each epoch.

(Sheet 2 of 3)

1.7.2 Export Wizard

Only the *Export Wizard* window exclusive to Inertial Explorer is discussed here. Refer to the *GrafNav/GrafNet 8.60 User Guide* for additional information concerning this feature.

IMU Epoch Settings

Epoch/Output Interval
The *Kalman/GPS interval* indicates the interval used during GPS processing, while the *IMU integration interval* displays the interval at which the IMU data was processed. These values can only be changed prior to processing. The *Output data interval* defines the interval to export solutions. The output interval can be set as high as 1000 Hz, regardless of what interval the data was processed at.

✉ The time range for which to obtain outputs can also be limited here.

Lever Arm/Offset
Allows for the coordinates of the IMU, calculated via the IMU Kalman filter, to be transferred to an alternate sensor's location.

Note the orientation of the frame in which these coordinates must be entered. You are also free to save your offset for future use via the *Favorites* button.

1.7.3 Export to RIEGL POF/POQ

Selecting this option launches the *Export RIEGL* dialog which supports the conversion of Inertial Explorer's native binary output (SBTC or SBIC file for tightly and loosely coupled processing respectively) to RIEGL POF (Position & Orientation file) format. The POF file format is supported by third party software manufacturers and as such is a convenient way to integrate Inertial Explorer into a larger workflow.

1.7.4 Export to SBET

Selecting this option launches the *Export SBET* dialog which supports the conversion of Inertial Explorer's native binary output (SBTC or SBIC for tightly and loosely coupled processing respectively) to Applanix SBET binary format. The Applanix SBET file format is supported by many third party software packages and thus this utility is a convenient way to integrate Inertial Explorer into a larger workflow.

1.8 Tools Menu

Refer to the *GrafNav/GrafNet 8.60 User Guide* for information regarding all of the options available via this menu.

1.9 Interactive Windows

Refer to the *GrafNav/GrafNet 8.60 User Guide* for information regarding the *Map Window* and the features available within it.

1.10 Processing Window

Table 1, IMU Plots on page 33, contains a list of the additional parameters available for viewing in Inertial Explorer during processing. Display these values via the *View* button in the *Processing Window*.

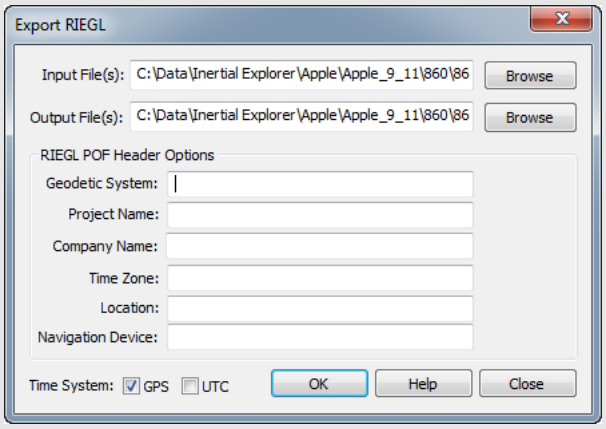
The values in the *GrafNav/GrafNet 8.60 User Guide* differ in the manner in which they are computed depending on the mode of processing being performed.

If the GNSS is being processed, then the values displayed are those computed in the Kalman filter. However, during the IMU processing, the values displayed reflect those calculated in the IMU Kalman filter, using the GNSS information as updates. Ideally, these values should agree. When they do not, monitor the position and velocity misclosure.

Table 1: IMU Plots (continued)

Plot	Description
IMU-GPS Lever Arm	This plots presents the body-frame components of the lever arm offset between the IMU and GNSS antenna. If the offset was manually entered, then this plot has constant horizontal lines. If left to be solved by the Kalman filter, this plot shows the computed values.
IMU Heading COG difference	This plot is the difference between the IMU heading and the GNSS course-over-ground values. Effects of crabbing shows up as a direct bias in this plot.
Velocity Separation	Plots the difference between the East, North and Up components of velocity computed during forward and reverse processing. Requires that both directions be processed and combined.
IMU-GPS Position Misclosure	This plot shows the difference between the GNSS solution and the mechanized INS positions obtained from the GNSS/INS processing. This is a good analysis tool used to check the GNSS/INS solution as well as checking INS stability. Large jumps or spikes may indicate a bad INS solution, whereas separations nearing zero confirms the GPS solution.
IMU-GPS Velocity Misclosure	This plot shows the difference between the GNSS calculated velocity and the mechanized INS velocity obtained from the GNSS/INS processing. Another good analysis tool used to check INS stability.
Raw IMU Data Values	Use this plot to see the raw gyroscope and accelerometer measurements as they appear in the IMR file.

(Sheet 3 of 3)



1.11 Help Menu

1.11.1 Help Topics

Opens an HTML version of this manual, with the GrafNav portion included. This feature is a quick and easily accessible reference.

1.12 About Inertial Explorer

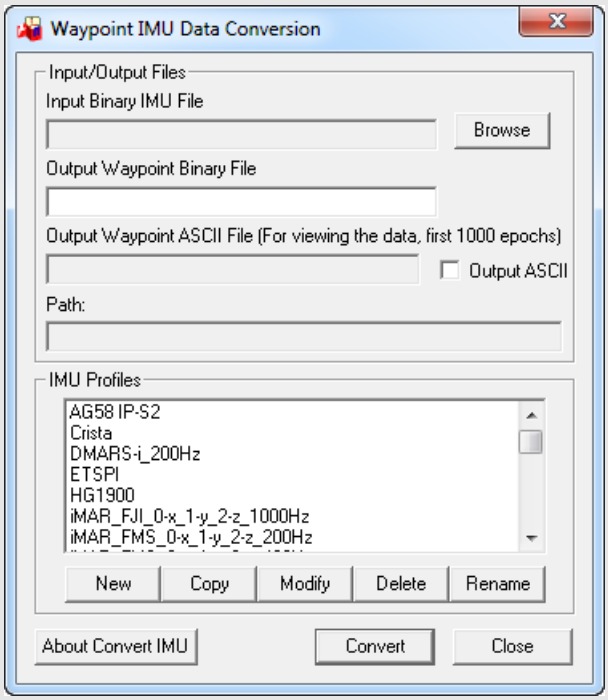
This window displays information about the software version, build dates, copyright information, hardware lock key information and DLL information.

Access the hardware key utility from this window by clicking *Key Util (Upgrade)*. This tool is useful for upgrades if using USB licensing. The *Dependent Files* window displays a list of executables and DLLs associated with Inertial Explorer. The date and time of the files are shown, as well as a quick description of the file.

Chapter 2 Conversion Utilities

2.1 Raw IMU Data Converter

The *IMU Data Converter* utility is a Win32 application program that converts custom data formats into a generic raw IMU data format. This utility is available exclusively to users of Inertial Explorer and may be accessed from *File | Convert | Raw IMU Data to Waypoint Generic (IMR)*.



2.1.1 Waypoint IMU Data Conversion

Input/Output Files
Refers to the names and locations of all input and output files.

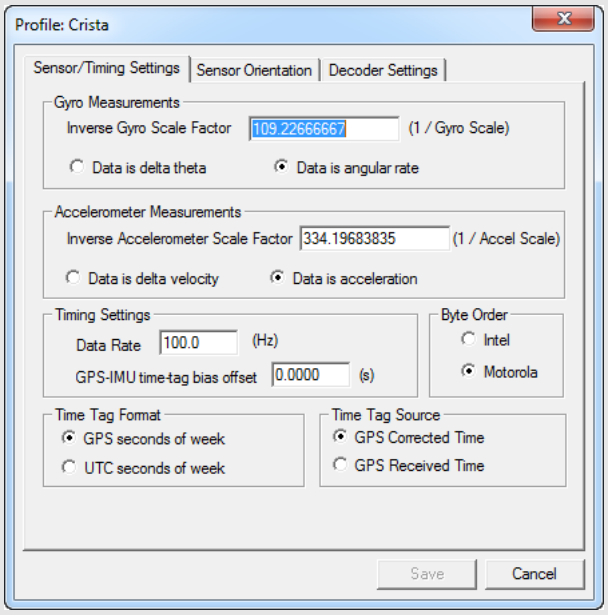
Input Binary IMU File
Click the *Browse* button to locate the raw IMU data file.

Output Waypoint Binary File
By default, the binary output file created is given the same filename as the input file, but with an IMR extension. It is saved to the directory containing the input file.

Output Waypoint ASCII File
If the *Output ASCII* option is enabled, the utility generates an ASCII file containing the GPS time, as well as the gyroscope and acceleration measurements of all three axes for the first thousand epochs. Use this to detect any errors that may occur during the conversion, such as the use of an incorrect scale factor.

Path
Displays the path to the directory containing the input file. All output files created by this utility are saved to this directory.

IMU Profiles
Displays a scroll-down list of profiles available for use during conversion. Each profile contains a set of conversion parameters designed to decode measurement data files produced by the indicated sensor. Choose one profile from the list, or, if necessary, create one. See *Section 2.1.2, Creating / Modifying a Conversion Profile* on page 38 for help. After all the appropriate fields have been entered, click the *Convert* button to start converting IMU data into IMR format. A message window appears to show the status of the conversion process.



2.1.2 Creating / Modifying a Conversion Profile

New
Creates a customized profile to convert a unique format into Waypoint's generic IMR format. This is used for custom scale factors, data rates, and orientations in raw data files.

Modify
Allows changes to be made to an existing profile.

Delete
Deletes an existing profile.

Rename
Renames an existing profile.

Sensor/Timing Settings

Gyro Measurements
Pertains to the measurements made by the gyroscopes.

✉ The inverse value of the scale factor is required. For example, a scale factor of 0.0004, which can be represented fractionally by 1/2500, should be entered as 2500.

The gyro measurements can take the form of *delta theta*, where angular increments are being observed, or *angular rate.*

Accelerometers Measurements
Similar to the scale factor of the gyro measurements, the inverse of the accelerometer scale factor is required. As well, the accelerometer measurements can take two forms, the first being *data velocity*, or *velocity increments*, and the other being *acceleration*.

Timing Settings
Enter the data collection rate of the IMU sensor and specify any offset that may exist between the GNSS and the IMU time tags.

Byte Order
This flag must be properly set.

✉ If the proper byte order is not specified, the decoding of the binary raw file will fail.

Time Tag Format
There are two options available as the data is either acquired in the GPS time frame or the UTC time frame. This must be correctly identified in order for the IMU data to be properly aligned with the GNSS data.

Time Tag Source
Specify whether the time tags represent the *GPS Corrected Time* or the *GPS Received Time*.

2.1.3 Sensor Orientation Settings

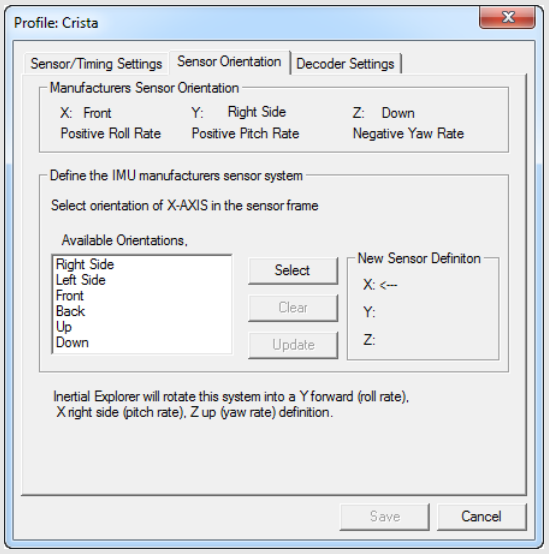
Define the orientation of the IMU here using the steps in the shaded box.

✉ The orientation will always be right-handed.

2.1.4 Decoder Settings

Specifies which library is used to perform the conversion, based on the input format of the raw data file. For most sensors, this should be left untouched.

✉ For SPAN, the IMU decoding is handled through the GNSS decoder.



How to define the orientation of the IMU:

1. Specify the X-direction by selecting the direction that corresponds to the X-axis of the sensor frame.
2. Click *Select* to set that direction to the X-axis.
3. Specify the Y-direction by selecting the direction that corresponds to the Y-axis of the sensor frame.
4. Click *Select* to set that direction to the Y-axis.

✉ Given the constraint that the frame is right-handed, this direction will be automatically determined by the software.

5. Click *Update* to apply the new sensor orientation to the profile.

✉ If a mistake is made at any point during the process, click *Clear* to start over.

6. Click *Save* to save the new profile.

✉ It should immediately appear in the scroll-down list under the IMU *Profiles* box of the main window.

Chapter 3 Data and File Formats

3.1 Data Formats

In theory, virtually any IMU sensor can be used with Inertial Explorer. The only requirement is that the data be logged in the format provided in this section, which allows easy decoding with the *IMU Data Conversion* utility described in *Section 2.1.1, Waypoint IMU Data Conversion* on page 37.

Table 2, Binary Structure of Raw Data presents the binary structure in which the conversion utility expects the raw IMU data to be logged.

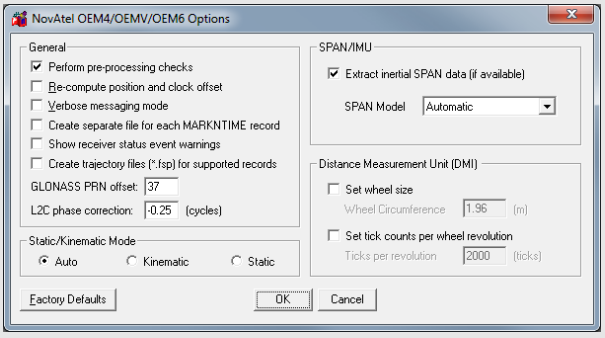
3.1.1 NovAtel's SPAN Technology

With the use of NovAtel's SPAN technology, note the difference in the IMU data decoding procedure. Since the raw IMU data measurements are embedded into the same binary file containing the raw GNSS measurements, only one step is needed to separate the data and convert it into the Waypoint Group's format. Therefore, the *Raw IMU Data Converter* utility does not need to be used.

Instead, decode the GNSS and IMU data simultaneously via the *Convert Raw GNSS data to GPB* utility, which can be accessed from *File | Convert*. When adding the measurement file to the *Convert Files* window for decoding, ensure that the drop-down menu under the *Receiver Type* box has been set to *NovAtel OEM4/OEMV/OEM6*. Then, click either the *Global Options* or *Options* button to gain access to the IMU decoding settings.

Table 2: Binary Structure of Raw Data

Word	Size (bytes)	Type	Description
GpsTime	8	real	time of the current IMU rate measurements in GPS seconds of the week
GyroX	4	long	scaled X-body axis gyro measurement as an angular increment or angular rate
GyroY	4	long	scaled Y-body axis gyro measurement as an angular increment or angular rate
GyroZ	4	long	scaled Z-body axis gyro measurement as an angular increment or angular rate
AccelX	4	long	scaled X-body axis accelerometer measurement as a velocity increment or acceleration
AccelY	4	long	scaled Y-body axis accelerometer measurement as a velocity increment or acceleration
AccelZ	4	long	scaled Z-body axis accelerometer measurement as a velocity increment or acceleration



3.2 File Formats

3.2.1 IMR File

Waypoint converts all custom IMU raw binary formats into a generic format (IMR), which is read from Inertial Explorer following the decoding process in *IMU Data Converter*. See *Chapter 2, Conversion Utilities* on page 37 for more details.

Because it contains vital information for reading and decoding the data, the first 512 bytes of the generic IMU data format is a header which must be filled in, read and interpreted. In a C/C++ structure definition, the generic format header has the following fields:

```
struct imr_header_type
{
   char   szHeader[8];              // $IMURAW[\0] - NULL terminated ASCII string

   char   bIsIntelOrMotorola;       // 0 – Intel (Little Endian) - default
                                    // 1 – Motorola (Big Endian) - swap bytes for IExplorer
                                    // This can be set for any user who directly writes in our
                                    // format with a Big Endian processor. IExplorer will swap the bytes

   double dVersionNumber;           // Program version number (i.e. 8.60)

   int    bDeltaTheta;              // Default is 1, which indicates the data to follow will be delta
                                    // thetas, meaning angular increments (i.e. scale and divide by
                                    // by dDataRateHz to get degrees/second). If the flag is set to 0, then
                                    // the data will be read directly as scaled angular rates

   int    bDeltaVelocity;           // Default is 1, which indicates the data to follow will be delta v's,
                                    // meaning velocity increments (i.e. scale and divide by
                                    // dDataRateHz to get m/s^2). If the flag is set to 0, then the data will
                                    // be read directly as scaled accelerations

   double dDataRateHz;              // i.e. 100.0 records/second. If you do not know it, set this to zero
                                    // and then fill it in from the interface dialog boxes

   double dGyroScaleFactor;         // Scale (multiply) the gyro measurements by this to get degrees/sec,
                                    // if bDeltaTheta=0. Scale the gyros by this to get degrees, if
                                    // bDeltaTheta =1. If you do not know it, then the data can not be
                                    // processed. Our default is to store the gyro data in 0.01 arcsec
                                    // increments or 0.01 arcsec/sec, so that GYRO_SCALE = 360000

   double dAccelScaleFactor;        // Scale (multiply) the accel measurements by this to get m/s^2
                                    // if bDeltaVelocity=0. Scale the accels by this to get m/s, if
                                    // bDeltaVelocity =1. If you do not know it, the data can not be
                                    // processed. Our default is to store the accel data in 1e-6 m/s
                                    // increments or 1e-6 m/s^2, so that ACCEL_SCALE = 1000000

   int    iUtcOrGpsTime;            // Defines the time-tags as being in UTC or GPS seconds of the week
                                    // 0 – Unknown (default is GPS), 1 – UTC, 2 – GPS

   int    iRcvTimeOrCorrTime;       // Defines whether the GPS time-tags are on the nominal top of the
                                    // second or are corrected for receiver time bias
                                    // 0 – do not know (default is corrected time)
                                    // 1 – receive time on the nominal top of the epoch
                                    // 2 – corrected time i.e. corr_time = rcv_time - rcvr_clock_bias
```

```
    double dTimeTagBias;          // default is 0.0, but if you have a known millisecond-level bias in
                                  // your GPS→INS time tags, then enter it here

    char szImuName[32];           // Name or type of inertial unit that is being used

    unsigned char reserved1[4];   // Reserved for future use; bytes should be zeroed

    char szProgramName[32];       // Name of calling program; skip if writing directly to this format

    time_type tCreate;            // Creation time; skip if writing directly to this format (12 bytes)

    bool bLeverArmValid;          // Set to true if the sensor definition that follows is valid
                                  // Lever arm is from IMU to GPS phase center

    long lXoffset;                // X value of lever arm, in millimetres
    long lYoffset;                // Y value of lever arm, in millimetres
    long lZoffset;                // Z value of lever arm, in millimetres

    char Reserved[354];           // Reserved for future use; bytes should be zeroed
};
```

The single header, which is a total of 512 bytes long, is followed by a structure of the following type for each IMU measurement epoch:

```
typedef struct
{
    double Time;                  // GPS time frame – seconds of the week
    long gx,gy,gz;                // delta theta or angular rate depending on flag in the header
    long ax,ay,az;                // delta v or acceleration depending on flag in the header

} INS_type;                       // this is the binary structure type expected in GPSIMU
```

✉ The angular increments (or angular rates) are scaled long integers. The scale factor to obtain a double precision word must be supplied in the header. Similarly, the accelerations (or velocity increments) are signed four byte words and must be scaled by a double precision variable given in the header.

3.2.2 DMR File

✉ All odometer data must be written into Waypoint's generic format (DMR) before it can be used within Inertial Explorer.

```
struct dmi_hdr_type
{
   char szHdr[8];                  // $DMIRAW[\0] - NULL terminated ASCII string

   short sHdrSize;                 // Size of header, in bytes. Must be set to 512

   short sRecSize;                 // Size of each record (refer to dmi_lrec_type and dmi_drec_type)
                                   // 12 + 8*sDim if sValueType = DMI_VALUE_DOUBLE
                                   // 12 + 4*sDim if sValueType = DMI_VALUE_LONG
                                   // where sDim is number of DMI sensors

   short sValueType;               // Number type (DMI_VALUE_LONG/DOUBLE)
                                   // 0 if logging data using LONG values
                                   // 1 if logging data using DOUBLE precision

   short sMeasType;                // Measurement type (distance or speed)
                                   // 1 if logging a distance measurement
                                   // 2 if logging a speed measurement

   short sDim;                     // Number of DMI sensors
                                   // Maximum is 3, but only 1 can be used in Inertial Explorer

   short sRes;                     // Measurement resolution of DMI
                                   // 1 if low resolution (i.e. only makes measurements on the full wheel revolution)
                                   // 3 if high resolution (i.e. makes measurements at partials of a wheel revolution
                                   // or on fixed time intervals)

   short sDistanceType;            // Type of distance measurement
                                   // Must be set if sMeasType = 1 (distance measurements)
                                   // 1 if logging accumulated tick count
                                   // 2 if logging distance, in metres
                                   // 3 if logging accumulated distance, in metres

   short sVelocityType;            // Type of velocity measurement
                                   // Must be set if sMeasType = 2 (velocity measurements)
                                   // 1 if logging velocity in metres/second
                                   // 2 if logging velocity in ticks/second

   double dScale;                  // Scale factor (m/count or m/s/count)
                                   // Must be set if sValueType is set to 0
                                   // 1.0 if logging accumulated tick count or ticks/seconds
                                   // If logging in metres or metres/second, then dScale will
                                   // scale measurements into corresponding units

   char szAxisName[DMI_MAX_DIM][16];
                                   // Name of various axes/DMI; optional; NULL terminated

   double dWheelSize;              // Size of the wheel, circumference in metres
                                   // Must be set if logging accumulated tick count or ticks/second
```

```
    long lTicksPerRevolution;       // Number of tick counts per wheel revolution
                                    // Must be set if logging accumulated tick count or ticks/second

    char cExtra2[420];              // Reserved for future use; bytes should be zeroed
};
```

The single header, which is a total of 512 bytes, is followed by one of the following structure types for each DMI measurement record:

```
struct dmi_lrec_type                // If logging using LONG values
{
    short sSync;                    // Sync byte
                                    // Set to 0xffee

    short sWeek;                    // GPS week number; set to -1 if not known

    double dTime;                   // GPS time of week, in seconds

    unsigned long lValue[DMI_NUM_DIM];
                                    // values (counts)
                                    // DMI_NUM_DIM should be equal to sDim
};

struct dmi_drec_type                // If logging using DOUBLE precision
{
    short sSync;                    // Sync byte
                                    // Set to 0xffee

    short sWeek;                    // GPS week number; set to -1 if not known

    double dTime;                   // GPS time of week, in seconds

    double dValue[DMI_NUM_DIM]; // values (double precision)
                                    // DMI_NUM_DIM should be equal to sDim
};
```

3.2.3 HMR File

The 256 byte header contains information that is vital to processing and must be filled in. The C/C++ structure definition of the HMR header is as follows:

```
typedef struct
{
   char szTitleStr[12];              // $IMUHEADING[\0]; NULL terminated ASCII string
   unsigned char ucType;             // Set to 1 if external or 2 if dual antenna
   double dBoreSightRotationZ;       // Heading boresight rotation about Z, in degrees
                                     // Set to zero if unknown
                                     // Use positive boresight rotation as clockwise from north
                                     // IE will use the yaw definition by negating this so we have a
                                     // right-hand definition that fits with internal computations
   double dBoreSightRotationZStdDev;
                                     // accuracy of the boresight, in degrees; zero if unknown
   char Extra[227];                  // Reserved; bytes should be zeroed
} heading_hdr_type;                  // 256 bytes
```

The single header is then followed by the 34-byte structure type below for each heading update record:

```
typedef struct
{
   double dGpsTime;                  // GPS time of week, in seconds
   short sGpsWeek;                   // GPS week number; set to -1 if unknown
   double dHeading;                  // Heading update, in decimal degrees
                                     // Use positive rotation clockwise from north
                                     // IE uses yaw (i.e. rotation counterclockwise from north)
                                     // yaw = -heading
   float fHeadingStdDev;             // Standard deviation of update (decimal degrees)
                                     // zero if unknown
   float fBaselineLength;            // Distance between antennas, in metres
                                     // Only if ucType = 2 (dual antenna)
   float fPitch;                     // Pitch between two antennas, in degrees
                                     // Only if ucType = 2 (dual antenna)
   float fPitchStdDev;               // Standard deviation of the pitch, in degrees; zero if unknown
} heading_rec_type;                  // 34 bytes
```

3.2.4 PVA File

PVA files contain external position, velocity and/or attitude updates and can be input to Inertial Explorer within the *Updates* tab of the *Advanced IMU* options. The C/C++ structure definition of the PVA header is as follows:

```
struct pva_hdr_type                 //3848 bytes
{
    char headerString[12];          //Must be set equal to "$PVABIN\r\n\0"
    uint16_t fileVersion;           //Must be set equal to 1
    uint16_t headerSize;            //Must be set equal to sizeof(this) or 3848 bytes
    uint16_t reserved1;             //Reserved bytes
    uint16_t waypointVersion[3];    //[major, minor, build] (e.g. [8, 60, 1234]) (Optional)
    char programName[32];           //Program that created the file (Optional)
    uint32_t numRecs;               //Number of update records
    uint16_t recordSize;            //Must be set equal to sizeof(pva_rec_type) or 232 bytes
    uint16_t reserved3;             //Reserved bytes

    uint32_t reserved4;             //Reserved bytes
    uint16_t reserved5;             //Reserved bytes
    uint16_t updateLocation;        //Location the update is applied at. 0 = IMU, 1 = GNSS Antenna, 10 = Sensor
    uint16_t reserved6;             //Reserved bytes
    uint16_t reserved7;             //Reserved bytes
    uint16_t reserved8;             //Reserved bytes
    uint16_t reserved9[5];          //Reserved bytes

    char datum[32];                 //Datum name, "\0" if unknown
    char reserved10[128];           //Reserved bytes

    pva_offset_type Offset;         //Sensor offset information. Only necessary if updates are applied at
                                    //the sensor location

    uint16_t reserved11[1780];      //Reserved bytes
};
```

The pva_offset_type structure is defined as:

```
//Information about the sensor translational and rotational offset
struct pva_offset_type              //40 bytes
{
    //Sensor to IMU offset in the IMU body frame (Y-forward, X-right, Z-up) (m)
    float leverArmX;                //X-axis
    float leverArmY;                //Y-axis
    float leverArmZ;                //Z-axis
    uint16_t reserved[2];           //Reserved bytes
    double boresightAttitude[3];    //Boresight attitude, must match attitude input, only required if using
                                    // attitude updates (RPH) (deg)
};
```

The header is then followed by individual 232 byte PVA records, defined as follows:

```
//PVA record
struct pva_rec_type                 //232 bytes
{
    double updateEndTime;           //Time of week of end of update (0 s-604800 s)
    double reserved1;               //Reserved bytes
    uint16_t updateWeek;            //GPS Week number of update
    uint16_t reserved2;             //Reserved bytes
```

```
    //Bitmask of the type of update in this record. 0x01 = Position, 0x02 = Velocity, 0x04 = Attitude.
    //e.g. to apply position and velocity updates: updateType = 0; updateType |= 0x01; updateType |= 0x02;
    uint32_t updateType;
    char ID[16];                    //Update ID (Optional)
    uint16_t recordCheck;           //== 0xC0DE. Checks that record is valid
    uint16_t reserved4[3];          //Reserved bytes

    double position[3];             //Geographic coordinates [latitude, longitude, height] (deg/m)
    double attitude[3];             //Attitude [roll, pitch, heading] (deg)
    float velocity[3];              //Local level velocity [east, north, up] (m/s)

    float positionSD[3];            //Position standard deviation [east, north, up] (m)
    float attitudeSD[3];            //Attitude standard deviation [roll, pitch, heading] (deg)
    float velocitySD[3];            //Velocity standard deviation [east, north, up] (m/s)

    //Off-diagonal correlation coefficients (unitless)
    //Note: these values are optional, leave as 0 if they are unknown
    //Note: these are correlation coefficients (NOT covariance) defined as
    // cc = covar[i][j] / ( stdev[i] * stdev[j] ), scaled to a signed 16-bit
    // integer using the scale factor 65536. Algorithm:
    //    cc_int = (int)cc_float * 65536;
    //    if( cc_int < -65535 ) cc_int = -65535;
    //    if( cc_int >  65535 ) cc_int =  65535;
    //    cc_int16 = (int16_t)cc_int;
    //Note: this matrix can never be positive definite (e.g. cc != 1.0)
    // a function will be provided to do this conversion
    //Note: only the upper triangular (no diagonal) is stored. This requires 36 elements
    // for a 9 x 9 matrix. Use zeros if there is no known correlation
    //Note: correlation between velocity and attitude is almost always near zero
    //Input order chart (starting at element 0):
    //      #     0     1     2     3     4     5     6     7
    //            #     8     9     10    11    12    13    14
    //                  #     15    16    17    18    19    20
    //                        #     21    22    23    24    25
    //                              #     26    27    28    29
    //                                    #     30    31    32
    //                                          #     33    34
    //                                                #     35
    //                                                      #
    int16_t upperDiagCorrCoeffs[36];

    unsigned char reserved5[16];    //Reserved bytes
};
```

3.3 Output Files

This section discusses the different output files that are created when processing with Inertial Explorer.

3.3.1 FIL/RIL/FTL/RTL Files

Message Log files echo all error and warning messages sent to the *Process Window* during INS processing.

The forward and reverse loosely and tightly coupled message log files contain all messages output by the processing engine. Inertial Explorer assigns priority levels to all messages generated by the processor and only high priority messages are output to the *Process Window* during GNSS and INS data processing. All messages generated by the processor (regardless of priority) are output to the message log files. These files can be useful in helping to find problems that have not been automatically solved by Inertial Explorer's outlier detection routines.

3.3.2 FIM/RIM/FTM/RTM Files

These files contain the trajectory information computed by the inertial filter and are available for both tightly-coupled and loosely-coupled processing. They are output at the GNSS update interval.

The first line of the output file always begins with *$OUTREC*, and is followed by the version number, the processing engine, and the type of output. An example is given below:

```
$OUTREC Ver8.30.0329 GPSINSDLL Forward GpsInsOutput
```

The format of these ASCII trajectory files is outlined within the header of the forward/reverse files and are not discussed here.

3.3.3 SBTC/SBIC Files

SBTC is Inertial Explorer's extension for the smoothed and combined (forward+reverse) tightly coupled trajectory. SBIC is the equivalent for loosely coupled processing. The following is the C/C++ structure definition of the header, which is 512 bytes:

```
typedef struct
{
   char Str[16];                    // $IMUOUT
   long HdrSize;                    // size of this header
   long IsExtended;                 // true if extended format used
   long RecSize;                    // size of this record (see note below)
   long Reserved1;                  // for later, zero
   double Interval;                 // data interval (s)
   char ProgramName[32];            // program name that created this file
   char VersionName[32];            // version that produced file
   char Direction[16];              // "Forward", "Reverse" or "Combined"
   char Reserved2[392];             // reserved for future use (zero at creation of new file)
} imu_hdr_type;
```

✉ The RecSize variable should always be checked as the structure size and format may change in future versions. Contact Waypoint Support if you have any questions.

The single header is then followed by the 121-byte structure type below for each processed epoch:

```
typedef struct
{
   double GpsTime;                  // GPS time of this record (seconds of week)
   short WeekNum;                   // week number
   unsigned char FixedFlagAndHeave[2];  // bit 0: Fixed solution=on/float solution=off
                                    // bits 1-14: heave (+/-16.384 m)
                                    // bit 15: sign of heave
   plh_type_double GeoPos;          // geographic position of this record (deg, deg, m)
   fxyz_type LLVel;                 // local level velocity (m/s)
   fxyz_type LLAcc;                 // local level acceleration (m/s^2)
   iatt_type LLAtt;                 // local level attitude (deg) (scaled)
   fxyz_type LLAttDot;              // body frame rotational rate (deg/s)
   fxyz_type stdPos;                // position standard deviations (m)
   fxyz_type stdVel;                // velocity standard deviations (m/s)
   fxyz_type stdAtt;                // attitude standard deviations (deg)
   char Reserved[1];
   } imu_outrec_type;

typedef struct
{
   double phi, lamda;               // latitude and longitude (deg)
   double ht;                       // ellipsoidal height (m)
} plh_type_double;

typedef struct
{
   float x, y, z;
} fxyz_type;

typedef struct
{
   signed long iRoll, iPitch, iYaw;     // attitude (deg); scaled by 1.0e-6
} iatt_type;
```

The rotation matrix from body-to-local level is defined as: $R_B^{LL} = R_3(yaw) * R_1(pitch) * R_2(roll)$

Index

S

T

U

V