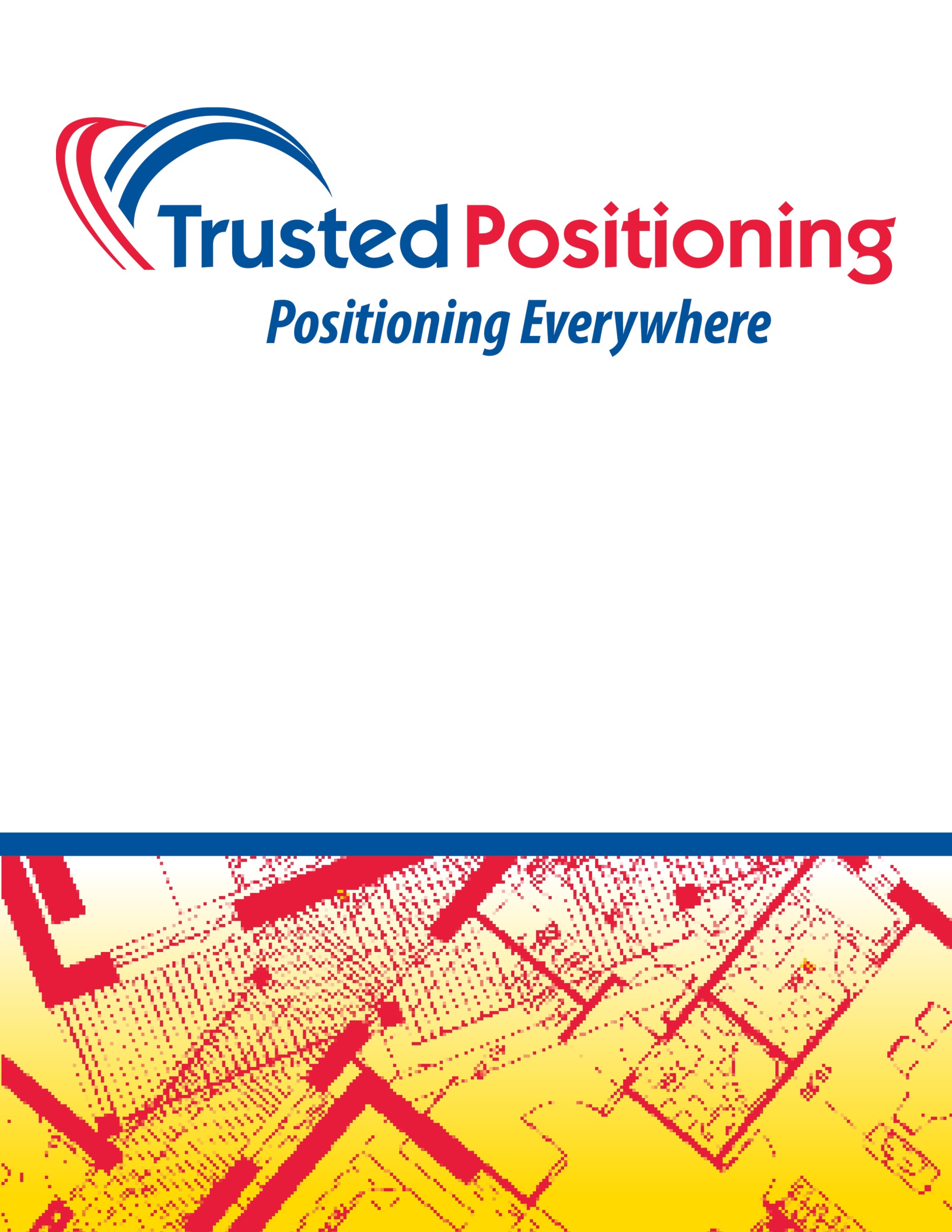
****

Trusted Positioning Navigation API

INTERNAL [TPN (f) (t), TMN(l)(a), TVN]

(Version Mj.MI.P-R)

Contents

[Introduction 2](#_Toc416260414)

[1. Acronyms and Abbreviations 3](#_Toc416260415)

[2. Definitions 4](#_Toc416260416)

[3. Naming conventions 5](#_Toc416260417)

[4. Files 6](#_Toc416260418)

[TPN\_DSL 7](#_Toc416260419)

[API Functions 7](#_Toc416260420)

[4.1. flag\_name\_ 7](#_Toc416260421)

[4.2. Entities 7](#_Toc416260422)

[5. TPN Free 8](#_Toc416260423)

[API Functions 8](#_Toc416260424)

[5.1. Flags 8](#_Toc416260425)

[5.2. Structures 9](#_Toc416260426)

[5.3. Enums 10](#_Toc416260427)

[5.4. Entities 11](#_Toc416260428)

[6. API Functions 12](#_Toc416260429)

[6.1. tpp\_get\_api\_version 12](#_Toc416260430)

[6.2. tpp\_create\_navigation\_session 12](#_Toc416260431)

[6.3. tpp\_initialize\_tpn\_free 12](#_Toc416260432)

[6.4. tpp\_initialize\_tpn\_tethered 13](#_Toc416260433)

[6.5. tpp\_initialize\_tmn\_land 13](#_Toc416260434)

[6.6. tpp\_initialize\_tmn\_aerial 14](#_Toc416260435)

[6.7. tpp\_initialize\_tvn 14](#_Toc416260436)

[6.8. tpp\_process\_gnss\_pvt 15](#_Toc416260437)

[6.9. tpp\_advance\_navigation\_step 15](#_Toc416260438)

[6.10. tpp\_process\_barometer 16](#_Toc416260439)

[6.11. tpp\_process\_magnetometer 16](#_Toc416260440)

[6.12. tpp\_process\_speed 16](#_Toc416260441)

[6.13. tpp\_process\_multiple\_antenna 16](#_Toc416260442)

[6.14. tpp\_process\_operator\_2d\_position 17](#_Toc416260443)

[6.15. tpp\_process\_wireless 17](#_Toc416260444)

[6.16. tpp\_process\_height 17](#_Toc416260445)

[6.17. tpp\_process\_call\_information 18](#_Toc416260446)

[6.18. tpp\_process\_floor\_information 18](#_Toc416260447)

[6.19. tpp\_process\_platform\_heading 19](#_Toc416260448)

[6.20. tpp\_process\_9dof\_quaternions 19](#_Toc416260449)

[6.21. tpp\_process\_6dof\_quaternions 19](#_Toc416260450)

[6.22. tpp\_process\_imu\_biases 20](#_Toc416260451)

[6.23. tpp\_process\_misalignment 20](#_Toc416260452)

[6.24. tpp\_process\_venue\_map 20](#_Toc416260453)

[6.25. tpp\_set\_mode\_of\_transit 21](#_Toc416260454)

[6.26. tpp\_set\_zupt\_mode 21](#_Toc416260455)

[6.27. tpp\_set\_misalignment 21](#_Toc416260456)

[6.28. tpp\_set\_magnetometer\_calibration 22](#_Toc416260457)

[6.29. tpp\_set\_use\_case 22](#_Toc416260458)

[6.30. tpp\_set\_orientation\_based\_on\_pitch (INTERNAL ONLY) 22](#_Toc416260459)

[6.31. tpp\_set\_device\_heading 23](#_Toc416260460)

[6.32. tpp\_add\_anchor\_point 23](#_Toc416260461)

[6.33. tpp\_add\_synchronization\_event 23](#_Toc416260462)

[6.34. tpp\_stop\_navigation 24](#_Toc416260463)

[6.35. tpp\_delete\_navigation\_session 24](#_Toc416260464)

[6.36. tpp\_prerun\_reset\_zupt\_thresholds 24](#_Toc416260465)

[6.37. tpp\_prerun\_compute\_zupt\_thresholds\_imu 24](#_Toc416260466)

[6.38. tpp\_run\_backward\_smoothing 25](#_Toc416260467)

[6.39. tpp\_process\_gnss\_observations 25](#_Toc416260468)

[6.40. tpp\_process\_glonass\_ephemeris 26](#_Toc416260469)

[6.41. tpp\_process\_gps\_ephemeris 26](#_Toc416260470)

[6.42. tpp\_multi\_device\_process\_position\_velocity 26](#_Toc416260471)

[6.43. tpp\_multi\_device\_process\_heading 27](#_Toc416260472)

[6.44. tpp\_multi\_device\_process\_secondary\_gnss\_pvt 27](#_Toc416260473)

[7. Data Types 29](#_Toc416260474)

[7.1. TppNavigationSessionHandle 29](#_Toc416260475)

[7.2. TppInitializationStruct 29](#_Toc416260476)

[7.3. TppApiVersionStruct 40](#_Toc416260477)

[7.4. TppGnssPvtMessageStruct 41](#_Toc416260478)

[7.5. TppImuMessageStruct 41](#_Toc416260479)

[7.6. TppBarometerMessageStruct 41](#_Toc416260480)

[7.7. TppMagnetometerMessageStruct 42](#_Toc416260481)

[7.8. TppSpeedMessageStruct 43](#_Toc416260482)

[7.9. TppMultipleAntennaMessageStruct 43](#_Toc416260483)

[7.10. TppOperator2dPositionMessageStruct 43](#_Toc416260484)

[7.11. TppWirelessMessageStruct 43](#_Toc416260485)

[7.12. TppHeightMessageStruct 44](#_Toc416260486)

[7.13. TppFloorInformationMessageStruct 44](#_Toc416260487)

[7.14. TppCallInformationMessageStruct 45](#_Toc416260488)

[7.15. TppPlatformHeadingMessageStruct 45](#_Toc416260489)

[7.16. TppDeviceHeadingMessageStruct 45](#_Toc416260490)

[7.17. TppQuaternionsMessageStruct 45](#_Toc416260491)

[7.18. TppImuBiasesMessageStruct 45](#_Toc416260492)

[7.19. TppVenueMapMessageStruct 46](#_Toc416260493)

[7.20. TppProcessMisalignmentMessageStruct 47](#_Toc416260494)

[7.21. TppSetMisalignmentMessageStruct 47](#_Toc416260495)

[7.22. TppSolutionStruct 47](#_Toc416260496)

[7.23. TppPacketStruct 48](#_Toc416260497)

[7.24. TppModeOfTransitEnum 50](#_Toc416260498)

[7.25. TppZuptModeEnum 50](#_Toc416260499)

[7.26. TppMagnetometerCalibrationEnum 50](#_Toc416260500)

[7.27. TppUseCaseEnum 50](#_Toc416260501)

[7.28. TppOrientationBasedOnPitchEnum (INTERNAL only) 51](#_Toc416260502)

[7.29. TppAnchorPointStruct 51](#_Toc416260503)

[7.30. TppZuptThresholdsStruct 51](#_Toc416260504)

[7.31. TppMultiDeviceLatencyStruct 51](#_Toc416260505)

[7.32. TppMultiDevicePositionVelocityStruct 51](#_Toc416260506)

[7.33. TppMultiDeviceHeadingStruct 52](#_Toc416260507)

[7.34. TppReturnStatusEnum 53](#_Toc416260508)

[7.35. TppGnssObservationsMessageStruct 57](#_Toc416260509)

[7.36. TppGpsEphemerisMessageStruct 58](#_Toc416260510)

[7.37. TppGlonassEphemerisMessageStruct 59](#_Toc416260511)

[8. Using Trusted Positioning Library 60](#_Toc416260512)

[8.1. Initialization 60](#_Toc416260513)

[8.2. Data Processing 73](#_Toc416260514)

[8.3. Output Array Parsing 74](#_Toc416260515)

[9. Output Array Entities 75](#_Toc416260516)

[9.1. Time 75](#_Toc416260517)

[9.2. Position 75](#_Toc416260518)

[9.3. Position Standard Deviation 75](#_Toc416260519)

[9.4. Velocity 75](#_Toc416260520)

[9.5. Velocity Standard Deviation 75](#_Toc416260521)

[9.6. Attitude 76](#_Toc416260522)

[9.7. Attitude Standard Deviation 76](#_Toc416260523)

[9.8. Accelerometer Bias 76](#_Toc416260524)

[9.9. Gyroscope Bias 76](#_Toc416260525)

[9.10. Heading Misalignment 76](#_Toc416260526)

[9.11. Platform Heading 77](#_Toc416260527)

[9.12. Stride Information 77](#_Toc416260528)

[9.13. Number of Steps 77](#_Toc416260529)

[9.14. Floor Number 77](#_Toc416260530)

[9.15. Converged Gyroscope Bias 77](#_Toc416260531)

[9.16. Flags - Magnetometer and Barometer 77](#_Toc416260532)

[9.17. Mode of Transit 78](#_Toc416260533)

[9.18. Use Case 78](#_Toc416260534)

[9.19. Flags – GNSS, Speed and Static Status 79](#_Toc416260535)

[9.20. Flags – Navigation Phase 79](#_Toc416260536)

[9.21. Orientation Based On Pitch 80](#_Toc416260537)

[9.22. Distance Travelled 80](#_Toc416260538)

[9.23. DSL Height (For TPP/DSL only) 80](#_Toc416260539)

[9.24. Raw IMU Data\* 80](#_Toc416260540)

[9.25. GNSS PVT Data\* 80](#_Toc416260541)

[9.26. Barometer Data\* 81](#_Toc416260542)

[9.27. Magnetometer Data\* 81](#_Toc416260543)

[9.28. Speed Data\* 82](#_Toc416260544)

[9.29. Operator 2D Position Data\* 82](#_Toc416260545)

[9.30. Wireless Data\* 82](#_Toc416260546)

[9.31. Floor Information Data\* 84](#_Toc416260547)

[9.32. Call Information Data\* 84](#_Toc416260548)

[9.33. Set Mode of Transit Event Data\* 84](#_Toc416260549)

[9.34. Platform Heading Data\* 84](#_Toc416260550)

[9.35. Device Heading Data\* 85](#_Toc416260551)

[9.36. 9-DOF Quaternions Data\* 85](#_Toc416260552)

[9.37. IMU Biases Data\* 85](#_Toc416260553)

[9.38. Process Misalignment Data\* 86](#_Toc416260554)

[9.39. Set ZUPT Mode Event Data\* 86](#_Toc416260555)

[9.40. Set Misalignment Event Data\* 86](#_Toc416260556)

[9.41. Set Magnetometer Calibration Status Event Data\* 86](#_Toc416260557)

[9.42. Set Use Case Event Data\* 87](#_Toc416260558)

[9.43. 6-DOF Quaternions Data\* 87](#_Toc416260559)

[9.44. Venue Map Data\* 87](#_Toc416260560)

[9.45. Anchor Point Data\* 88](#_Toc416260561)

[9.46. Synchronization Event Data\* 89](#_Toc416260562)

[9.47. Orientation Based On Pitch Data\* 89](#_Toc416260563)

[9.48. Multi-Device Position Velocity Data\* 89](#_Toc416260564)

[9.49. Multi-Device Heading Data\* 90](#_Toc416260565)

[9.50. Multi-Device Secondary GNSS PVT Data\* 92](#_Toc416260566)

[9.51. Multi-Device GNSS PVT Data Source\* 92](#_Toc416260567)

[9.52. In-run Magnetometer Calibration Information^ [Internal Debugging Only] 93](#_Toc416260568)

[9.53. Internal Debugging Information (1)^ [Internal Debugging Only] 94](#_Toc416260569)

[10. DEPRECATED ENTITIES 97](#_Toc416260570)

[10.1. Magnetometer Data\* [Deprecated – As of version 3.0.0] 97](#_Toc416260571)

[10.2. Wi-Fi 2D Position Data\* [Deprecated – As of version 4.1.0] 97](#_Toc416260572)

[10.3. Operator 2D Position Data\* [Deprecated as of version 4.1.0] 97](#_Toc416260573)

[10.4. Wi-Fi Data\* [Deprecated] 98](#_Toc416260574)

[11. SUGGESTED ENTITIES 100](#_Toc416260575)

[11.1. Accelerometer Bias Standard Deviation 100](#_Toc416260576)

[11.2. Gyroscope Bias STD Outputs 100](#_Toc416260577)

[11.3. Corrected Accel Outputs: 100](#_Toc416260578)

[11.4. Corrected Gyro Outputs: 100](#_Toc416260579)

[11.5. Declination Angle 100](#_Toc416260580)

[11.6. Mag Pre-Calibration Outputs: 101](#_Toc416260581)

[11.7. Mag Online Calibration Outputs: 101](#_Toc416260582)

[11.8. Magnetometer Derived Heading 101](#_Toc416260583)

[11.9. Corrected Mag Outputs: 101](#_Toc416260584)

[11.10. Barometer Height 101](#_Toc416260585)

[11.11. Odometer Scale Factor: 102](#_Toc416260586)

[11.12. Corrected Odometer Speed: 102](#_Toc416260587)

[11.13. Corrected odometer Speed STD: 102](#_Toc416260588)

[11.14. Roll Misalignment Outputs: 102](#_Toc416260589)

[11.15. Roll Misalignment STD Output: 102](#_Toc416260590)

[11.16. Pitch Misalignment Output: 102](#_Toc416260591)

[11.17. Pitch Misalignment STD Output: 102](#_Toc416260592)

[11.18. Heading Misalignment STD Output: 103](#_Toc416260593)

[11.19. Resource Consumption 103](#_Toc416260594)

[11.20. Flags - Events 103](#_Toc416260595)

[11.21. Flags – External Position 103](#_Toc416260596)

[11.22. Flags – Vertical and Secondary Filter Info 104](#_Toc416260597)

[11.23. GNSS Raw Data\* 104](#_Toc416260598)

[11.24. GPS Ephemeris Data\* 104](#_Toc416260599)

[11.25. GLONASS Ephemeris Data\* 105](#_Toc416260600)

[11.26. Ionospheric Parameters\* 106](#_Toc416260601)

[11.27. Satellite Parameters\* 106](#_Toc416260602)

[11.28. Multiple Antenna Data\* 107](#_Toc416260603)

[11.29. Temperature Sensor Data\* 107](#_Toc416260604)

[11.30. Heading Data\* 107](#_Toc416260605)

[11.31. Map Matching Data\* 107](#_Toc416260606)

[11.32. Raw Map database Option1\* 107](#_Toc416260607)

[11.33. Raw Map database Option2\* 108](#_Toc416260608)

[11.34. WiFi Beacon Message\* 108](#_Toc416260609)

[11.35. Height Data\* 108](#_Toc416260610)

[11.36. Heading Data\* 108](#_Toc416260611)

[11.37. Other Wireless Position/Heading Data\* 109](#_Toc416260612)

[12. Output Entities For Each Version 110](#_Toc416260613)

[12.1. 2.0.0-b 110](#_Toc416260614)

[12.2. 2.1.0-rc/3.0.0 110](#_Toc416260615)

[12.3. 4.0.0-x 111](#_Toc416260616)

[12.4. 4.1.0-x 112](#_Toc416260617)

[13. Appendix A: Predefined Configurations 114](#_Toc416260618)

[13.1. Predefined GNSS Configurations 114](#_Toc416260619)

[13.2. Predefined Accelerometer Configurations 114](#_Toc416260620)

[13.3. Predefined Gyroscope Configurations 114](#_Toc416260621)

[13.4. Predefined Motion Constraints Configurations 115](#_Toc416260622)

[Change Log 116](#_Toc416260623)

[Revision History 119](#_Toc416260624)

## Introduction

Trusted Positioning Navigation API is a proprietary interface designed by Trusted Positioning. The document presents the function prototypes and data types available for application developers via the API. The first two sections explain the acronyms, abbreviations and definitions that are used throughout the document. Section (3) explains the naming conventions used by the API. The fourth, fifth and sixth sections present the files, function prototypes and data types that the application developers will use to embed multi-sensor based integration navigation functionality in their applications. Then, section (7) explains how to use the API by example code.

Trusted Positioning library is written in ANSI C code and does not depend on any third party libraries. The library is tested and supported on a variety of operating systems and platforms including – but not limited to – Microsoft Windows, and UNIX-Based operating systems.

## Acronyms and Abbreviations

The list of acronyms and abbreviations used in the document is presented in Table 1.

Table List of Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| GNSS | Global Navigation Satellite System |
| GPS | Global Positioning System |
| PVT | Position-Velocity-Time |
| IMU | Inertial Measurement Unit |
| TPP | Trusted Positioning Platform |
| T-PN,TPN | Trusted Portable Navigator |
| T-MN,TMN | Trusted Machine Navigator |
| T-VN,TVN | Trusted Vehicle Navigator |
| ZUPT | Zero Velocity Update |
| m | Metre |
| sec | Second |
| deg | Degree |
| rad | Radian |
| ˚C | Degree Celsius |
| mBar | milli-Bar |
| mG | milli-Gauss |
| Hz | Hertz |
| mGal | milli-Gal |

## Definitions

Table List of Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Navigation Device Frame | The navigation device is the device containing the sensor assembly.  Trusted Positioning uses a device frame in which:   1. X-axis is the forward axis of the navigation device 2. Z-axis is the vertical axis of the navigation device pointing downwards 3. Y-axis is the lateral axis completing the right hand rule with the X, and Z axes |
| Epoch | Processing step related to the IMU data rate which is the highest available sensors’ rate. |
| Height | Unless otherwise stated, height can be either “height above sea level” or “height above ellipsoid”, but it should be consistent across all height inputs to the API functions through the same navigation session (i.e. from initialization to stop). The height from the navigation solution will follow the same definition as the input. |
| Heading | The heading angle is measured clockwise from the North direction. |
| Standard Deviation | Any standard deviation values provided to the library functions should be greater than zero. |
| Application Developer | The user of the library. “Application Developer” and “Developer” are used interchangeably in the document. In some instances, “User” will also be used. |
| Operator | The user of the application implemented by the developer. |

## Naming conventions

The naming conventions used for files, functions, and data types are defined in Table 3.

Table Naming Conventions Used by Trusted Positioning API

|  |  |
| --- | --- |
|  | All files, functions and data types start with “tpp” which stands for **T**rusted **P**ositioning **P**latform. The first letter can be lower or upper case depending on the rules specified in this table. |
| Files | File names are lower-case and words are separated by underscores.  **Example:** tpp\_data\_types.h, tpp\_api\_functions.h |
| Functions | Function names are lower-case and words are separated by underscores.  **Example:** tpp\_process\_barometer(…)  **Note:** The parameters are organized so that input parameters are ordered before output parameters. |
| Data Types | Data types start with a capital letter and a capital letter is added for each word, with no underscores between words. The type ends with the word Struct or Enum depending on the type.  **Example:** TppImuMessageStruct, TppReturnStatusEnum |
| Structure Members | Structure member variables do not start with “tpp” because they are enclosed within the structure. Member variables are lower-case and they have a trailing underscore.  **Example:** timetag\_ |
| Enumerators | Enumerators are upper-case and words are separated by underscores.  **Example:** TPP\_INVALID\_IMU\_MESSAGE |

## Files

There are three files that the application developer will have access to, in order to embed the functionality provided by the library in the developer’s code. The files are presented in Table 4.

Table Trusted Positioning Library Files

|  |  |
| --- | --- |
| **File Name** | **Description** |
| tpp\_data\_types.h | This header file defines the initialization, input messages, output structure, and return-status data types that can be used by the developer directly in the code without any need to implement the structures from this document. |
| tpp\_api\_functions.h | This header file defines the API function pointers types that the developer can use directly when using the library. |
| LIBRARY- FILE | The library file that implements the functionality specified by the API. The name of the library file along with the file extension depends on the platform that the library is compiled for.  The name of the library file would follow the convention below:  **libtpp\_<*navigator*>\_<processor>\_<*operating\_system*>.<*extension*>**  For example, the name for a TPN/Free library for Android would be:  **libtpp\_tpn\_free\_arm\_android.so** |

## TPN\_DSL

### API Functions

tpp\_get\_api\_version

tpp\_create\_navigation\_session

tpp\_initialize \_dsl

tpp\_advance\_navigation\_step

tpp\_process\_barometer

tpp\_process\_6dof\_quaternions

tpp\_process\_imu\_biases

tpp\_delete\_navigation\_session

### flag\_name\_

“DEVICE”

“BAROMETER”

“GNSS”

“PLATFORM\_HEADING”

“POSITION”

“MODE\_OF\_TRANSIT”

“USE\_CASE”

“GYROSCOPE\_BIASES”

“ACCELEROMETER\_CONFIGURATION”

“[GYROSCOPE\_X\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)”

“[GYROSCOPE\_Y\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)”

“[GYROSCOPE\_Z\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)”

“DEBUG\_DATA”

### Entities

Entity Id: 0x008C

## TPN Free

### API Functions

tpp\_get\_api\_version

tpp\_create\_navigation\_session

tpp\_initialize\_tpn\_free

tpp\_process\_gnss\_pvt

tpp\_advance\_navigation\_step

tpp\_process\_barometer

tpp\_process\_magnetometer

tpp\_process\_speed

tpp\_process\_operator\_2d\_position

tpp\_process\_wireless

tpp\_process\_height

tpp\_process\_call\_information

tpp\_process\_floor\_information

tpp\_process\_platform\_heading

tpp\_process\_9dof\_quaternions

tpp\_process\_6dof\_quaternions

tpp\_process\_imu\_biases

tpp\_process\_misalignment

tpp\_process\_venue\_map

tpp\_set\_mode\_of\_transit

tpp\_set\_zupt\_mode

tpp\_set\_misalignment

tpp\_set\_magnetometer\_calibration

tpp\_set\_use\_case

tpp\_set\_device\_heading

tpp\_add\_anchor\_point

tpp\_stop\_navigation

tpp\_delete\_navigation\_session

tpp\_prerun\_reset\_zupt\_thresholds

tpp\_prerun\_compute\_zupt\_thresholds\_imu

tpp\_process\_gnss\_observation­s

### Flags

“[DEVICE](#DEVICE_FLAG)”

“[START\_OPTION](#START_OPTION_FLAG)”

“[MAGNETOMETER](#MAGNETOMETER_FLAG)”

“[BAROMETER](#BAROMETER)”

“[SPEED](#SPEED_FLAG)”

“[GNSS](#GNSS_START_OPTION_FLAG)”

“[WIFI](#WIFI_FLAG)”

“[PLATFORM\_HEADING](#PLATFORM_HEADING_FLAG)”

“[POSITION](#PLATFORM_POSITION_FLAG)”

“[MISALIGNMENT](#MISALIGNMENT_FLAG)”

“[MODE\_OF\_TRANSIT](#MODE_OF_TRANSIT_FLAG)”

“USE\_CASE”

“[FLOOR\_INFORMATION](#FLOOR_INFORMATION_FLAG)”

“[GYROSCOPE\_BIASES](#GYROSCOPE_BIASES_FLAG)”

“[ACCELEROMETER\_CONFIGURATION](#ACCELEROMTER_CONFIGURATION)”

[GYROSCOPE\_X\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)”

“[GYROSCOPE\_Y\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)”

“[GYROSCOPE\_Z\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)”

“[MOTION\_CONSTRAINTS](#MOTION_CONSTRAINTS_FLAG)”

“[DEBUG\_DATA](#DEBUG_DATA_FLAG)”

“OUTPUT\_POSITION\_ERROR\_WEIGHTING

“OUTPUT\_VELOCITY\_ERROR\_WEIGHTING”

“OUTPUT\_ATTITUDE\_ERROR\_WEIGHTING”“ACCELEROMETER\_BIASES”

“ZUPT\_THRESHOLDS”

“DECLINATION\_ANGLE”

“REPLACE\_ATTITUDE\_WITH\_6DOF”

“PROCESSING\_MODE”

“BACKWARD\_SMOOTHING”

“LIBRARY\_CALL\_TYPE”

### Structures

TppApiVersionStruct

TppGnssPvtMessageStruct

TppImuMessageStruct

TppBarometerMessageStruct

TppMagnetometerMessageStruct

TppSpeedMessageStruct

TppOperator2dPositionMessageStruct

TppWirelessMessageStruct

TppHeightMessageStruct

TppFloorInformationMessageStruct

TppCallInformationMessageStruct

TppPlatformHeadingMessageStruct

TppDeviceHeadingMessageStruct

TppQuaternionsMessageStruct

TppImuBiasesMessageStruct

TppVenueMapMessageStruct

TppProcessMisalignmentMessageStruct

TppSetMisalignmentMessageStruct

TppSetMisalignmentMessageStruct

TppSolutionStruct

TppPacketStruct

Output Array Structure

### Enums

6.24. TppModeOfTransitEnum

TppZuptModeEnum

TppMagnetometerCalibrationEnum

TppUseCaseEnum

TppAnchorPointStruct

TppZuptThresholdsStruct

TppReturnStatusEnum

SUCCESS

TPP\_SYSTEM\_NOT\_INITIALIZED

* TPP\_INVALID\_IMU\_MESSAGE
* TPP\_INVALID\_GNSS\_PVT\_MESSAGE
* TPP\_INVALID\_BAROMETER\_MESSAGE
* TPP\_INVALID\_MAGNETOMETER\_MESSAGE
* TPP\_INVALID\_SPEED\_MESSAGE
* ~~TPP\_INVALID\_2D\_POSITION\_MESSAGE~~
* TPP\_INVALID\_HEIGHT\_MESSAGE
* TPP\_INVALID\_FLOOR\_INFORMATION\_MESSAGE
* TPP\_INVALID\_CALL\_INFORMATION\_MESSAGE
* ~~TPP\_INVALID\_PROCESS\_2D\_POSITION\_CALL~~
* TPP\_INVALID\_SET\_MODE\_OF\_TRANSIT\_CALL
* TPP\_INVALID\_FLAG\_MAGNETOMETER
* TPP\_INVALID\_FLAG\_BAROMETER
* TPP\_INVALID\_FLAG\_SPEED
* TPP\_INVALID\_FLAG\_GNSS
* TPP\_INVALID\_HEADING\_TO\_START
* TPP\_INVALID\_POSITION\_TO\_START
* TPP\_INVALID\_FLAG\_WIFI
* TPP\_INVALID\_FLAG\_PLATFORM\_HEADING
* TPP\_INVALID\_FLAG\_POSITION
* TPP\_INVALID\_FLAG\_MISALIGNMENT
* TPP\_INVALID\_FLAG\_MODE\_OF\_TRANSIT
* TPP\_INVALID\_FLAG\_MODE\_OF\_TRANSIT
* TPP\_INVALID\_FLAG\_GYROSCOPE\_BIASES
* TPP\_INVALID\_GYROSCOPE\_BIASES\_TO\_START
* TPP\_INVALID\_FLAG\_ACCELEROMETER\_CONFIGURATION
* TPP\_INVALID\_FLAG\_GYROSCOPE\_X\_CONFIGURATION
* TPP\_INVALID\_FLAG\_GYROSCOPE\_Y\_CONFIGURATION
* TPP\_INVALID\_FLAG\_GYROSCOPE\_Z\_CONFIGURATION
* TPP\_INVALID\_FLAG\_DEBUG\_DATA
* TPP\_INVALID\_FLAG\_MOTION\_CONSTRAINTS
* TPP\_INVALID\_IMU\_MESSAGE\_TIMETAG
* TPP\_INVALID\_IDENTIFIER\_TO\_START
* TPP\_INVALID\_MODE\_OF\_TRANSIT\_TO\_START
* TPP\_INVALID\_FLAG\_BACKWARD\_SMOOTHING
* TPP\_BACKWARD\_SMOOTHING\_FILES\_CREATION\_FAILURE
* TPP\_INVALID\_RUN\_BACKWARD\_SMOOTHING\_CALL
* TPP\_RUN\_BACKWARD\_SMOOTHING\_FAILURE
* TPP\_INVALID\_SYSTEM\_DATE\_TO\_START
* TPP\_INVALID\_FLAG\_START\_OPTION
* TPP\_INVALID\_OPTION\_SENSORS\_ONLY
* TPP\_INVALID\_OPTION\_SENSORS\_RELIABLE\_WIRELESS
* TPP\_INVALID\_OPTION\_SENSORS\_WIRELESS
* TPP\_INVALID\_FLAG\_ACCELEROMETER\_BIASES
* TPP\_INVALID\_PLATFORM\_HEADING\_MESSAGE
* TPP\_INVALID\_QUATERNIONS\_MESSAGE
* TPP\_INVALID\_SET\_ZUPT\_MODE\_CALL
* TPP\_INVALID\_SET\_MISALIGNMENT\_MESSAGE
* TPP\_INVALID\_SET\_MISALIGNMENT\_CALL
* TPP\_INVALID\_SET\_MAGNETOMETER\_CALIBRATION\_CALL
* TPP\_INVALID\_SET\_USE\_CASE\_CALL
* TPP\_INVALID\_FLAG\_OUTPUT\_POSITION\_ERROR\_WEIGHTING
* TPP\_INVALID\_FLAG\_OUTPUT\_VELOCITY\_ERROR\_WEIGHTING
* TPP\_INVALID\_FLAG\_OUTPUT\_ATTITUDE\_ERROR\_WEIGHTING
* TPP\_INVALID\_IMU\_BIASES\_MESSAGE
* TPP\_INVALID\_OPERATOR\_2D\_POSITION\_MESSAGE
* TPP\_INVALID\_WIRELESS\_MESSAGE
* TPP\_INVALID\_PROCESS\_WIFI\_CALL
* TPP\_INVALID\_PROCESS\_MISALIGNMENT\_MESSAGE
* TPP\_INVALID\_DEVICE\_HEADING\_MESSAGE
* TPP\_INVALID\_NAVIGATION\_SESSION\_HANDLE
* TPP\_INVALID\_FLAG\_DECLINATION\_ANGLE
* TPP\_INVALID\_FLAG\_REPLACE\_ATTITUDE\_WITH\_6DOF
* TPP\_INVALID\_ANCHOR\_POINT\_PACKET
* TPP\_INVALID\_START\_ANCHOR\_POINT
* TPP\_INVALID\_END\_ANCHOR\_POINT
* TPP\_INVALID\_FLAG\_PROCESSING\_MODE

### Entities

Time Id: **0x00EA**

Position Id: **0x0016**

Position Standard Deviation Id: **0x00BD**

Velocity Id: **0x00A2**

Velocity Standard Deviation Id: **0x005F**

Attitude Id: **0x00E9**

Attitude Standard Deviation Id: **0x0060**

Accelerometer Bias Id: **0x006F**

Gyroscope Bias Id: **0x002E**

## API Functions

The section presents the functions for the application developers.

### tpp\_get\_api\_version

|  |  |
| --- | --- |
| Prototype | void tpp\_get\_api\_version(TppApiVersionStruct\* version\_struct\_pointer) |
| Summary | The function is used to get the version of the API |
| Parameters | [OUT] version\_struct\_pointer |
| Return Value | [NONE] |
| Notes | 1. The function returns the major, minor, patch and release versions of the API  2. The major, minor, patch and release versions of the API can also be accessed through the #define statements that are provided in the file “tpp\_data\_types.h”:  TPP\_API\_VERSION\_MAJOR  TPP\_API\_VERSION\_MINOR  TPP\_API\_VERSION\_PATCH  TPP\_API\_VERSION\_RELEASE\_ID  TPP\_API\_VERSION\_RELEASE\_NUMBER  TPP\_API\_VERSION\_SET\_NUMBER |

### tpp\_create\_navigation\_session

|  |  |
| --- | --- |
| Prototype | TppNavigationSessionHandle tpp\_create\_navigation\_session\_handle() |
| Summary | The function is used to create a navigation session handle. |
| Parameters | [NONE] |
| Return Value | If successful, a valid handle is returned. Otherwise, NULL is returned. |
| Notes | 1. The function must be called once before any other functions that use a parameter of type TppNavigationSessionHandle is called.  2. If the other API functions are called with a NULL handle, TPP\_INVALID\_NAVIGATION\_SESSION\_HANDLE is returned. |

### tpp\_initialize\_tpn\_free

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_initialize\_tpn\_free(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppInitializationStruct\* initialization\_struct\_pointer , TppPacketStruct\* init\_packet\_struct\_pointer) |
| Summary | The function is used to initialize TPN Free. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] initialization\_struct\_pointer  [OUT] init\_packet\_struct\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Section 7.28 shows all the values that can be returned by the initialization function. |
| Notes | 1. The function must be called once after tpp\_create\_navigation\_session() and before any other functions are called.  2. The init\_packet\_struct\_pointer variable contains the populated byte array and its size that the developer can use to write such array in the beginning of any dataset file ahead of the integrated solution output packets that are generated every IMU sample.  3. The application developer is responsible for allocating the memory for the byte\_array\_pointer\_ member of the input argument init\_packet\_struct\_pointer with the size TPP\_INIT\_BYTE\_ARRAY\_LENGTH and consequently is responsible for freeing the allocated memory space. |

### tpp\_initialize\_tpn\_tethered

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_initialize\_tpn\_tethered(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppInitializationStruct\* initialization\_struct\_pointer , TppPacketStruct\* init\_packet\_struct\_pointer) |
| Summary | The function is used to initialize TPN Tethered. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] initialization\_struct\_pointer  [OUT] init\_packet\_struct\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Section 7.28 shows all the values that can be returned by the initialization function. |
| Notes | 1. The function must be called once before any other functions are called.  2. The init\_packet\_struct\_pointer variable contains the populated byte array and its size that the developer can use to write such array in the beginning of any dataset file ahead of the integrated solution output packets that are generated every IMU sample.  3. The application developer is responsible for allocating the memory for the byte\_array\_pointer\_ member of the input argument init\_packet\_struct\_pointer with the size TPP\_INIT\_BYTE\_ARRAY\_LENGTH and consequently is responsible for freeing the allocated memory space. |

### tpp\_initialize\_tmn\_land

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_initialize\_tmn\_land(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppInitializationStruct\* initialization\_struct\_pointer , TppPacketStruct\* init\_packet\_struct\_pointer) |
| Summary | The function is used to initialize TMN Land. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] initialization\_struct\_pointer  [OUT] init\_packet\_struct\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Section 7.28 shows all the values that can be returned by the initialization function. |
| Notes | 1. The function must be called once before any other functions are called.  2. The init\_packet\_struct\_pointer variable contains the populated byte array and its size that the developer can use to write such array in the beginning of any dataset file ahead of the integrated solution output packets that are generated every IMU sample.  3. The application developer is responsible for allocating the memory for the byte\_array\_pointer\_ member of the input argument init\_packet\_struct\_pointer with the size TPP\_INIT\_BYTE\_ARRAY\_LENGTH and consequently is responsible for freeing the allocated memory space. |

### tpp\_initialize\_tmn\_aerial

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_initialize\_tmn\_aerial(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppInitializationStruct\* initialization\_struct\_pointer , TppPacketStruct\* init\_packet\_struct\_pointer) |
| Summary | The function is used to initialize TMN Aerial. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] initialization\_struct\_pointer  [OUT] init\_packet\_struct\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Section 7.28 shows all the values that can be returned by the initialization function. |
| Notes | 1. The function must be called once before any other functions are called.  2. The init\_packet\_struct\_pointer variable contains the populated byte array and its size that the developer can use to write such array in the beginning of any dataset file ahead of the integrated solution output packets that are generated every IMU sample.  3. The application developer is responsible for allocating the memory for the byte\_array\_pointer\_ member of the input argument init\_packet\_struct\_pointer with the size TPP\_INIT\_BYTE\_ARRAY\_LENGTH and consequently is responsible for freeing the allocated memory space. |

### tpp\_initialize\_tvn

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_initialize\_tvn(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppInitializationStruct\* initialization\_struct\_pointer , TppPacketStruct\* init\_packet\_struct\_pointer) |
| Summary | The function is used to initialize TVN. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] initialization\_struct\_pointer  [OUT] init\_packet\_struct\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Section 7.28 shows all the values that can be returned by the initialization function. |
| Notes | 1. The function must be called once before any other functions are called.  2. The init\_packet\_struct\_pointer variable contains the populated byte array and its size that the developer can use to write such array in the beginning of any dataset file ahead of the integrated solution output packets that are generated every IMU sample.  3. The application developer is responsible for allocating the memory for the byte\_array\_pointer\_ member of the input argument init\_packet\_struct\_pointer with the size TPP\_INIT\_BYTE\_ARRAY\_LENGTH and consequently is responsible for freeing the allocated memory space. |

### tpp\_process\_gnss\_pvt

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_gnss\_pvt(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppGnssPvtMessageStruct\* gnss\_pvt\_message\_pointer) |
| Summary | This function is used when a valid data message from GNSS receiver is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] gnss\_pvt\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise, the return value is TPP\_INVALID\_GNSS\_PVT\_MESSAGE or TPP\_INVALID\_PROCESS\_GNSS\_PVT\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. The function must be used only when valid GNSS data is available with a 3D fix status from the GNSS receiver. |

### tpp\_advance\_navigation\_step

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_advance\_navigation\_step(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppImuMessageStruct\* imu\_message\_pointer,  TppSolutionStruct\* solution\_struct\_pointer,  TppPacketStruct\* output\_packet\_struct\_pointer) |
| Summary | This function is used when a valid IMU data message is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] imu\_message\_pointer  [OUT] solution\_struct\_pointer  [OUT] output\_packet\_struct\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise, the return value is TPP\_INVALID\_IMU\_MESSAGE, TPP\_SYSTEM\_NOT\_INITIALIZED, TPP\_INVALID\_IMU\_MESSAGE\_TIMETAG or TPP\_INVALID\_OUTPUT |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid IMU data is available.  3. If GNSS or any other sensor data sample is available at the same epoch at which an IMU data sample is available, tpp\_advance\_navigation\_step() should be called last after tpp\_process\_gnss\_pvt() and any other functions.  4. The application developer is responsible for allocating the memory for the byte\_array\_pointer\_ member of the input argument output\_packet\_struct\_pointer with the size TPP\_OUTPUT\_BYTE\_ARRAY\_LENGTH and consequently is responsible for freeing the allocated memory space.  5. The developer can set any of the arguments solution\_struct\_pointer or output\_packet\_struct\_pointer to NULL if the navigation solution is not required. |

### tpp\_process\_barometer

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_barometer(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppBarometerMessageStruct\* barometer\_message\_pointer) |
| Summary | This function is used when a valid barometer data message is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] barometer\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise, the return value is TPP\_INVALID\_BAROMETER\_MESSAGE or TPP\_INVALID\_PROCESS\_BAROMETER\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid barometer data is available. |

### tpp\_process\_magnetometer

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_magnetometer(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppMagnetometerMessageStruct\* magnetometer\_message\_pointer) |
| Summary | This function is used when a valid magnetometer data message is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] magnetometer\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise, the return value is TPP\_INVALID\_MAGNETOMETER\_MESSAGE or TPP\_INVALID\_PROCESS\_MAGNETOMETER\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid magnetometer data is available. |

### tpp\_process\_speed

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_speed(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppSpeedMessageStruct\* speed\_message\_pointer) |
| Summary | This function is used when a valid speed data message is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] speed\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is TPP\_INVALID\_SPEED\_MESSAGE or TPP\_INVALID\_PROCESS\_SPEED\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid speed data is available. |

### tpp\_process\_multiple\_antenna

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_multiple\_antenna(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppMultipleAntennaMessageStruct\* multiple\_antenna\_message\_pointer) |
| Summary | This function is used when a valid multiple antenna data message is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] multiple\_antenna\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is TPP\_INVALID\_MULTIPLE\_ANTENNA\_MESSAGE or TPP\_INVALID\_PROCESS\_MULTIPLE\_ANTENNA\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid multiple antenna data is available. |

### tpp\_process\_operator\_2d\_position

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_operator\_2d\_position(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppOperator2dPositionMessageStruct\* operator\_2d\_position\_message\_pointer) |
| Summary | This function takes a valid 2D position and applies it as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] operator\_2d\_position\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is TPP\_INVALID\_OPERATOR\_2D\_POSITION\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when a valid 2D position update is available as an input from the operator. In this case, the output position will jump to the position provided to this function.  Note that the GNSS position update should not be used with this function but instead it should be used with tpp\_process\_gnss\_pvt() |

### tpp\_process\_wireless

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_wireless (  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppWirelessMessageStruct\* wireless\_message\_pointer) |
| Summary | This function takes valid wireless data and applies it as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] wireless\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is TPP\_INVALID\_WIRELESS\_MESSAGE or TPP\_INVALID\_PROCESS\_WIFI\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly. |

### tpp\_process\_height

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_height(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppHeightMessageStruct\* height\_message\_pointer) |
| Summary | This function takes a valid height and applies it as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] height\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_HEIGHT\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when a valid height is available as an input from any external system or operator. Note that the GNSS height update and barometer height should not be used with this function but instead they should be used with their respective functions. |

### tpp\_process\_call\_information

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_call\_information(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppCallInformationMessageStruct\* call\_information\_message\_pointer) |
| Summary | This function takes the call information (i.e. call mode, proximity detected, headset type, and speaker activity) and applies it as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] call\_information\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_CALL\_INFORMATION\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. If the developer wants to use this function, then it must be called at the highest rate of sensor data which is the IMU data rate. It should be called before tpp\_advance\_navigation\_step() |

### tpp\_process\_floor\_information

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_floor\_information(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppFloorInformationMessageStruct\* floor\_information\_message\_pointer) |
| Summary | This function takes floor information (i.e. floor number and height between floors) from the developer and applies it as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] floor\_information\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_FLOOR\_INFORMATION\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid floor information is available as an input from the operator.  3. A zero value passed to the height\_between\_floors\_ member in the input structure will cause the navigation solution to use the recent value of the height between floors which can be one of (a) a positive value to the height\_between\_floors\_ member in the input structure in a previous call to this function (b) the height between floors specified during initialization by flag\_value\_2\_ in “FLOOR\_INFORMATION” or its default value.  4. A negative value passed to the height\_between\_floors\_ member in the input structure will return TPP\_INVALID\_FLOOR\_INFORMATION |

### tpp\_process\_platform\_heading

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_platform\_heading(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppPlatformHeadingMessageStruct\* platform\_heading\_message\_pointer) |
| Summary | This function takes a valid platform heading and applies it as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] platform\_heading\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code TPP\_INVALID\_PLATFORM\_HEADING\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when a valid platform heading is available as an input from any external system or operator. |

### tpp\_process\_9dof\_quaternions

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_9dof\_quaternions(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppQuaternionsMessageStruct\* quaternions\_message\_pointer) |
| Summary | This function takes a valid 9dof quaternion vector to derive the pitch, roll and heading and applies the derived information as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] quaternions\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code TPP\_INVALID\_QUATERNIONS\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly. |

### tpp\_process\_6dof\_quaternions

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_6dof\_quaternions(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppQuaternionsMessageStruct\* quaternions\_message\_pointer) |
| Summary | This function takes a valid 6dof quaternion vector to derive the pitch, roll and heading and applies the derived information as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] quaternions\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code TPP\_INVALID\_QUATERNIONS\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly. |

### tpp\_process\_imu\_biases

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_imu\_biases(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppImuBiasesMessageStruct\* imu\_biases\_message\_pointer) |
| Summary | This function takes the gyroscope and accelerometer biases and may apply those biases as corrections to the gyroscope and accelerometer data values input to tpp\_advance\_navigation\_step() |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] imu\_biases\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code TPP\_INVALID\_IMU\_BIASES |
| Notes | 1. The function must be used only after the system is initialized properly.  2. The function shall be called before tpp\_advance\_navigation\_step()at the same rate. |

### tpp\_process\_misalignment

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_misalignment(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppProcessMisalignmentMessageStruct\* misalignment\_message\_pointer) |
| Summary | This function takes the misalignment angle and its standard deviation and applies it as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] misalignment\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code TPP\_INVALID\_PROCESS\_MISALIGNMENT\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly. |

### tpp\_process\_venue\_map

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_venue\_map(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppVenueMapMessageStruct\* venue\_map\_message\_pointer) |
| Summary | This function takes venue map related information (such as 2D position, height, platform heading, and the map entity) from the developer and applies it as an update to the navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] venue\_map\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_VENUE\_MAP\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid map information is available. |

### tpp\_set\_mode\_of\_transit

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_set\_mode\_of\_transit(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  TppModeOfTransitEnum mode\_of\_transit\_enum) |
| Summary | This function sets the current mode of transit of the navigator to the one specified as an input. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] mode\_of\_transit\_enum |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_SET\_MODE\_OF\_TRANSIT\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. The function can be used to change the mode of transit, only if the developer has initialized mode of transit for the navigator to be automatic. |

### tpp\_set\_zupt\_mode

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_set\_zupt\_mode(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  TppZuptModeEnum zupt\_mode\_enum) |
| Summary | This function sets the current ZUPT mode of the navigator to the one specified as an input. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] zupt\_mode\_enum |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_SET\_ZUPT\_MODE\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly. |

### tpp\_set\_misalignment

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_set\_misalignment(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppSetMisalignmentMessageStruct\* set\_misalignment\_message\_pointer) |
| Summary | This function enables or disables the automatic misalignment estimation. When the misalignment estimation is turned off, the developer can specify the misalignment angle for the navigator to use. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] set\_misalignment\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_SET\_MISALIGNMENT\_MESSAGE or TPP\_INVALID\_SET\_MISALIGNMENT\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. The function can be used to turn the automatic misalignment estimation on or off, only if the developer has initialized the misalignment for the navigator to be automatic. |

### tpp\_set\_magnetometer\_calibration

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_set\_magnetometer\_calibration(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  TppMagnetometerCalibrationEnum magnetometer\_calibration\_enum) |
| Summary | This function enables or disables the automatic magnetometer calibration. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] magnetometer\_calibration\_enum |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_SET\_MAGNETOMETER\_CALIBRATION\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. The function can be used to enable or disable the magnetometer calibration, only if the magnetometer was initialized to be on. |

### tpp\_set\_use\_case

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_set\_use\_case(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  TppUseCaseEnum use\_case\_enum) |
| Summary | This function sets the use case (such as pocket, hand dangling, or others). |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] use\_case\_enum |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_SET\_USE\_CASE\_CALL |
| Notes | 1. The function must be used only after the system is initialized properly.  2. The function can be used to set the use case, only if the developer has initialized the mode of transit for the navigator to be walking or automatic. |

### tpp\_set\_orientation\_based\_on\_pitch (INTERNAL ONLY)

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_set\_orientation\_based\_on\_pitch(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  TppOrientationBasedOnPitchEnum orientation\_enum) |
| Summary | This function sets the orientation based on pitch angle (Vertical Up, Horizontal, or Vertical Down) |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] orientation\_enum |
| Return Value | If successful, TPP\_SUCCESS is returned. |
| Notes | 1. The function must be used only after the system is initialized properly. |

### tpp\_set\_device\_heading

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_set\_device\_heading(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppDeviceHeadingMessageStruct\* device\_heading\_message\_pointer) |
| Summary | This function takes a valid device angle directly, or computes the device heading from a valid platform heading and a misalignment angle, and sets it for the navigator. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] device\_heading\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code TPP\_INVALID\_DEVICE\_HEADING\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when a valid device heading is available, or a valid platform heading and misalignment angle are available, as an input from any external system or operator. |

### tpp\_add\_anchor\_point

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_add\_anchor\_point(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  Const TppAnchorPointMessageStruct\* anchor\_point\_message\_pointer) |
| Summary | The function adds an anchor point to the output packet. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] ***anchor\_point\_message\_pointer*** |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator*,* TPP\_INVALID\_ANCHOR\_POINT\_PACKET, TPP\_INVALID\_START\_ANCHOR\_POINT *or* TPP\_INVALID\_END\_ANCHOR\_POINT |
| Notes | 1. The function must be used only after the system is initialized properly.  2. The function must be used only when a valid anchor point is available. Those anchor points are needed if the developer is using Invensense Smoothing Library (ISL) along with the TPP library. |

### tpp\_add\_synchronization\_event

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_add\_synchronization\_event(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  TppTypeUint64 event\_number) |
| Summary | The function adds a unique (i.e. within a navigation session) event number to the output packet. This event number should be greater than zero and is mainly used to give the capability to synchronize between datasets generated from multiple systems. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] event\_number |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID \_SYNCHRONIZATION\_EVENT |
| Notes | 1. The function must be used only after the system is initialized properly. |

### tpp\_stop\_navigation

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_stop\_navigation(  TppNavigationSessionHandle tpp\_navigation\_session\_handle) |
| Summary | This function is used to stop navigation |
| Parameters | [IN] tpp\_navigation\_session\_handle |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_STOP\_NAVIGATION\_FAILURE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. The function must be called before the main application exits or before restarting navigation in the same session. |

### tpp\_delete\_navigation\_session

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_delete\_navigation\_session\_handle(  TppNavigationSessionHandle tpp\_navigation\_session\_handle) |
| Summary | The function is used to delete a navigation session handle. |
| Parameters | [IN] tpp\_navigation\_session\_handle |
| Return Value | If successful, a TPP\_SUCCESS is returned |
| Notes | 1. The function can be called after tpp\_stop\_navigation()to reclaim the memory allocated for a specific navigation session. |

### tpp\_prerun\_reset\_zupt\_thresholds

|  |  |
| --- | --- |
| Prototype | void tpp\_prerun\_reset\_zupt\_thresholds() |
| Summary | This function is used to reset the data saved to compute the ZUPT thresholds. |
| Parameters | [NONE] |
| Return Value | [NONE] |
| Notes | 1. The function must be called before calling the other functions that are used to compute the ZUPT thresholds. |

### tpp\_prerun\_compute\_zupt\_thresholds\_imu

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_prerun\_compute\_zupt\_thresholds\_imu(  const TppImuMessageStruct\* imu\_message\_pointer, TppZuptThresholdsStruct\* zupt\_threshold\_struct,  unsigned long\* number\_of\_valid\_samples\_required) |
| Summary | This function is to compute the ZUPT threshold values from IMU samples only. |
| Parameters | [IN] imu\_message\_pointer  [OUT] zupt\_threshold\_struct  [OUT] number\_of\_valid\_samples\_required |
| Return Value | 1. TPP\_ZUPT\_THRESHOLDS\_NOT\_RESET is returned when tpp\_prerun\_reset\_zupt\_thresholds() is not called before calling the current function.  2. TPP\_INVALID\_IMU\_MESSAGE is returned when the 1st parameter passed, imu\_message\_pointer, is a null pointer.  3. TPP\_COMPUTATION\_IN\_PROGRESS is returned if there are no errors, while the function is computing the ZUPT thresholds. The user can refer to the 3rd parameter, number\_of\_valid\_samples\_required, to know the number of samples needed by the function to compute the thresholds.  4. TPP\_SUCCESS is returned when the thresholds are ready. The user can refer to the 2nd parameter, zupt\_threshold\_struct, where the threshold data is filled. |
| Notes | 1. The function is used to compute the ZUPT threshold values from IMU samples only. |

### tpp\_run\_backward\_smoothing

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_run\_backward\_smoothing(  TppNavigationSessionHandle tpp\_navigation\_session\_handle  ) |
| Summary | This function is used to run backward smoothing and generate a file that contains the backward smoothed navigation solution. |
| Parameters | [IN] tpp\_navigation\_session\_handle |
| Return Value | If successful, TPP\_SUCCESS is returned.  Section 7.28 shows all the values that can be returned by the function. |
| Notes | 1. The function can be used only after the **BACKWARD\_SMOOTHING** initialization flag is initialized successfully, with **flag\_value\_1\_** set to “on” and **flag\_value\_2\_** set to a folder path in which the backward smoothed navigation solution is saved.  2. The function must be used only after the system is stopped successfully.  3. A file with the name “**bs\_solution.dat**” containing the backward smoothed navigation solution is generated by this function in the path specified during initialization.  4. “**bs\_solution.dat**” is a *binary* file that contains the backward smoothed navigation solution. An output packet, with the structure specified in section 7.23.1, is present for every IMU sample that was input to **tpp\_advance\_navigation\_step()**. |

### tpp\_process\_gnss\_observation­s

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_gnss\_observations(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppGnssObservationsMessageStruct\* gnss\_obs\_message\_pointer) |
| Summary | This function is used when a valid GNSS observations data message is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] gnss\_obs\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise, the return value is the error code enumerator TPP\_INVALID\_GNSS\_OBSERVATIONS\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid GNSS observations data is available. |

### tpp\_process\_glonass\_ephemeris

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_glonass\_ephemeris(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppGlonassEphemerisMessageStruct\* glonass\_eph\_message\_pointer) |
| Summary | This function is used when a valid GLONASS Ephemeris data message is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] glonass\_eph\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_GLONASS\_EPHEMERIS\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid GLONASS Ephemeris data is available. |

### tpp\_process\_gps\_ephemeris

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_process\_gps\_ephemeris(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  const TppGpsEphemerisMessageStruct\* gps\_eph\_message\_pointer) |
| Summary | This function is used when a valid GPS Ephemeris data message is available. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] gps\_eph\_message\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator TPP\_INVALID\_GPS\_EPHEMERIS\_MESSAGE |
| Notes | 1. The function must be used only after the system is initialized properly.  2. This function must be used only when valid GPS Ephemeris data is available. |

### tpp\_multi\_device\_process\_position\_velocity

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_multi\_device\_process\_position\_velocity(  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  unsigned char number\_of\_devices,  ***const TppMultiDeviceLatencyStruct\* multi\_device\_latency\_pointer,***  ***const*** ***TppMultiDevicePositionVelocityStruct\* multi\_device\_position\_velocity\_pointer,***  ) |
| Summary | The function is used to process the position and velocity information from a number of devices in a multi-device system. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] number\_of\_devices  [IN] ***multi\_device\_latency\_pointer***  [IN] multi\_device\_position\_velocity\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator  TPP\_MULTI\_DEVICE\_INVALID\_NUMBER\_OF\_DEVICES or  TPP\_MULTI\_DEVICE\_INVALID\_LATENCY\_MESSAGE or  TPP\_MULTI\_DEVICE\_INVALID\_POSITION\_VELOCITY\_MESSAGE or |
| Notes | 1. The function must be used only for a multi-device system.  2. The function must be used only after the system is initialized properly.  3. The number of devices should be at least 2 including the information generated on the device processing all the information from other devices in the multi-device system. |

### tpp\_multi\_device\_process\_heading

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_multi\_device\_process\_heading (  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  unsigned char number\_of\_devices,  ***const TppMultiDeviceLatencyStruct\* multi\_device\_latency\_pointer,***  ***const*** ***TppMultiDeviceHeadingStruct\* multi\_device\_heading\_pointer,***  ) |
| Summary | The function is used to process the heading information from a number of devices in a multi-device system. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] number\_of\_devices  [IN] ***multi\_device\_latency\_pointer***  [IN] multi\_device\_heading\_pointer |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise the return value is the error code enumerator  TPP\_MULTI\_DEVICE\_INVALID\_NUMBER\_OF\_DEVICES or  TPP\_MULTI\_DEVICE\_INVALID\_LATENCY\_MESSAGE or  TPP\_MULTI\_DEVICE\_INVALID\_HEADING\_MESSAGE or |
| Notes | 1. The function must be used only for a multi-device system.  2. The function must be used only after the system is initialized properly.  3. The number of devices should be at least 2 including the information generated on the device processing all the information from other devices in the multi-device system. |

### tpp\_multi\_device\_process\_secondary\_gnss\_pvt

|  |  |
| --- | --- |
| Prototype | TppReturnStatusEnum tpp\_multi\_device\_process\_secondary\_gnss\_pvt (  TppNavigationSessionHandle tpp\_navigation\_session\_handle,  signed char navigation\_phase,  const TppGnssPvtMessageStruct\* gnss\_pvt\_message\_pointer***,***  ) |
| Summary | One of the devices in the multi-device system is assumed to share its GNSS data with the rest of the devices in the system (whether they have a GNSS receiver or not). The function is used to process the GNSS data message shared from this secondary device in the system. |
| Parameters | [IN] tpp\_navigation\_session\_handle  [IN] navigation\_phase  [IN] ***gnss\_pvt\_message\_pointer*** |
| Return Value | If successful, TPP\_SUCCESS is returned.  Otherwise, the return value is TPP\_INVALID\_GNSS\_PVT\_MESSAGE or TPP\_INVALID\_PROCESS\_GNSS\_PVT\_CALL |
| Notes | 1. The function must be used only for a multi-device system.  2. The function must be used only after the system is initialized properly.  3. The function must be used only when valid GNSS data is available with a 3D fix status from the GNSS receiver on the secondary device. |

## Data Types

This section defines the data types in the API.

### TppNavigationSessionHandle

Defined as a type definition of pointer to void [typedef void\* TppNavigationSessionHandle]

### TppInitializationStruct

The initialization structures are used during the initialization of the navigator. The main initialization structure is “TppInitializationStruct” and the developer will need to correctly fill a variable of this type by specifying the number of initialization flags and an array of the type “TppInitializationFlagStruct”. An initialization flag is a structure that is composed of six entries. The first entry is the name of the flag while the other five entries represent the values that are related to the flag’s name. Section 8.1 explains how to initialize the navigator with the “TppInitializationStruct” structure.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | number\_of\_initialization\_flags\_ | UINT16 | N/A | Number of initialization flags. |
| 2 | initialization\_flag\_pointer\_ | TppInitializationFlagStruct\* | N/A | Pointer to an array of initialization flags. An entry in the array is of type TppInitializationFlagStruct |

#### TppInitializationFlagStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | flag\_name\_ | CHAR\* | N/A | A pointer to characters that specifies the name of the flag. |
| 2 | flag\_value\_1\_ | CHAR\* | N/A | A pointer of characters that represents the first value related to the flag name. |
| 3 | flag\_value\_2\_ | CHAR\* | N/A | A pointer of characters that represents the second value related to the flag name. |
| 4 | flag\_value\_3\_ | CHAR\* | N/A | A pointer of characters that represents the third value related to the flag name. |
| 5 | flag\_value\_4\_ | CHAR\* | N/A | A pointer of characters that represents the fourth value related to the flag name. |
| 6 | flag\_value\_5\_ | CHAR\* | N/A | A pointer of characters that represents the fifth value related to the flag name. |

Table 5 shows a list of the available flag names, followed by the values that each flag can take.

Table List of Flag Names

|  |  |
| --- | --- |
| **flag\_name\_** | **Description** |
| “[DEVICE](#DEVICE_FLAG)” | Specifies the type of the device; whether it is a phone or a tablet or any other type.  **DEFAULT (If flag name not specified):** {“DEVICE”, “phone”} |
| “[START\_OPTION](#START_OPTION_FLAG)” | Specifies if the navigation solution will  (1) Start without wireless systems and continue to work with the sensors only  (2) Start with reliable data from the wireless systems (i.e. GNSS or Wi-Fi)  (3) Start immediately and accepts the wireless systems as it becomes available.  **DEFAULT (If flag name not specified):** {“START\_OPTION”, “sensors\_reliable\_wireless”} |
| “[MAGNETOMETER](#MAGNETOMETER_FLAG)” | Specifies whether the data from the magnetometer will be used in the navigation solution or not, along with other settings if the magnetometer will be used.  **DEFAULT (If flag name not specified):** {“MAGNETOMETER”, “off”} |
| “[BAROMETER](#BAROMETER)” | Specifies whether the data from the barometer will be used in the navigation solution or not, along with other settings if the barometer will be used.  **DEFAULT (If flag name not specified):** {“BAROMETER”, “off”} |
| “[SPEED](#SPEED_FLAG)” | Specifies whether the data from the speed sensor will be used in the navigation solution or not, along with other settings if the speed sensor will be used.  **DEFAULT (If flag name not specified):** {“SPEED”, “off”} |
| “[GNSS](#GNSS_START_OPTION_FLAG)” | Specifies whether the data from GNSS will be used in the navigation solution or not, along with other settings if GNSS will be used. See also “**PLATFORM\_HEADING**” flag.  **For TMN and TVN:**  **DEFAULT (If flag name not specified):**  {“GNSS”, “on”,“gnss\_precision”,“1.0”,”1.0”}  **For TPN:**  **DEFAULT (If flag name not specified):**  {“GNSS”, “on”,“gnss\_high\_sensitivity”,“1.0”,”1.0”}  **Notes:**  (1) If the developer specified the value for the “**GNSS**” flag to be “**off**”, the developer must use either the “**PLATFORM\_HEADING**” flag to specify the initial heading of the platform or the “**MAGNETOMETER**” flag and set the first flag value to be “**on**”; otherwise, the initialization will not be successful (i.e. TPP\_INVALID\_HEADING\_TO\_START will be returned by tpp\_initialize\_SSS())  (2) If the developer specified the value for the “**GNSS**” flag to be “**off**”, the developer must also use the “**POSITION**” flag to specify the initial position of the platform; otherwise, the initialization will not be successful (i.e. TPP\_INVALID\_POSITION\_TO\_START will be returned by tpp\_initialize\_SSS())  **For TMN and TVN:**  (3) If the developer specified the value of **flag\_value\_1\_** for “**START\_OPTION**” to be “**sensors\_only**”, then **flag\_value\_1\_** for “**GNSS**” flag should be set to “**off**”.  **For TPN:**  (3) If the developer specified the value for the “**GNSS**” flag to be “**off**”, the “**MODE\_OF\_TRANSIT**” flag should be set to “**walking**”; otherwise, the initialization will not be successful (i.e. TPP\_INVALID\_MODE\_OF\_TRANSIT\_TO\_START will be returned by tpp\_initialize\_SSS()).  (4) If the developer specified the value of **flag\_value\_1\_** for “**START\_OPTION**” to be “**sensors\_only**”, then **flag\_value\_1\_** for “**GNSS**” flag should be set to “**off**”, and the “**MODE\_OF\_TRANSIT**” flag should be set to “**walking**” for the initialization to be successful. |
| “[WIFI](#WIFI_FLAG)” | Specifies whether the data from Wi-Fi will be used in the navigation solution or not, along with other settings if Wi-Fi will be used.  **DEFAULT (If flag name not specified):**  {“WIFI”, “off” } |
| “[MULTIPLE\_ANTENNA](#MULTIANTENNA_FLAG)” | Specifies whether the data from the multiple antenna system will be used in the navigation solution or not, along with other settings if multiple antenna system will be used.  **DEFAULT (If flag name not specified):**  {“MULTIPLE\_ANTENNA”, “off”} |
| “[PLATFORM\_HEADING](#PLATFORM_HEADING_FLAG)” | **For TMN and TVN:**  Specifies the initial heading of the platform.  If the developer wants the navigator to start computing the integrated navigation solution without waiting for good GNSS data, the developer can use this flag to specify the initial heading of the platform; this is needed if a calibrated magnetometer is unavailable or the value for the “**MAGNETOMETER**” flag is “**off**”.  **For TPN:**  Specifies the initial heading of the platform.  If the developer specified the value for the “**GNSS**” flag to be “**on**”, this is an optional flag.  For more information, refer to the “[**GNSS**](#GNSS_FLAG_DESCRIPTION)” flag description. |
| “[POSITION](#PLATFORM_POSITION_FLAG)” | **For TMN and TVN:**  Specifies the initial position (Latitude, Longitude, and Height) of the platform.  If the developer wants the navigator to start computing the integrated navigation solution without waiting for good GNSS data, the developer can use this flag to specify the initial position of the platform.  **For TPN:**  Specifies the initial position (Latitude, Longitude, and Height) of the platform.  If the developer specified the value for the “**GNSS**” flag to be “**on**”, this is an optional flag.  For more information, refer to the “[**GNSS**](#GNSS_FLAG_DESCRIPTION)” flag description. |
| “[MISALIGNMENT](#MISALIGNMENT_FLAG)” | Specifies values that are related to the misalignment angle of the device with respect to the platform.  **DEFAULT (If flag name not specified):** {“MISALIGNMENT”, “automatic”, “misalignment\_normal”} |
| “[MODE\_OF\_TRANSIT](#MODE_OF_TRANSIT_FLAG)” | Specifies the mode of transit of the navigator.  **DEFAULT (If flag name not specified):**  {“MODE\_OF\_TRANSIT”, “automatic”}  For more information, refer to the “[**GNSS**](#GNSS_FLAG_DESCRIPTION)” flag description. |
| “[USE\_CASE](#USE_CASE_FLAG)” | Specifies the use case of the navigator.  **DEFAULT (If flag name not specified):**  {“USE\_CASE”, “automatic”} |
| “[FLOOR\_INFORMATION](#FLOOR_INFORMATION_FLAG)” | Specifies the initial floor information.  **DEFAULT (If flag name not specified):** {“FLOOR\_INFORMATION”, “1”, “3.0”} |
| “[GYROSCOPE\_BIASES](#GYROSCOPE_BIASES_FLAG)” | Specifies the initial biases for the gyroscopes.  **DEFAULT (If flag name not specified):** N/A.  The user must specify the gyroscope biases during initialization. Otherwise, the initialization function will return an error. |
| “[ACCELEROMETER\_CONFIGURATION](#ACCELEROMTER_CONFIGURATION)” | Specifies the configuration for the accelerometers.  **For TMN and TVN:**  **DEFAULT (If flag name not specified):**  {“ACCELEROMETER\_CONFIGURATION”, “accelerometer\_5”}  **For TPN:**  **DEFAULT (If flag name not specified):**  {“ACCELEROMETER\_CONFIGURATION”, “accelerometer\_1”} |
| “[GYROSCOPE\_X\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)” | Specifies the configuration for the x-axis gyroscope.  **For TMN and TVN:**  **DEFAULT (If flag name not specified):**  {“GYROSCOPE\_X\_CONFIGURATION”, “gyroscope\_5”}  **For TPN:**  **DEFAULT (If flag name not specified):**  {“GYROSCOPE\_X\_CONFIGURATION”, “gyroscope\_1”} |
| “[GYROSCOPE\_Y\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)” | Specifies the configuration for the y-axis gyroscope.  **For TMN and TVN:**  **DEFAULT (If flag name not specified):**  {“GYROSCOPE\_Y\_CONFIGURATION”, “gyroscope\_5”}  **For TPN:**  **DEFAULT (If flag name not specified):**  {“GYROSCOPE\_Y\_CONFIGURATION”, “gyroscope\_1”} |
| “[GYROSCOPE\_Z\_CONFIGURATION](#GYROSCOPE_X_CONFIGURATION_FLAG)” | Specifies the configuration for the z-axis gyroscope.  **For TMN and TVN:**  **DEFAULT (If flag name not specified):**  {“GYROSCOPE\_Z\_CONFIGURATION”, “gyroscope\_5”}  **For TPN:**  **DEFAULT (If flag name not specified):**  {“GYROSCOPE\_Z\_CONFIGURATION”, “gyroscope\_1”} |
| “[MOTION\_CONSTRAINTS](#MOTION_CONSTRAINTS_FLAG)” | Specifies a configuration of two velocity constraints.  **For TMN and TVN:**  **DEFAULT (If flag name not specified):**  {“MOTION\_CONSTRAINTS”, “motion\_constraints\_4”}  **For TPN:**  **DEFAULT (If flag name not specified):**  {“ MOTION\_CONSTRAINTS”, “motion\_constraints\_1”} |
| “[DEBUG\_DATA](#DEBUG_DATA_FLAG)” | Specifies whether the entities that hold the input data will be generated in the “Output Per Epoch” packet or not. Those entities are mainly used for debugging by Trusted Positioning.  **DEFAULT (If flag name not specified):**  {“ DEBUG\_DATA”, “off”} |
| “[OUTPUT\_POSITION\_ERROR\_WEIGHTING](#OUTPUT_POSITION_ERROR_WEIGHTING_FLAG)” | Specifies the value of weighting for the position standard deviation computed by the navigator.  **DEFAULT (If flag name not specified):**  {“OUTPUT\_POSITION\_ERROR\_WEIGHTING”, “1”, “1”, “1”} |
| “[OUTPUT\_VELOCITY\_ERROR\_WEIGHTING](#OUTPUT_VELOCITY_ERROR_WEIGHTING_FLAG)” | Specifies the value of weighting for the velocity standard deviation computed by the navigator.  **DEFAULT (If flag name not specified):**  {“OUTPUT\_VELOCITY\_ERROR\_WEIGHTING”, “1”, “1”, “1”} |
| “[OUTPUT\_ATTITUDE\_ERROR\_WEIGHTING](#OUTPUT_ATTITUDE_ERROR_WEIGHTING_FLAG)” | Specifies the value of weighting for the attitude standard deviation computed by the navigator.  **DEFAULT (If flag name not specified):**  {“OUTPUT\_ATTITUDE\_ERROR\_WEIGHTING”, “1”, “1”, “1”} |
| “[ACCELEROMETER\_BIASES](#ACCELEROMETER_BIASES_FLAG)” | Specifies the initial biases for the accelerometers.  **DEFAULT (If flag name not specified):**  {“ACCELEROMETER\_BIASES”, “0.0”, “0.0”, “0.0”} |
| “[ZUPT\_THRESHOLDS](#ZUPT_THRESHOLDS_FLAG)” | Specifies the initial ZUPT thresholds.  **DEFAULT**  1.If flag name is not specified and **flag\_value\_1\_** for the “**GNSS**” flag is “**on**”, the default values will {“ZUPT\_THRESHOLDS”, “0.0”, “0.0”, “0.0”}  2. If flag name is not specified and **flag\_value\_1\_** for the “GNSS” flag is “off”, there is no default value and the user must specify the ZUPT thresholds during initialization. Otherwise, the initialization function will return an error. |
| “[DECLINATION\_ANGLE](#DECLINATION_ANGLE_FLAG)” | Specifies the initial declination angle.  **DEFAULT (If flag name not specified):**  {“DECLINATION\_ANGLE”, “0.0” } |
| “[REPLACE\_ATTITUDE\_WITH\_6DOF](#REPLACE_ATTITUDE_WITH_6DOF_FLAG)” | Specifies whether to replace the attitude computation done in the library with 6dof or not  **DEFAULT (If flag name not specified):**  {“REPLACE\_ATTITUDE\_WITH\_6DOF”, “off” } |
| “[PROCESSING\_MODE](#PROCESSING_MODE_FLAG)” | Specifies whether the navigator is processing the data in the forward or backward mode.  **DEFAULT (If flag name not specified):**  {“ PROCESSING\_MODE”, “forward” } |
| “[BACKWARD\_SMOOTHING](#DEBUG_DATA)” | Specifies whether the information required to run the backward smoothing functionality will be saved or not. If the user called **tpp\_run\_backward\_smoothing()** with this information not being saved, the function will return an error.  **DEFAULT (If flag name not specified):**  {“BACKWARD\_SMOOTHING”, “off”} |
| “[LIBRARY\_CALL\_TYPE](#LIBRARY_CALL_TYPE_FLAG)” | Specifies whether the library will be running in the standalone mode or the ISL mode.  DEFAULT (If flag name not specified):  {“LIBRARY\_CALL\_TYPE”, “standalone”} |

|  |  |
| --- | --- |
| **flag\_name\_** | **Description** |
| “CHEST\_MOUNT” | “on” or “off” |
| “NON\_HOLONOMIC” | “on” or “off” |
| “AUTOMATIC\_ZUPT\_IN\_MACHINE” | “on” or “off” |
| “ZUPT” | “on” or “off” |
| “TIGHTLY\_COUPLED” | “on” or “off” |
| ~~“AIRPLANE”~~ | ~~“on” or “off”~~ |
| “CHILDREN\_FILTERS” | “N” |
| ~~“GNSS\_SCALE\_FACTOR”~~ | ~~“F.F”~~ |
| ~~“MAGNETOMETER\_SCALE\_FACTOR”~~ | ~~“F.F”~~ |
| ~~“BAROMETER\_SCALE\_FACTOR”~~ | ~~“F.F”~~ |
| “PEDESTRIAN\_DEAD\_RECKONING” | “F.F” |
| ~~“WIFI”~~ | ~~“F.F”~~ |
| “OLD\_DATASET” | “on” or “off” |
| “SENSORS\_RATE” | “N” |
| “GNSS\_RATE” | “N” |
| ~~“IMU\_SYSTEM\_NAME”~~ |  |
| ~~“GYROSCOPE\_BIASES”~~ | ~~“F.F”, “F.F”, “F.F”~~ |
| “ACCELEROMETER\_BIASES” | “F.F”, “F.F”, “F.F” |
| “GNSS\_REJECTION” | “on” or “off” |

#### DEVICE flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “DEVICE” | |
| **flag\_value\_1\_** | “phone” or  “tablet” or  “head\_mount” or  “watch” | “phone” specifies to the navigator that the device is a phone.  “tablet” specifies to the navigator that the device is a tablet.  “head\_mount” specifies to the navigator that the device is head-mounted such as glasses.  “watch” specifies to the navigator that the device is a watch. |

#### START\_OPTION flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “START\_OPTION” | |
| **flag\_value\_1\_** | “sensors\_only”  or “sensors\_reliable\_wireless”  or  “sensors\_wireless” | “sensors\_only” specifies that only the data from the sensors will be used in the navigation solution.  If **flag\_value\_1\_** for **GNSS** or **WIFI** flags is “on”, the initialization function will not be successful (i.e. TPP\_INVALID\_OPTION\_SENSORS\_ONLY will be returned by tpp\_initialize\_SSS())  “sensors\_reliable\_wireless” specifies that the navigator will start with reliable wireless systems (GNSS or Wi-Fi or both) and the overall solution will be sensor/wireless integrated.  The navigator will provide a reliable navigation solution only when reliable 2D position (latitude and longitude) is provided from the GNSS or the Wi-Fi systems to the navigator via tpp\_process\_gnss\_pvt() or tpp\_process\_2d\_position(). A reliable 2D position is indicated by the standard deviation of the 2D position being less than 5 meters.  If **flag\_value\_1\_** for both **GNSS** and **WIFI** flags is “off”, the initialization function will not be successful (i.e. TPP\_INVALID\_OPTION\_SENSORS\_RELIABLE\_WIRELESS will be returned by tpp\_initialize\_SSS())  “sensors\_wireless” specifies that the navigator will start with or without wireless systems (GNSS or Wi-Fi or both). The navigator will provide a reliable navigation solution without waiting for a reliable 2D position from wireless systems. When a reliable 2D position is available, the navigator will utilize it in the navigation solution.  If **flag\_value\_1\_** for both **GNSS** and **WIFI** flags is “off”, the initialization function will not be successful (i.e. TPP\_INVALID\_OPTION\_SENSORS\_WIRELESS will be returned by tpp\_initialize\_SSS()).  If “sensors\_wireless” is used and the mode of transit is set to driving, the behavior of the navigator will be equivalent to “sensors\_reliable\_wireless” |

#### MAGNETOMETER flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “MAGNETOMETER” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies that the data from the magnetometer will be used in the navigation solution.  “off” specifies that the data from the magnetometer will not be used in the navigation solution. |
| **flag\_value\_2\_** | “F.F” [[1]](#footnote-1) | “F.F” specifies the value of weighting for the magnetometer standard deviation. This value must be provided if the developer specified **flag\_value\_1\_** to be “on”. In order not to change the default behavior of the navigator, the developer can use “1.0”. |

#### BAROMETER flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “BAROMETER” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies that the data from the barometer will be used in the navigation solution.  “off” specifies that the data from the barometer will not be used in the navigation solution. |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the value of weighting for the barometer standard deviation. This value must be provided if the developer specified **flag\_value\_1\_** to be “on”. In order not to change the default behavior of the navigator, the developer can use “1.0”. |

#### SPEED flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “SPEED” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies that the data from the speed sensor will be used in the navigation solution.  “off” specifies that the data from the speed sensor will not be used in the navigation solution. |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the value of weighting for the speed standard deviation. This value must be provided if the developer specified **flag\_value\_1\_** to be “on”. In order not to change the default behavior of the navigator, the developer can use “1.0”. |

#### GNSS flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “GNSS” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies that the data from the GNSS receiver will be used in the navigation solution.  “off” specifies that the data from the GNSS receiver will not be used in the navigation solution. |
| **flag\_value\_2\_** | PREDEFINED\_STRING | This value specifies the receiver type. This value must be provided if the value of the first flag is “on”.  The values that PREDEFINED\_STRING can take are defined and explained in Section 13.1. |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the value of weighting for the GNSS position standard deviation. This value must be provided if the developer specified **flag\_value\_1\_** to be “on”. In order not to change the default behavior of the navigator, the developer can use “1.0”. |
| **flag\_value\_4\_** | “F.F” | “F.F” specifies the value of weighting for the GNSS velocity standard deviation. This value must be provided if the developer specified **flag\_value\_1\_** to be “on”. In order not to change the default behavior of the navigator, the developer can use “1.0”. |

#### WIFI flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “WIFI” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies that the data from Wi-Fi will be used in the navigation solution.  “off” specifies that the data from Wi-Fi will not be used in the navigation solution. |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the value of weighting for the Wi-Fi standard deviation. This value must be provided if the developer specified **flag\_value\_1\_** to be “on”. In order not to change the default behavior of the navigator, the developer can use “1.0”. |

#### MULTIPLE\_ANTENNA flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “MULTIPLE\_ANTENNA” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies that the data from the multiple antenna will be used in the navigation solution.  “off” specifies that the data from the multiple antenna will not be used in the navigation solution. |

#### PLATFORM\_HEADING flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “PLATFORM\_HEADING” | |
| **flag\_value\_1\_** | “F.F” | “F.F” specifies the value of the initial platform heading in degrees that will be passed to the navigator. |

#### POSITION flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “POSITION” | |
| **flag\_value\_1\_** | “F.F” | “F.F” specifies the value of the initial latitude in degrees that will be passed to the navigator. |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the value of the initial longitude in degrees that will be passed to the navigator. |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the value of the initial height in metres that will be passed to the navigator. |

#### MISALIGNMENT flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “MISALIGNMENT” | |
| **flag\_value\_1\_** | “fixed” or “automatic” | “fixed” specifies that the misalignment angle set in the next field will be used in the navigation solution.  “automatic” specifies that the misalignment angle will be automatically calculated by the navigator. |
| **flag\_value\_2\_** | “F.F” if **flag\_value\_1**\_ is “fixed”  “misalignment\_normal” or “misalignment\_snap” if **flag\_value\_1\_** is “automatic” | **If flag\_value\_1\_ is “fixed”:**  “F.F” specifies the value of the misalignment angle in degrees that will be passed to the system if the developer specified **flag\_value\_1\_** to be “fixed”.  **If flag\_value\_1\_ is “automatic”:**  “misalignment\_normal” specifies that the misalignment estimation operates in normal mode without any approximations to the calculated angles.  “misalignment\_snap” specifies that the misalignment is snapped to specific quantized or discrete angles in handheld use cases. |

#### MODE\_OF\_TRANSIT flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “MODE\_OF\_TRANSIT” | |
| **flag\_value\_1\_** | “walking” or  “driving” or  “automatic” or  “running” or  “cycling” | “walking” specifies to the navigator to work in walking mode only.  “driving” specifies to the navigator to work in driving mode only.  “automatic” specifies to the navigator to work in automatic mode where it detects the mode of transit automatically.  “running” specifies to the navigator to work in running mode only.  “cycling” specifies to the navigator to work in cycling mode only. |

#### USE\_CASE flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “MODE\_OF\_TRANSIT” | |
| **flag\_value\_1\_** | “automatic” or  “torso\_and\_hand\_viewing” or  “pocket” or  “hand\_swinging” or  “arm” | “automatic” Specifies to the navigator to automatically detect the use case.  “torso\_and\_hand\_viewing” specifies to the navigator that the device is  “pocket” specifies to the navigator that the device is in pocket  “hand\_dangling” specifies to the navigator that the device is dangling in hand.  “arm” specifies to the navigator that the device is on the arm |

#### FLOOR\_INFORMATION flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “FLOOR\_INFORMATION” | |
| **flag\_value\_1\_** | “N”[[2]](#footnote-2) | “N” represents an integer which specifies the initial floor number. |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the height between floors in metres. |

#### GYROSCOPE\_BIASES flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “GYROSCOPE\_BIASES” | |
| **flag\_value\_1\_** | “F.F” | “F.F” specifies the bias value for the **X**-axis gyroscope in deg/sec |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the bias value for the **Y**-axis gyroscope in deg/sec |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the bias value for the **Z**-axis gyroscope in deg/sec |

#### ACCELEROMETER\_CONFIGURATION flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “ACCELEROMETER\_CONFIGURATION” | |
| **flag\_value\_1\_** | “accelerometer\_custom” or  PREDEFINED\_STRING | “accelerometer\_custom” specifies that a custom accelerometer configuration will be used and the following three flags must be filled accordingly.  The values that PREDEFINED\_STRING can take are defined and explained in Section 13.2. |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the bias instability value for the three accelerometers in mGal |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the velocity random walk (VRW) value for the three accelerometers in (m/sec/√hr) |
| **flag\_value\_4\_** | “F.F” | “F.F” specifies the bias correlation time value for three accelerometers in hr |

#### GYROSCOPE\_X\_CONFIGURATION flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “GYROSCOPE\_X\_CONFIGURATION” | |
| **flag\_value\_1\_** | “gyroscope\_custom” or  PREDEFINED\_STRING | “gyroscope\_custom” specifies that a custom gyroscope configuration will be used and the following three flags must be filled accordingly.  The values that PREDEFINED\_STRING can take are defined and explained in Section 13.3 |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the bias instability value for the **X**-axis gyroscope in deg/hr |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the angular random walk (ARW) value for the **X**-axis gyroscope in deg/√hr |
| **flag\_value\_4\_** | “F.F” | “F.F” specifies the bias correlation time value for the **X**-axis gyroscope in hr |

#### GYROSCOPE\_Y\_CONFIGURATION flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “GYROSCOPE\_Y\_CONFIGURATION” | |
| **flag\_value\_1\_** | “gyroscope\_custom” or  PREDEFINED\_STRING | “gyroscope\_custom” specifies that a custom gyroscope configuration will be used and the following three flags must be filled accordingly.  The values that PREDEFINED\_STRING can take are defined and explained in Section 13.3 |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the bias instability value for the **Y**-axis gyroscope in deg/hr |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the angular random walk (ARW) value for the **Y**-axis gyroscope in deg/√hr |
| **flag\_value\_4\_** | “F.F” | “F.F” specifies the bias correlation time value for the **Y**-axis gyroscope in hr |

#### GYROSCOPE\_Z\_CONFIGURATION flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “GYROSCOPE\_Z\_CONFIGURATION” | |
| **flag\_value\_1\_** | “gyroscope\_custom” or  PREDEFINED\_STRING | “gyroscope\_custom” specifies that a custom gyroscope configuration will be used and the following three flags must be filled accordingly.  The values that PREDEFINED\_STRING can take are defined and explained in Section 13.3 |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the bias instability value for the **Z**-axis gyroscope in deg/hr |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the angular random walk (ARW) value for the **Z**-axis gyroscope in deg/√hr |
| **flag\_value\_4\_** | “F.F” | “F.F” specifies the bias correlation time value for the **Z**-axis gyroscope in hr |

#### MOTION\_CONSTRAINTS flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “MOTION\_CONSTRAINTS” | |
| **flag\_value\_1\_** | “motion\_constraints\_custom” or  PREDEFINED\_STRING | “motion\_constraints\_custom” specifies that a custom motion constraints configuration will be used and the following two flags must be filled accordingly.  The values that PREDEFINED\_STRING can take are defined and explained in Section 0 |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the dynamic velocity constraint in m/sec. This value must be greater than zero. |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the static velocity constraint in m/sec. This value must be greater than zero. |

#### DEBUG\_DATA flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “DEBUG\_DATA” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies that the data entities used in debugging will be generated in the “Output Per Epoch” packet.  “off” specifies that no debug entities will be generated in the “Output Per Epoch” packet. |

#### OUTPUT\_POSITION\_ERROR\_WEIGHTING flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “OUTPUT\_POSITION\_ERROR\_WEIGHTING” | |
| **flag\_value\_1\_** | “N” | “N” specifies the value of weighting for the latitude standard deviation. The weighting can be any integer value from 1 to 255. |
| **flag\_value\_2\_** | “N” | “N” specifies the value of weighting for the longitude standard deviation. The weighting can be any integer value from 1 to 255. |
| **flag\_value\_3\_** | “N” | “N” specifies the value of weighting for the height standard deviation. The weighting can be any integer value from 1 to 255. |

#### OUTPUT\_VELOCITY\_ ERROR\_WEIGHTING flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “OUTPUT\_VELOCITY\_ERROR\_WEIGHTING” | |
| **flag\_value\_1\_** | “N” | “N” specifies the value of weighting for the north velocity standard deviation. The weighting can be any integer value from 1 to 255. |
| **flag\_value\_2\_** | “N” | “N” specifies the value of weighting for the east velocity standard deviation. The weighting can be any integer value from 1 to 255. |
| **flag\_value\_3\_** | “N” | “N” specifies the value of weighting for the down velocity standard deviation. The weighting can be any integer value from 1 to 255. |

#### OUTPUT\_ATTITUDE\_ ERROR\_WEIGHTING flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “OUTPUT\_ATTITUDE\_ERROR\_WEIGHTING” | |
| **flag\_value\_1\_** | “N” | “N” specifies the value of weighting for the roll standard deviation. The weighting can be any integer value from 1 to 255. |
| **flag\_value\_2\_** | “N” | “N” specifies the value of weighting for the pitch standard deviation. The weighting can be any integer value from 1 to 255. |
| **flag\_value\_3\_** | “N” | “N” specifies the value of weighting for the heading standard deviation. The weighting can be any integer value from 1 to 255. |

#### ACCELEROMETER\_BIASES flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “ACCELEROMETER\_BIASES” | |
| **flag\_value\_1\_** | “F.F” | “F.F” specifies the bias value for the **X**-axis accelerometer in m/sec/sec |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the bias value for the **Y**-axis accelerometer in m/sec/sec |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the bias value for the **Z**-axis accelerometer in m/sec/sec |

#### ZUPT\_THRESHOLDS flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “ZUPT\_THRESHOLDS” | |
| **flag\_value\_1\_** | “F.F” | “F.F” specifies the ZUPT threshold for the **X**-axis |
| **flag\_value\_2\_** | “F.F” | “F.F” specifies the ZUPT threshold for the **Y**-axis |
| **flag\_value\_3\_** | “F.F” | “F.F” specifies the ZUPT threshold for the **Z**-axis |

#### DECLINATION\_ANGLE flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “DECLINATION\_ANGLE” | |
| **flag\_value\_1\_** | “F.F” | “F.F” specifies the declination angle in degrees within the range [-180,180] |

#### REPLACE\_ATTITUDE\_WITH\_6DOF flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “REPLACE\_ATTITUDE\_WITH\_6DOF” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies that the 6dof will be used to compute the attitude  “off” specifies that the attitude computations done in the library will be used instead of the 6dof |

#### PROCESSING\_MODE flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “PROCESSING\_MODE” | |
| **flag\_value\_1\_** | “forward” or “backward” | “forward” specifies that the navigator is processing the data in forward mode  “backward” specifies that the navigator is processing the data in backward mode |

#### BACKWARD\_SMOOTHING flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “BACKWARD\_SMOOTHING” | |
| **flag\_value\_1\_** | “on” or “off” | “on” specifies to the navigator to save specific information needed to run backward smoothing. In this case, the user should set **flag\_value\_2\_** with a folder path where a file containing the backward smoothed solution will be generated.  “off” specifies to the navigator not to save any information for the backward smoothing functionality. If the user calls **tpp\_run\_backward\_smoothing()** at the end of the navigation session, with the value of this flag being “off”, an error will be returned. |
| **flag\_value\_2\_** | FOLDER\_PATH | The path of the folder where a file containing the backward smoothed solution is generated. The name of the generated file is “**bs\_solution.dat**”. Temporary files are also generated which contain information needed to run backward smoothing. The application using the library should have write access to the specified location on the storage device. |

#### LIBRARY\_CALL\_TYPE flag values

|  |  |  |
| --- | --- | --- |
| **flag\_name\_** | “LIBRARY\_CALL\_TYPE” | |
| **flag\_value\_1\_** | “standalone” or “isl” | “standalone” specifies that the TPN library will be running in the standalone mode. This is the default setting if the flag does not exist.  “isl” specifies that the TPN library will be running in Invensense Smoothing Library (ISL) mode. In this case, TPN library will work together with ISL to output enhanced navigation results. |

### TppApiVersionStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | major\_ | UINT8 | N/A | API major version |
| 2 | minor\_ | UINT8 | N/A | API minor version |
| 3 | patch\_ | UINT8 | N/A | API patch version |
| 4 | release\_id\_ | UINT8 | N/A | API release id  0x00: Alpha  0x01: Beta  0x02: Release Candidate  0x03: Ready for Production Release |
| 5 | release\_number\_ | UINT8 | N/A | API release number. The number should be greater than 0. This number is only valid when the API release is under development (i.e. Not a “Ready for Production” Release) |
| 6 | type\_ | UINT8 | N/A | ‘F’: Full  ‘M’: Micro |
| 7 | set\_number\_ | UINT16 | N/A | Library set number. A unique number that identifies the library used. |

The version number that is used to track the API/Library changes is as follows:

Major**.**Minor**.**Patch**-**ReleaseId**.**ReleaseNumber

**Examples**:

(1) 2.0.0-rc.4: 4th release candidate version of the 2.0.0 library

(2) 3.0.2: Ready for production 3.0.2 version of the library. Ready for production releases are specified by the release id being 0x03 and in that case, the release number is discarded.

### TppGnssPvtMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_timetag\_ | DOUBLE64 | sec | An absolute time-tag that increments according to the rate of the GNSS data provided. GPS time or UTC time can be used to fill this field, but it has to be consistent throughout the navigation session. |
| 2 | latitude\_ | DOUBLE64 | deg |  |
| 3 | longitude\_ | DOUBLE64 | deg |  |
| 4 | height\_ | FLOAT32 | m |  |
| 5 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 6 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 7 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 8 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 10 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 11 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 12 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 13 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 14 | dop\_data\_available\_ | UINT8 | N/A | 0x00: DOP data is not available  0x01: DOP data is available  If this field is set to 0x01, then the following fields (**horizontal\_dop\_** and **vertical\_dop\_**) should be filled accordingly. |
| 15 | horizontal\_dop\_ | FLOAT32 | N/A |  |
| 16 | vertical\_dop\_ | FLOAT32 | N/A |  |

### TppImuMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | timetag\_ | DOUBLE64 | sec | time-tag |
| 2 | data\_gyroscope\_x\_ | DOUBLE64 | deg/sec |  |
| 3 | data\_gyroscope\_y\_ | DOUBLE64 | deg/sec |  |
| 4 | data\_gyroscope\_z\_ | DOUBLE64 | deg/sec |  |
| 5 | data\_accelerometer\_x\_ | DOUBLE64 | m/sec/sec |  |
| 6 | data\_accelerometer\_y\_ | DOUBLE64 | m/sec/sec |  |
| 7 | data\_accelerometer\_z\_ | DOUBLE64 | m/sec/sec |  |

### TppBarometerMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | height\_ | FLOAT32 | m |  |
| 2 | height\_standard\_deviation\_ | FLOAT32 | m |  |

**Notes:**

1. In case the pressure is available instead of the height, the developer can compute the height (in metres) above sea level from the pressure (in mBar) according to the below equation:

1. The height provided in the TppBarometerMessageStruct can be either “height above sea level” or “height above ellipsoid” through the navigation session without being consistent with the type of the height input provided to the other API functions.
2. To get the height standard deviation, the developer can collect static data from the barometer in the normal room temperature and compute the standard deviation for the data collected which will be the value used for the member height\_standard\_deviation\_. If the height standard deviation is not available, then the field should be filled by 1.0.

### TppMagnetometerMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | raw\_data\_available\_ | UINT8 | N/A | 0x00: raw data is not available  0x01: raw data is available |
| 2 | raw\_data\_x\_ | FLOAT32 | mG |  |
| 3 | raw\_data\_y\_ | FLOAT32 | mG |  |
| 4 | raw\_data\_z\_ | FLOAT32 | mG |  |
| 5 | raw\_data\_accuracy\_flag\_ | UINT8 | N/A | 0x00: Flag not available  0x01: Invalid data  0x02: Valid data |
| 6 | calibrated\_data\_available\_ | UINT8 | N/A | 0x00: calibrated data is not available  0x01: calibrated data is available |
| 7 | calibrated\_data\_x\_ | FLOAT32 | mG |  |
| 8 | calibrated\_data\_y\_ | FLOAT32 | mG |  |
| 9 | calibrated\_data\_z\_ | FLOAT32 | mG |  |
| 10 | calibrated\_data\_accuracy\_flag\_ | UINT8 | N/A | 0x00: Flag not available  0x01:Unreliable  0x02: Low Accuracy  0x03: Medium Accuracy  0x04: High Accuracy |
| 11 | calibration\_status\_changed\_flag\_ | UINT8 | N/A | Specifies if the status of the calibration changed when the magnetometer providing the calibrated data is reset or when the the calibrated signals are saturated.  0x00: Flag not available  0x01: No change in status  0x02: Calibration status changed |
| 12 | heading\_available\_ | UINT8 | N/A | 0x00: heading data is not available  0x01: heading data is available |
| 13 | heading\_ | FLOAT32 | deg | The heading is with respect to the True North (i.e. The heading provided is expected to be corrected with the magnetic declination angle) and not the Magnetic North. |
| 14 | heading\_standard\_deviation\_ | FLOAT32 | deg | See notes. |

**Notes:**

1. If the heading standard deviation is not available, then the field should be filled by 12.0

### TppSpeedMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | speed\_ | FLOAT32 | m/sec |  |
| 2 | reserved\_1\_ | UINT8 | N/A |  |
| 2 | reverse\_ | UINT8 | N/A | 0x00: Unavailable  0x01:Forward  0x02: Reverse |

### TppMultipleAntennaMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | heading\_ | FLOAT32 | deg |  |
| 2 | heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 3 | roll\_ | FLOAT32 | deg |  |
| 4 | roll\_standard\_deviation\_ | FLOAT32 | deg |  |
| 5 | pitch\_ | FLOAT32 | deg |  |
| 6 | pitch\_standard\_deviation\_ | FLOAT32 | deg |  |

### TppOperator2dPositionMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | latitude\_ | DOUBLE64 | deg |  |
| 2 | longitude\_ | DOUBLE64 | deg |  |

### TppWirelessMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | source\_ | UINT8 | N/A | 0x01: Bluetooth  0x11: Wi-Fi Source (1)  0x12: Wi-Fi Source (2) |
| 2 | latency\_available\_ | UINT8 | N/A | 0x00: Latency and the corresponding standard deviation are not available.  0x01: Latency and the corresponding standard deviation are available. |
| 3 | latency\_ | FLOAT32 | sec | Delay between the time the Wi-Fi 2D position is requested and the time the position is received and sent to the navigator.  If there is no delay, the user can use zero. |
| 4 | latency\_standard\_deviation\_ | FLOAT32 | sec |  |
| 5 | position\_2d\_available\_ | UINT8 | N/A | 0x00: 2d position and the corresponding standard deviation are not available  0x01: 2d position and the corresponding standard deviation are available |
| 6 | latitude\_ | DOUBLE64 | deg |  |
| 7 | longitude\_ | DOUBLE64 | deg |  |
| 8 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 10 | height\_available\_ | UINT8 | N/A | 0x00: Height and the corresponding standard deviation are not available  0x01: Height and the corresponding standard deviation are available |
| 11 | height\_ | FLOAT32 | m |  |
| 12 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 13 | velocity\_2d\_available\_ | UINT8 | N/A | 0x00: 2d velocity and the corresponding standard deviation are not available  0x01: 2d velocity and the corresponding standard deviation are available |
| 14 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 15 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 16 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 17 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 18 | velocity\_down\_available\_ | UINT8 | N/A | 0x00: Down velocity and the corresponding standard deviation are not available  0x01: Down velocity and the corresponding standard deviation are available |
| 19 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 20 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 21 | platform\_heading\_available\_ | UINT8 | N/A | 0x00: Platform heading and the corresponding standard deviation are not available  0x01: Platform heading and the corresponding standard deviation are available |
| 22 | platform\_heading\_ | FLOAT32 | deg |  |
| 23 | platform\_heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 24 | floor\_information\_available\_ | UINT8 | N/A | 0x00: Floor information is not available  0x01: Floor information is available |
| 24 | floor\_number\_ | INT16 | N/A |  |

### TppHeightMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | height\_ | FLOAT32 | m |  |
| 2 | height\_standard\_deviation\_ | FLOAT32 | m |  |

### TppFloorInformationMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | floor\_number\_ | INT16 | N/A |  |
| 2 | height\_between\_floors\_ | FLOAT32 | m |  |

### TppCallInformationMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | is\_call\_active\_ | UINT8 | N/A | 0x00: Call Inactive  0x01: Call Active |
| 2 | is\_proximity\_detected\_ | UINT8 | N/A | 0x00: No Proximity Detected  0x01: Proximity Detected |
| 3 | is\_speaker\_active\_ | UINT8 | N/A | 0x00: Phone speaker Inactive  0x01: Phone speaker Active |
| 4 | headset\_type\_ | UINT8 | N/A | 0x00: None  0x01: Wired or wireless Earphone is used |

### TppPlatformHeadingMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | heading\_ | FLOAT32 | deg |  |
| 2 | heading\_standard\_deviation\_ | FLOAT32 | deg |  |

### TppDeviceHeadingMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | device\_heading\_available\_ | UINT8 | N/A | **0x00:** Device heading value is not available directly and will be computed from the **platform\_heading\_** and **misalignment\_angle\_** fields.  **0x01**: Device heading value is available directly from the **device\_heading\_** field. |
| 2 | device\_heading\_ | FLOAT32 | deg |  |
| 3 | platform\_heading\_ | FLOAT32 | deg |  |
| 4 | misalignment\_angle\_ | FLOAT32 | deg |  |

### TppQuaternionsMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | value\_0\_ | FLOAT32 | N/A | The scalar component of the quaternion. |
| 2 | value\_1\_ | FLOAT32 | N/A | The 1st vector component of the quaternion. |
| 3 | value\_2\_ | FLOAT32 | N/A | The 2nd vector component of the quaternion. |
| 4 | value\_3\_ | FLOAT32 | N/A | The 3rd vector component of the quaternion. |

### TppImuBiasesMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | bias\_gyroscope\_source\_ | INT8 | N/A | -1: Biases Not Available (Discard values)  **Other values to be defined by ISJ and ICA** |
| 2 | bias\_gyroscope\_accuracy\_ | FLOAT32 | deg/sec | Accuracy for gyroscope biases |
| 3 | bias\_gyroscope\_x\_ | FLOAT32 | deg/sec | Bias for gyroscope-x |
| 4 | bias\_gyroscope\_y\_ | FLOAT32 | deg/sec | Bias for gyroscope-y |
| 5 | bias\_gyroscope\_z\_ | FLOAT32 | deg/sec | Bias for gyroscope-z |
| 6 | bias\_accelerometer\_source\_ | INT8 | N/A | -1: Biases Not Available (Discard values)  **Other values to be defined by ISJ and ICA** |
| 7 | bias\_accelerometer\_accuracy\_ | FLOAT32 | m/sec/sec | Accuracy for accelerometer biases |
| 8 | bias\_accelerometer\_x\_ | FLOAT32 | m/sec/sec | Bias for accelerometer-x |
| 9 | bias\_accelerometer\_y\_ | FLOAT32 | m/sec/sec | Bias for accelerometer-y |
| 10 | bias\_ accelerometer\_z\_ | FLOAT32 | m/sec/sec | Bias for accelerometer-z |

### TppVenueMapMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | position\_2d\_available\_ | UINT8 | N/A | 0x00: 2d position and the corresponding standard deviation are **not available**  0x01: 2d position and the corresponding standard deviation are **available but not reliable**  0x02: 2d position and the corresponding standard deviation are **available and reliable** |
| 2 | latitude\_ | DOUBLE64 | deg |  |
| 3 | longitude\_ | DOUBLE64 | deg |  |
| 4 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 5 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 6 | height\_available\_ | UINT8 | N/A | 0x00: Height and the corresponding standard deviation are not available  0x01: Height and the corresponding standard deviation are available |
| 7 | height\_ | FLOAT32 | m |  |
| 8 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | platform\_heading\_available\_ | UINT8 | N/A | 0x00: Platform heading and the corresponding standard deviation are **not available**  0x01: Platform heading and the corresponding standard deviation are **available but not reliable**  0x01: Platform heading and the corresponding standard deviation are **available and reliable** |
| 10 | platform\_heading\_ | FLOAT32 | deg |  |
| 11 | platform\_heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 12 | map\_entity\_available\_ | UINT8 | N/A | 0x00: Map entity is not available  0x01: Map entity is available |
| 13 | map\_entity\_ | UINT16 | N/A | Specifies an entity in the map. Examples for map entities are stairs, elevators, escalators, types of rooms, and others. |
| 15 | step\_length\_scale\_available\_ | UINT8 | N/A | Flag to notify if both fields (#16) Step Length Scale and (#17) Use case for Step length scale are available or not.  0x00: Fields are not available  0x01: Fields are available |
| 16 | step\_length\_scale\_ | FLOAT32 | N/A |  |
| 17 | use\_case\_for\_step\_length\_scale\_ | UINT8 | N/A |  |
| 18 | mode\_of\_motion\_for\_step\_length\_scale\_ | UINT8 | N/A |  |

### TppProcessMisalignmentMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | angle\_ | FLOAT32 | deg |  |
| 2 | angle\_standard\_deviation\_ | FLOAT32 | deg |  |

### TppSetMisalignmentMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | estimation\_on\_ | UINT8 | N/A | 0x00: Automatic misalignment estimation turned off  0x01: Automatic misalignment estimation turned on |
| 2 | angle\_available\_ | UINT8 | N/A | 0x00: Misalignment angle is not available  0x01: Misalignment angle is available  When the misalignment estimation is turned off, the user can specify a fixed misalignment angle by setting this field to ‘1’ and setting the angle in the **angle\_** field.  When the misalignment estimation is turned off, and if the user specified this field to be ‘0’, then the navigator will use the last estimated misalignment. |
| 3 | angle\_ | FLOAT32 | deg |  |

### TppSolutionStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | navigation\_phase\_ | INT8 | N/A | -1: Pre-alignment  0: Alignment  +1: Navigation/Available  +2: Navigation/Reliable  +3: Navigation/Vertical Alignment  +4: Navigation/Drive to Walk |
| 2 | timetag\_ | DOUBLE64 | sec |  |
| 3 | latitude\_ | DOUBLE64 | deg |  |
| 4 | longitude\_ | DOUBLE64 | deg |  |
| 5 | height\_ | FLOAT32 | m |  |
| 6 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 7 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 8 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 10 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 11 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 12 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 13 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 14 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 15 | roll\_ | FLOAT32 | deg |  |
| 16 | pitch\_ | FLOAT32 | deg |  |
| 17 | heading\_ | FLOAT32 | deg | Device heading |
| 18 | roll\_standard\_deviation\_ | FLOAT32 | deg |  |
| 19 | pitch\_standard\_deviation\_ | FLOAT32 | deg |  |
| 20 | heading\_standard\_deviation\_ | FLOAT32 | deg | Device heading standard deviation |
| 21 | platform\_heading\_ | FLOAT32 | deg | This field is equivalent to the **heading\_** field |

### TppPacketStruct

Variable of this data type is used in the parameter list of two functions: tpp\_initialize\_SSS() and tpp\_advance\_navigation\_step()to return a packed byte array and its valid size.

The structure of this byte array is dependent on which of the functions is used to populate the array.

The packed byte array which is populated when tpp\_initialize\_SSS() is called, is used to enable Trusted Positioning debugging. In this case, the maximum size that should be allocated for the array which the structure member byte\_array\_pointer\_ points to is **550** bytes.

The structure of the byte array which is populated when tpp\_advance\_navigation\_step() is called is presented in Section 7.23.1. In this case, the maximum size that should be allocated for the array which the structure member byte\_array\_pointer\_ points to is **1500** bytes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | byte\_array\_pointer\_ | UINT8\* | N/A | Pointer to the byte array. |
| 2 | returned\_byte\_array\_length\_ | UINT16 | N/A | This field specifies the exact used length of the byte array. This value will be used by the developer when  (1) Writing the byte array directly to a dataset file which will be used for Trusted Positioning debugging.  Or (2) Extracting specific entities from the output byte array. |

#### Output Array Structure

The structure of the byte array that is populated by tpp\_advance\_navigation\_step()is shown below. The entities, their identification numbers and structures are specified in Section 9.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Units** | **Description** |
| 1 | synchronization\_byte\_a\_ | UINT8 | 0x54 | 1st Synchronization Byte |
| 2 | synchronization\_byte\_b\_ | UINT8 | 0x50 | 2nd Synchronization Byte |
| 3 | synchronization\_byte\_c\_ | UINT8 | 0x49 | 3rd Synchronization Byte |
| 4 | payload\_length\_ | UINT16 | N/A | Length of the payload (i.e. packet body) attached to the packet header excluding the packet checksum. |
| 5 | packet\_id\_ | UINT16 | N/A | **0x0003** |
| 6 | stream\_id\_ | UINT16 | N/A | Identifies the stream identification number of the current epoch if more than one output stream exists |
| 7 | epoch\_number\_ | UINT32 | N/A | Number of the current epoch |
| 8 | number\_of\_entities\_ | UINT16 | N/A | Number of entities (**E**) within the current epoch. The maximum number of entities is 65,535. Each entity in the packet has two fields: entity\_id\_ and entity\_start\_address\_ |
| 9 | entity\_id\_(1) | UINT16 | N/A | The Identification number for the 1st entity in the payload. |
| 10 | entity\_start\_address\_(1) | UINT16 | N/A | The start address of the 1st entity |
| - | ….. | - | - | - |
| 9+(E-1)\*2 | entity\_id\_(E) | UINT16 | N/A | The Identification number for the Eth entity in the payload. |
| 9+(E-1)\*2+1 | entity\_start\_address\_(E) | UINT16 | N/A | The start address of the Eth entity |
| 9+(E-1)\*2+2 | entity\_(1) | - | N/A | 1st entity in the packet |
| … | …. | - | - | - |
| - | entity\_ (E) | - | N/A | Eth entity in the packet |
| - | checksum | UINT8[2] | N/A | Two checksum bytes computed according to the Fletcher-8 checksum algorithm. |

The Pseudo-code for computing the two checksum bytes at the end of the packet, according to the Fletcher-8 checksum algorithm, is below:

Code Sample Fletcher-8 Checksum Algorithm Pseudo code

//N:length of the bytes on which the checksum is computed, which in this case are all the

//packet bytes excluding the first 3 header bytes and the last 2 checksum bytes

//J:The index of the first element on which the checksum is computed

//Buffer:The output packet buffer

CK**[**0**]** **=** 0

CK**[**1**]** **=** 0

**for(** I**=**J **;** I**<(**J**+**N**)** **;** I**++** **)**

**{**

CK**[**0**]** **=** CK**[**0**]** **+** Buffer**[**I**]**

CK**[**1**]** **=** CK**[**1**]** **+** CK**[**0**]**

**}**

### TppModeOfTransitEnum

|  |  |  |
| --- | --- | --- |
|  | **Member** | **Description** |
| 0 | TPP\_ MODE\_OF\_TRANSIT\_DRIVING | Specifies to the navigator to work in driving mode only. |
| 1 | TPP\_ MODE\_OF\_TRANSIT\_WALKING | Specifies to the navigator to work in walking mode only. |
| 2 | TPP\_MODE\_OF\_TRANSIT\_AUTOMATIC | Specifies to the navigator to work in automatic mode where the navigator will detect automatically if the mode of transit is walking or driving. |
| 3 | TPP\_MODE\_OF\_TRANSIT\_RUNNING | Specifies to the navigator to work in running mode only. |
| 4 | TPP\_MODE\_OF\_TRANSIT\_CYCLING | Specifies to the navigator to work in cycling mode only. |

### TppZuptModeEnum

|  |  |  |
| --- | --- | --- |
|  | **Member** | **Description** |
| 0 | TPP\_ZUPT\_MODE\_AUTOMATIC | Specifies to the navigator to work in the normal ZUPT mode where ZUPT is detected automatically. |
| 1 | TPP\_ZUPT\_MODE\_ON | Specifies to the navigator that the platform and device are in a static, no motion state. |
| 2 | TPP\_ZUPT\_MODE\_OFF | Specifies to the navigator that the platform or the device is moving. |

### TppMagnetometerCalibrationEnum

|  |  |  |
| --- | --- | --- |
|  | **Member** | **Description** |
| 0 | TPP\_MAGNETOMETER\_CALIBRATION\_OFF | Disables the automatic magnetometer calibration in the navigator. |
| 1 | TPP\_MAGNETOMETER\_CALIBRATION\_ON | Enables the automatic magnetometer calibration in the navigator. |

### TppUseCaseEnum

|  |  |  |
| --- | --- | --- |
|  | **Member** | **Description** |
| 0 | TPP\_USE\_CASE\_AUTOMATIC | Specifies to the navigator to automatically detect the use case. |
| 1 | TPP\_USE\_CASE\_TORSO\_AND\_HAND\_VIEWING | Example: Hand held, belt and other use cases that are not specified explicitly. |
| 2 | TPP\_USE\_CASE\_POCKET | Specifies to the navigator that the device is in pants’ pockets |
| 3 | TPP\_USE\_CASE\_HAND\_SWINGING | Specifies to the navigator that the device is swinging in hand |
| 4 | TPP\_USE\_CASE\_ARM | Specifies to the navigator that the phone is on the arm |
| ~~5~~ | ~~TPP\_USE\_CASE\_PURSE~~ | ~~Specifies to the navigator that the phone Is in a purse~~ |

### TppOrientationBasedOnPitchEnum (INTERNAL only)

|  |  |  |
| --- | --- | --- |
|  | **Member** | **Description** |
| -1 | TPP\_ORIENTATION\_VERTICAL\_DOWN |  |
| 0 | TPP\_ORIENTATION\_HORIZONTAL |  |
| +1 | TPP\_ORIENTATION\_VERTICAL\_UP |  |

### TppAnchorPointStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | type\_ | UINT8 | N/A | Anchor Point Type.  **0x01**: Start Anchor Point.  **0x02**: End Anchor Point.. |
| 2 | position\_available\_ | UNIT8 | N/A |  |
| 3 | latitude\_ | DOUBLE64 | deg |  |
| 4 | longitude\_ | DOUBLE64 | deg |  |
| 5 | height\_ | FLOAT32 | m |  |
| 6 | heading\_available\_ | UNIT8 | N/A |  |
| 7 | heading\_ | FLOAT32 | deg |  |

### TppZuptThresholdsStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | value\_x\_ | FLOAT32 | deg/sec |  |
| 2 | value\_y\_ | FLOAT32 | deg/sec |  |
| 3 | value\_z\_ | FLOAT32 | deg/sec |  |

### TppMultiDeviceLatencyStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | latency\_available\_ | UINT8 | N/A | 0x00: Latency is not available.  0x01: Latency is available |
| 2 | latency\_ | FLOAT32 | sec | Delay between the time the information is computed on the sender device and the time the information is processed on the receiver device.  If there is no delay, the user can use zero. |

### TppMultiDevicePositionVelocityStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | device\_identifier\_ | UINT64 | N/A | A unique identifier for the device that is sending this information.  When sending the device its own information, the value of the **device\_identifier\_** should be **zero**. |
| 2 | device\_type\_ | UINT8 | N/A | **0**: Phone  **1**: Tablet  **2**: Head-Mount  **3**: Watch |
| 3 | navigation\_phase\_ | INT8 | N/A | Equivalent to the **navigation\_phase\_** flag in TppSolutionStruct |
| 4 | gnss\_in\_use\_ | UINT8 | N/A | 0x00: GNSS data was not used in computing the navigation solution within the last 2 seconds.  0x01: GNSS data was used in computing the navigation solution within the last 2 seconds. |
| 5 | magnetometer\_in\_use\_ | UINT8 | N/A | 0x00: Magnetometer data was not used in computing the navigation solution within the last 2 seconds.  0x01: Magnetometer data was used in computing the navigation solution within the last 2 seconds. |
| 6 | barometer\_in\_use\_ | UINT8 | N/A | 0x00: Barometer data was not used in computing the navigation solution within the last 2 seconds.  0x01: Barometer data was used in computing the navigation solution within the last 2 seconds. |
| 7 | speed\_in\_use\_ | UINT8 | N/A | 0x00: Speed data was not used in computing the navigation solution within the last 2 seconds.  0x01: Speed data was used in computing the navigation solution within the last 2 seconds. |
| 8 | use\_case\_ | UINT8 | N/A |  |
| 9 | mode\_of\_transit\_ | UINT8 | N/A |  |
| 10 | latitude\_ | DOUBLE64 | deg |  |
| 11 | longitude\_ | DOUBLE64 | deg |  |
| 12 | height\_ | FLOAT32 | m |  |
| 13 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 14 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 15 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 16 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 17 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 18 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 19 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 20 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 21 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 22 | number\_of\_steps\_ | UINT32 | N/A |  |

### TppMultiDeviceHeadingStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | device\_identifier\_ | UINT64 | N/A | A unique identifier for the device that is sending this information.  When sending the device its own information, the value of the **device\_identifier\_** should be **zero**. |
| 2 | device\_type\_ | UINT8 | N/A | **0**: Phone  **1**: Tablet  **2**: Head-Mount  **3**: Watch |
| 3 | navigation\_phase\_ | INT8 | N/A | Equivalent to the values taken by the flag **navigation\_phase\_** in TppSolutionStruct |
| 4 | gnss\_in\_use\_ | UINT8 | N/A | 0x00: GNSS data was not used in computing the navigation solution within the last 2 seconds.  0x01: GNSS data was used in computing the navigation solution within the last 2 seconds. |
| 5 | magnetometer\_in\_use\_ | UINT8 | N/A | 0x00: Magnetometer data was not used in computing the navigation solution within the last 2 seconds.  0x01: Magnetometer data was used in computing the navigation solution within the last 2 seconds. |
| 6 | barometer\_in\_use\_ | UINT8 | N/A | 0x00: Barometer data was not used in computing the navigation solution within the last 2 seconds.  0x01: Barometer data was used in computing the navigation solution within the last 2 seconds. |
| 7 | speed\_in\_use\_ | UINT8 | N/A | 0x00: Speed data was not used in computing the navigation solution within the last 2 seconds.  0x01: Speed data was used in computing the navigation solution within the last 2 seconds. |
| 8 | use\_case\_ | UINT8 | N/A |  |
| 9 | mode\_of\_transit\_ | UINT8 | N/A |  |
| 10 | device\_heading\_ | FLOAT32 | deg |  |
| 11 | platform\_heading\_ | FLOAT32 | deg |  |
| 12 | heading\_misalignment\_ | FLOAT32 | deg |  |
| 13 | heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 14 | position\_north\_standard\_deviation | FLOAT32 | m |  |
| 15 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 16 | number\_of\_steps\_ | UINT32 | N/A |  |

### TppReturnStatusEnum

|  |  |  |
| --- | --- | --- |
|  | **Member** | **Description** |
| 0 | TPP\_SUCCESS | Returned if a call to any function in the library is successful. |
| 1 | TPP\_SYSTEM\_NOT\_INITIALIZED | Returned by tpp\_advance\_navigation\_step()if the function is called before the system is successfully initialized. |
| 2 | TPP\_INVALID\_IMU\_MESSAGE | Returned by tpp\_advance\_navigation\_step()when one or more members of the input parameter are invalid. |
| 3 | TPP\_INVALID\_GNSS\_PVT\_MESSAGE | Returned by tpp\_process\_gnss\_pvt()when one or more members of the input parameter are invalid. |
| 4 | TPP\_INVALID\_BAROMETER\_MESSAGE | Returned by tpp\_process\_barometer()when one or more members of the input parameter are invalid. |
| 5 | TPP\_INVALID\_MAGNETOMETER\_MESSAGE | Returned by tpp\_process\_magnetometer()when one or more members of the input parameter are invalid. |
| 6 | TPP\_INVALID\_SPEED\_MESSAGE | Returned by tpp\_process\_speed()when one or more members of the input parameter are invalid. |
| 7 | TPP\_INVALID\_MULTIPLE\_ANTENNA\_MESSAGE | Returned by tpp\_process\_multiple\_antenna()when one or more members of the input parameter are invalid. |
| ~~8~~ | ~~TPP\_INVALID\_2D\_POSITION\_MESSAGE~~ | ~~Returned by tpp\_process\_2d\_position()when one or more members of the input parameter are invalid.~~ |
| 9 | TPP\_INVALID\_HEIGHT\_MESSAGE | Returned by tpp\_process\_height()when one or more members of the input parameter are invalid. |
| 10 | TPP\_INVALID\_FLOOR\_INFORMATION\_MESSAGE | Returned by tpp\_process\_floor\_information()when one or more members of the input parameter are invalid. |
| 11 | TPP\_INVALID\_CALL\_INFORMATION\_MESSAGE | Returned by tpp\_process\_call\_information()when one or more members of the input parameter are invalid. |
| 12 | TPP\_INVALID\_OUTPUT | Returned by tpp\_advance\_navigation\_step()when the output byte array is invalid. |
| 13 | TPP\_STOP\_NAVIGATION\_FAILURE | Returned by tpp\_stop\_navigation()when the navigator failed to stop. |
| 14 | TPP\_INVALID\_PROCESS\_GNSS\_PVT\_CALL | Returned by tpp\_process\_gnss\_pvt()when the function is called after the developer specified that the data from GNSS will not be used in the navigation solution. |
| 15 | TPP\_INVALID\_PROCESS\_BAROMETER\_CALL | Returned by tpp\_process\_barometer()when the function is called after the developer specified that the data from the barometer will not be used in the navigation solution. |
| 16 | TPP\_INVALID\_PROCESS\_MAGNETOMETER\_CALL | Returned by tpp\_process\_magnetometer()when the function is called after the developer specified that the data from the magnetometer will not be used in the navigation solution. |
| 17 | TPP\_INVALID\_PROCESS\_SPEED\_CALL | Returned by tpp\_process\_speed()when the function is called after the developer specified that the data from the speed sensor will not be used in the navigation solution. |
| 18 | TPP\_INVALID\_PROCESS\_MULTIPLE\_ANTENNA\_CALL | Returned by tpp\_process\_multiple\_antenna()when the function is called after the developer specified that the data from the multiple antenna system will not be used in the navigation solution. |
| ~~19~~ | ~~TPP\_INVALID\_PROCESS\_2D\_POSITION\_CALL~~ | ~~Returned by tpp\_process\_2d\_position()when the function is called after the developer specified in the initialization that Wi-Fi will not be used and then provided Wi-FI data to the function.~~ |
| 20 | TPP\_INVALID\_SET\_MODE\_OF\_TRANSIT\_CALL | Returned by tpp\_set\_mode\_of\_transit() when the current mode of transit cannot be set to the one specified as input. |
| 21 | TPP\_INVALID\_FLAG\_MAGNETOMETER | Returned by tpp\_initialize\_SSS()when the values specified for the **MAGNETOMETER** initialization flag are not valid. |
| 22 | TPP\_INVALID\_FLAG\_BAROMETER | Returned by tpp\_initialize\_SSS()when the values specified for the **BAROMETER** initialization flag are not valid. |
| 23 | TPP\_INVALID\_FLAG\_SPEED | Returned by tpp\_initialize\_SSS()when the values specified for the **SPEED** initialization flag are not valid. |
| 24 | TPP\_INVALID\_FLAG\_GNSS | Returned by tpp\_initialize\_SSS()when the values specified for the **GNSS** initialization flag are not valid. |
| 25 | TPP\_INVALID\_HEADING\_TO\_START | Returned by tpp\_initialize\_SSS()when the values specified for the **GNSS**, **PLATFORM\_HEADING**, and **MAGNETOMETER** flags are not valid or off. |
| 26 | TPP\_INVALID\_POSITION\_TO\_START | Returned by tpp\_initialize\_SSS()when the values specified for the **GNSS** and **POSITION** flags are not valid. |
| 27 | TPP\_INVALID\_FLAG\_WIFI | Returned by tpp\_initialize\_SSS()when the values specified for the **WIFI** initialization flag are not valid. |
| 28 | TPP\_INVALID\_FLAG\_MULTIPLE\_ANTENNA | Returned by tpp\_initialize\_SSS()when the values specified for the **MULTIPLE\_ANTENNA** initialization flag are not valid. |
| 29 | TPP\_INVALID\_FLAG\_PLATFORM\_HEADING | Returned by tpp\_initialize\_SSS()when the values specified for the **PLATFORM\_HEADING** initialization flag are not valid. |
| 30 | TPP\_INVALID\_FLAG\_POSITION | Returned by tpp\_initialize\_SSS()when the values specified for the **POSITION** initialization flag are not valid. |
| 31 | TPP\_INVALID\_FLAG\_MISALIGNMENT | Returned by tpp\_initialize\_SSS()when the values specified for the **MISALIGNMENT** initialization flag are not valid. |
| 32 | TPP\_INVALID\_FLAG\_MODE\_OF\_TRANSIT | Returned by tpp\_initialize\_SSS()when the values specified for the **MODE\_OF\_TRANSIT** initialization flag are not valid. |
| 33 | TPP\_INVALID\_FLAG\_FLOOR\_INFORMATION | Returned by tpp\_initialize\_SSS()when the values specified for the **FLOOR\_INFORMATION** initialization flag are not valid. |
| 34 | TPP\_INVALID\_FLAG\_GYROSCOPE\_BIASES | Returned by tpp\_initialize\_SSS()when the values specified for the **GYROSCOPE\_BIASES** initialization flag are not valid. |
| 35 | TPP\_INVALID\_GYROSCOPE\_BIASES\_TO\_START | Returned by tpp\_initialize\_SSS()when the **GYROSCOPE\_BIASES** flag is not used in the initialization by the user. |
| 36 | TPP\_INVALID\_FLAG\_ACCELEROMETER\_CONFIGURATION | Returned by tpp\_initialize\_SSS()when the values specified for the **ACCELEROMETER\_CONFIGURATION** initialization flag are not valid. |
| 37 | TPP\_INVALID\_FLAG\_GYROSCOPE\_X\_CONFIGURATION | Returned by tpp\_initialize\_SSS()when the values specified for the **GYROSCOPE\_X\_CONFIGURATION** initialization flag are not valid. |
| 38 | TPP\_INVALID\_FLAG\_GYROSCOPE\_Y\_CONFIGURATION | Returned by tpp\_initialize\_SSS()when the values specified for the **GYROSCOPE\_Y\_CONFIGURATION** initialization flag are not valid. |
| 39 | TPP\_INVALID\_FLAG\_GYROSCOPE\_Z\_CONFIGURATION | Returned by tpp\_initialize\_SSS()when the values specified for the **GYROSCOPE\_Z\_CONFIGURATION** initialization flag are not valid. |
| 40 | TPP\_INVALID\_FLAG\_DEBUG\_DATA | Returned by tpp\_initialize\_SSS()when the values specified for the **DEBUG\_DATA** initialization flag are not valid. |
| 41 | TPP\_INVALID\_FLAG\_MOTION\_CONSTRAINTS | Returned by tpp\_initialize\_SSS()when the values specified for the **MOTION\_CONSTRAINTS** initialization flag are not valid. |
| 42 | TPP\_INVALID\_IMU\_MESSAGE\_TIMETAG | Returned by tpp\_advance\_navigation\_step() when the time-tag in the IMU message is incorrect with respect to the time-tag of the previous IMU message. In this case, the navigator stops and the system need to be reinitialized. |
| 43 | TPP\_INVALID\_IDENTIFIER\_TO\_START | Returned by tpp\_initialize\_SSS(), on specific platforms, when an invalid identifier is read from the platform. |
| 44 | TPP\_INVALID\_MODE\_OF\_TRANSIT\_TO\_START | Returned by tpp\_initialize\_SSS()when the values specified for the **GNSS**, and **MODE\_OF\_TRANSIT** flags are not valid. |
| 45 | TPP\_INVALID\_FLAG\_BACKWARD\_SMOOTHING | Returned by tpp\_initialize\_SSS()when the values specified for the **BACKWARD\_SMOOTHING** initialization flag are not valid. |
| 46 | TPP\_BACKWARD\_SMOOTHING\_FILES\_CREATION\_FAILURE | Returned by tpp\_initialize\_SSS()when the library fails to create the backward smoothed solution file and other temporary files in the folder path specified by **flag\_value\_2\_** in the **BACKWARD\_SMOOTHING** flag. |
| 47 | TPP\_INVALID\_RUN\_BACKWARD\_SMOOTHING\_CALL | Returned by tpp\_run\_backward\_smoothing()when the navigator was initialized with **BACKWARD\_SMOOTHING** flag being “**off**” or when the navigator is not stopped before calling this function. |
| 48 | TPP\_RUN\_BACKWARD\_SMOOTHING\_FAILURE | Returned by tpp\_run\_backward\_smoothing() when  (1) No information was saved by the navigator for backward smoothing to run successfully.  (2) If the navigator is stopped before it outputs a reliable navigation solution.  (3) If the library failed to save the backward smoothed navigation solution to the file “**bs\_solution.dat**” |
| 55 | TPP\_INVALID\_SYSTEM\_DATE\_TO\_START | Returned by tpp\_initialize\_SSS() when the system date is an invalid date with respect to the date of building the library. |
| 56 | TPP\_INVALID\_FLAG\_START\_OPTION | Returned by tpp\_initialize\_SSS()when the value specified for the **START\_OPTION** initialization flag is not valid. |
| 57 | TPP\_INVALID\_OPTION\_SENSORS\_ONLY | Returned by tpp\_initialize\_SSS() when **flag\_value\_1\_** for the **START\_OPTION** flag is “sensors\_only” and **flag\_value\_1\_** for **GNSS** or **WIFI** flags is “on” (i.e. data from the GNSS receiver and Wi-Fi is specified to be used by the navigator) |
| 58 | TPP\_INVALID\_OPTION\_SENSORS\_RELIABLE\_WIRELESS | Returned by tpp\_initialize\_SSS() when **flag\_value\_1\_** for the **START\_OPTION** flag is “sensors\_reliable\_wireless” and **flag\_value\_1\_** for both **GNSS** and **WIFI** flags is “off” (i.e. data from the GNSS receiver and Wi-Fi is specified not to be used by the navigator) |
| 59 | TPP\_INVALID\_OPTION\_SENSORS\_WIRELESS | Returned by tpp\_initialize\_SSS() when **flag\_value\_1\_** for the **START\_OPTION** flag is “sensors\_wireless” and **flag\_value\_1\_** for both **GNSS** and **WIFI** flags is “off” (i.e. data from the GNSS receiver and Wi-Fi is specified not to be used by the navigator) |
| 60 | TPP\_INVALID\_FLAG\_DEVICE | Returned by tpp\_initialize\_SSS()when the value specified for the **DEVICE** initialization flag is not valid. |
| 61 | TPP\_INVALID\_FLAG\_ACCELEROMETER\_BIASES | Returned by tpp\_initialize\_SSS()when the values specified for the **ACCELEROMETER\_BIASES** initialization flag are not valid. |
| 62 | TPP\_INVALID\_PLATFORM\_HEADING\_MESSAGE | Returned by tpp\_process\_platform\_heading()when one or more members of the input parameter are invalid. |
| 63 | TPP\_INVALID\_QUATERNIONS\_MESSAGE | Returned by tpp\_process\_9dof\_quaternions()or tpp\_process\_6dof\_quaternions() when one or more members of the input parameter are invalid. |
| 64 | TPP\_INVALID\_SET\_ZUPT\_MODE\_CALL | Returned by tpp\_set\_zupt\_mode() when the ZUPT mode cannot be set to the one specified as input. |
| 65 | TPP\_INVALID\_SET\_MISALIGNMENT\_MESSAGE | Returned by tpp\_set\_misalignment()when one or more members of the input parameter are invalid. |
| 66 | TPP\_INVALID\_SET\_MISALIGNMENT\_CALL | Returned by tpp\_set\_misalignment() when the misalignment estimation cannot be set to the one specified as input. |
| 67 | TPP\_INVALID\_SET\_MAGNETOMETER\_CALIBRATION\_CALL | Returned by tpp\_set\_magnetometer\_calibration() when the magnetometer calibration setting cannot be set to the one specified as input. |
| 68 | TPP\_INVALID\_SET\_USE\_CASE\_CALL | Returned by tpp\_set\_use\_case() when the use case cannot be set to the one specified as input. |
| 69 | TPP\_INVALID\_FLAG\_OUTPUT\_POSITION\_ERROR\_WEIGHTING | Returned by tpp\_initialize\_SSS()when the values specified for the **OUTPUT\_POSITION\_ERROR\_WEIGHTING** initialization flag are not valid. |
| 70 | TPP\_INVALID\_FLAG\_OUTPUT\_VELOCITY\_ERROR\_WEIGHTING | Returned by tpp\_initialize\_SSS()when the values specified for the **OUTPUT\_VELOCITY\_ERROR\_WEIGHTING** initialization flag are not valid. |
| 71 | TPP\_INVALID\_FLAG\_OUTPUT\_ATTITUDE\_ERROR\_WEIGHTING | Returned by tpp\_initialize\_SSS()when the values specified for the **OUTPUT\_ATTITUDE\_ERROR\_WEIGHTING** initialization flag are not valid. |
| 72 | TPP\_INVALID\_IMU\_BIASES\_MESSAGE | Returned by tpp\_process\_imu\_biases()when one or more members of the input parameter are invalid. |
| 73 | TPP\_INVALID\_OPERATOR\_2D\_POSITION\_MESSAGE | Returned by tpp\_process\_operator\_2d\_position()when one or more members of the input parameter are invalid. |
| 74 | TPP\_INVALID\_WIRELESS\_MESSAGE | Returned by tpp\_process\_wireless()when one or more members of the input parameters are invalid. |
| 75 | TPP\_INVALID\_PROCESS\_WIFI\_CALL | Returned by tpp\_process\_wireless()when the function is called after the developer specified in the initialization that Wi-Fi will not be used and then provided Wi-FI data to the function. |
| 76 | TPP\_INVALID\_PROCESS\_MISALIGNMENT\_MESSAGE | Returned by tpp\_process\_misalignment()when one or more members of the input parameter are invalid. |
| 77 | TPP\_INVALID\_DEVICE\_HEADING\_MESSAGE | Returned by tpp\_set\_device\_heading()when one or more members of the input parameter are invalid. |
| 78 | TPP\_INVALID\_FLAG\_ZUPT\_THRESHOLDS | Returned by tpp\_initialize\_SSS()when the values specified for the **ZUPT\_THRESHOLDS** initialization flag are not valid. |
| 79 | TPP\_INVALID\_ZUPT\_THRESHOLDS\_TO\_START | Returned by tpp\_initialize\_SSS()when **flag\_value\_1\_** for the **GNSS** flag is “off” and the **ZUPT\_THRESHOLDS** flag is not used in the initialization by the user. |
| 80 | TPP\_INVALID\_NAVIGATION\_SESSION\_HANDLE | Returned by any function that uses the navigation session handle when the handle passed is invalid. |
| 81 | TPP\_INVALID\_FLAG\_DECLINATION\_ANGLE | Returned by tpp\_initialize\_SSS()when **flag\_value\_1\_** for the **DECLINATION\_ANGLE** flag is not valid. |
| 100 | TPP\_COMPUTATION\_IN\_PROGRESS | Returned by tpp\_prerun\_compute\_zupt\_thresholds\_imu() when the computation operation is still in progress and no values are ready for the user to use. |
| 101 | TPP\_ZUPT\_THRESHOLDS\_NOT\_RESET | Returned when tpp\_prerun\_reset\_zupt\_thresholds() is not called before calling the computation functions that are used for computing the ZUPT Threshold values. |
| 102 | TPP\_INVALID\_VENUE\_MAP\_MESSAGE | Returned by tpp\_process\_venue\_map() when one or more members of the input parameter are invalid. |
| 103 | TPP\_INVALID\_FLAG\_REPLACE\_ATTITUDE\_WITH\_6DOF | Returned by tpp\_initialize\_SSS()when the values specified for the **REPLACE\_ATTITUDE\_WITH\_6DOF** initialization flag are not valid. |
| 104 | TPP\_INVALID\_ANCHOR\_POINT\_PACKET | Returned by ***tpp\_add\_anchor\_point()*** when one or more members of ***anchor\_point\_message\_pointer*** are invalid. |
| 105 | TPP\_INVALID\_START\_ANCHOR\_POINT | Returned by tpp\_add\_anchor\_point() when one or more members of the **start** anchor point are invalid. |
| 106 | TPP\_INVALID\_END\_ANCHOR\_POINT | Returned by tpp\_add\_anchor\_point() when one or more members of the **end** anchor point are invalid, or if the developer wanted to add an end anchor point without adding a valid start anchor point first. |
| 107 | TPP\_INVALID\_FLAG\_PROCESSING\_MODE | Returned by tpp\_initialize\_SSS() when the values specified for **PROCESSING\_MODE** initialization flag are not valid |
| 108 | TPP\_INVALID\_FLAG\_USE\_CASE | Returned by tpp\_initialize\_SSS() when the values specified for **USE\_CASE** initialization flag are not valid |
| 109 | TPP\_INVALID\_FLAG\_LIBRARY\_CALL\_TYPE | Returned by tpp\_initialize\_SSS() when the values specified for **LIBRARY\_CALL\_TYPE** initialization flag are not valid |
| 150 | TPP\_MULTI\_DEVICE\_INVALID\_NUMBER\_OF\_DEVICES | Returned by the multi-device API functions when the number of devices is less than 2 or greater than 8. |
| 151 | TPP\_MULTI\_DEVICE\_INVALID\_LATENCY\_MESSAGE | Returned by the multi-device API functions when the one or more fields of the TppMultiDeviceLatencyStruct argument passed to the functions is invalid. |
| 152 | TPP\_MULTI\_DEVICE\_IDENTIFIER\_ZERO\_NOT\_PRESENT | Returned by the multi-device API functions when the device identifier of zero is not present in the list of device identifiers in TppMultiDevicePositionVelocityStruct |
| 153 | TPP\_MULTI\_DEVICE\_INVALID\_POSITION\_VELOCITY\_MESSAGE | Returned by tpp\_multi\_device\_process\_position\_velocity() when one or more fields of TppMultiDevicePositionVelocityStruct are invalid. |
| 154 | TPP\_MULTI\_DEVICE\_INVALID\_HEADING\_ MESSAGE | Returned by tpp\_multi\_device\_process\_heading() when one or more fields of TppMultiDeviceHeadingStruct are invalid. |
| 155 | TPP\_MULTI\_DEVICE\_INVALID\_SECONDARY\_GNSS\_DEVICE\_MESSAGE | Returned by tpp\_multi\_device\_process\_secondary\_gnss\_pvt()when one or more fields of TppGnssPvtMessageStruct are invalid. |
| 156 | TPP\_MULTI\_DEVICE\_INVALID\_PROCESS\_SECONDARY\_GNSS\_PVT\_CALL | Returned by tpp\_multi\_device\_process\_secondary\_gnss\_pvt()when the function is called after the developer specified that the data from GNSS will not be used in the navigation solution. |
| 255 | TPP\_INVALID\_SYNCHRONIZATION\_EVENT | Returned by tpp\_add\_synchronization\_event()when the value of the parameter passed is not greater than zero or when the value of the parameter passed is less than the value of the previous event number. |
|  | TPP\_INVALID\_GNSS\_OBSERVATIONS\_MESSAGE | Returned by tpp\_process\_gnss\_observations()when one or more members of the input parameter are invalid. |
|  | TPP\_INVALID\_GLONASS\_EPHEMERIS\_MESSAGE | Returned by tpp\_process\_glonass\_ephemeris()when one or more members of the input parameter are invalid. |
|  | TPP\_INVALID\_GPS\_EPHEMERIS\_MESSAGE | Returned by tpp\_process\_gps\_ephemeris()when one or more members of the input parameter are invalid. |

### TppGnssObservationsMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_week\_ | UINT32 | N/A | Week |
| 2 | gnss\_tow\_ | DOUBLE64 | sec | Time of week |
| 3 | satellite\_num\_ | UINT32 | N/A | Number of satellites |
| 4 | gnss\_prn\_ | UINT32 | N/A | PRN |
| 5 | satellite\_system\_ | UINT32 | N/A | Satellite System:  0x00: GPS  0x01: GLONASS  0x02:COMPASS  0x03: GALILEO |
| 6 | signal\_type\_ | UINT32 | N/A | 0x00: L1/CA  0x01: L2  0x02: L5 |
| 7 | phase\_lock\_ | INT32 | N/A | 0x00: Lose Lock  0x01: Lock |
| 8 | code\_lock\_ | INT32 | N/A | 0x00: Lose Lock  0x01: Lock |
| 9 | lose\_lock\_indicator | UINT32 | N/A | Lose Lock Indicator |
| 10 | cn0\_ | DOUBLE64 | dB-Hz | Carrier to Noise density |
| 11 | lock\_time\_ | DOUBLE64 | sec | Time since the latest lock |
| 12 | pseudorange\_ | DOUBLE64 | m |  |
| 13 | pseudorange \_std\_ | DOUBLE64 | m |  |
| 14 | doppler\_ | DOUBLE64 | Hz |  |
| 15 | doppler\_std\_ | DOUBLE64 | Hz |  |
| 16 | phase\_ | DOUBLE64 | Cycles |  |

### TppGpsEphemerisMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_prn\_ | UINT32 | N/A | PRN |
| 2 | gnss\_week\_ | UINT32 | N/A | Week number |
| 3 | gnss\_tow\_ | DOUBLE64 | sec | Time of week |
| 4 | code\_ | UINT32 | N/A | Code on L2  0x00: Reserved  0x01: P code on  0x02: C/A code on |
| 5 | ura\_ | DOUBLE64 | m | Satellite accuracy |
| 6 | health\_ | UINT32 | N/A | Satellite health  0x00: all navigation date are ok  0x01: some navigation date are bad |
| 7 | iodc\_ | UINT32 | N/A |  |
| 8 | flag\_ | UINT32 | N/A | Data flag on L2P |
| 9 | tgd\_ | DOUBLE64 | sec |  |
| 10 | toc\_ | DOUBLE64 | sec |  |
| 11 | af2\_ | DOUBLE64 | sec/sec/sec |  |
| 12 | af1\_ | DOUBLE64 | sec/sec |  |
| 13 | af0\_ | DOUBLE64 | sec |  |
| 14 | iode2\_ | UINT32 | N/A |  |
| 15 | crs\_ | DOUBLE64 | m |  |
| 16 | deltan\_ | DOUBLE64 | semi-circles/sec |  |
| 17 | m0\_ | DOUBLE64 | semi-circles |  |
| 18 | cuc\_ | DOUBLE64 | rad |  |
| 19 | ecc\_ | DOUBLE64 | N/A |  |
| 20 | cus\_ | DOUBLE64 | rad |  |
| 21 | sqrta\_ | DOUBLE64 | √m |  |
| 22 | toe\_ | DOUBLE64 | sec |  |
| 23 | cic\_ | DOUBLE64 | rad |  |
| 24 | omega0\_ | DOUBLE64 | rad/sec |  |
| 25 | cis | DOUBLE64 | rad |  |
| 26 | i0\_ | DOUBLE64 | semi-circles |  |
| 27 | crc\_ | DOUBLE64 | m |  |
| 28 | omega\_ | DOUBLE64 | semi-circles |  |
| 29 | omegadot\_ | DOUBLE64 | semi-circles/sec |  |
| 30 | iode3\_ | UINT32 | N/A |  |
| 31 | idot\_ | DOUBLE64 | semi-circles/sec |  |

### TppGlonassEphemerisMessageStruct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_prn\_ | UINT32 | N/A |  |
| 2 | frequency\_ | INT32 | N/A | Frequency offset |
| 3 | toe\_ | DOUBLE64 | sec |  |
| 4 | toff\_ | DOUBLE64 | sec |  |
| 5 | iode\_ | UINT32 | N/A |  |
| 6 | health\_ | UINT32 | N/A | Satellite health  0x00: all navigation date are ok  0x01: some navigation date are bad |
| 7 | pos\_x\_ | DOUBLE64 | m |  |
| 8 | pos\_y\_ | DOUBLE64 | m |  |
| 9 | pos\_z\_ | DOUBLE64 | m |  |
| 10 | vel\_x\_ | DOUBLE64 | m/sec |  |
| 11 | vel\_y\_ | DOUBLE64 | m/sec |  |
| 12 | vel\_z\_ | DOUBLE64 | m/sec |  |
| 13 | acc\_x\_ | DOUBLE64 | m/sec/sec |  |
| 14 | acc\_y\_ | DOUBLE64 | m/sec/sec |  |
| 15 | acc\_z\_ | DOUBLE64 | m/sec/sec |  |
| 16 | taun\_ | DOUBLE64 | sec |  |
| 17 | gamman\_ | DOUBLE64 | sec/sec |  |
| 18 | tof\_ | DOUBLE64 | sec |  |
| 19 | age\_ | INT32 | N/A |  |

## Using Trusted Positioning Library

Trusted Positioning API provides functions and data types to do the following:

1. Initialize the navigator.
2. Process the sensors’ data that is pushed in by the developer in the specified format as defined herein.
3. Output a structured byte array that the developer can parse and extract the required data or save in a file.

### ­Initialization

#### TMN Initialization

Following is a code sample to show how to do the initialization for TMN. First, the developer creates a navigation session handle, and then specifies an array of flags where each entry in the array specifies the flag name and the corresponding value(s).

In this sample code, the developer specifies the first flag name to be “**MAGNETOMETER**” with the first and second values of this flag being “**on**” and “**1.0**” which corresponds to the navigator utilizing the magnetometer data with original weighting as provided in the standard deviation field.

The developer then specifies the second flag name to be “**BAROMETER**” with the first and second values of this flag being “**on**” and “**1.0**” which corresponds to the navigator utilizing the barometer data with original weighting as provided in the standard deviation field.

The developer then specifies the number of initialization flags and the flags array in a variable of type tpp\_initialization\_structure and passes a pointer of the variable to tpp\_initialize\_tmn\_SSS(). As there are no other flags, TMN will use the default settings for those flags.

Code Sample TMN Initialization (1)

//Creating a navigation session handle

TppNavigationSessionHandle tpp\_handle = tpp\_create\_navigation\_session\_handle();

if(tpp\_handle **==** **NULL**)

{

**return;**

}

//Holds the number of initialization flags along with a pointer to the initialization structure //array

TppInitializationStruct tpp\_initialization\_struct**;**

//Used as a short name instead of "tpp\_initialization\_struct.tpp\_initialization\_flag\_pointer\_"

TppInitializationFlagStruct**\*** flags\_struct\_pointer**;**

//Holds the actual number of flags that the developer initialized

unsigned short flags\_counter**;**

//Holds the number of maximum number of initialization flags the developer will allocate space //for

const unsigned short MAXIMUM\_NUMBER\_OF\_FLAGS **=** 20**;**

//Holds the returned array from tpp\_initialize\_fff()and its size to be written to a file for

//Trusted Positioning debugging

TppOutputStruct tpp\_initialization\_output\_array\_struct**;**

TppReturnStatusEnum tpp\_return\_status\_enum**;**

tpp\_initialization\_output\_array\_struct**.**byte\_array\_pointer\_ **=**

**(**TppTypeUint8**\*)**malloc**(**TPP\_INIT\_BYTE\_ARRAY\_LENGTH**);**

flags\_counter **=** 0**;**

flags\_struct\_pointer **=**

**(**TppInitializationFlagStruct**\*)** malloc**(**MAXIMUM\_NUMBER\_OF\_FLAGS **\*** **sizeof(**TppInitializationFlagStruct**));**

memset**(**flags\_struct\_pointer**,** 0**,**MAXIMUM\_NUMBER\_OF\_FLAGS **\*** **sizeof(**TppInitializationFlagStruct**));**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].** flag\_name\_**,**"BAROMETER"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_**,**"on"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc**(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.8f" **,** 1.0**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"MAGNETOMETER"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_**,**"on"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc**(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.8f" **,** 1.0**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"GYROSCOPE\_BIASES"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**10**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc**(**10**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **=** **(**char**\*)** malloc**(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **,** "%.5f" **,** 0.0**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.5f" **,** 0.0**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **,** "%.5f" **,** 0.0**);**

flags\_counter**++;**

tpp\_initialization\_structure**.**number\_of\_initialization\_flags\_ **=** flags\_counter**;**

tpp\_initialization\_struct**.**tpp\_initialization\_flag\_pointer\_ **=** flags\_struct\_pointer**;**

tpp\_return\_status**=**tpp\_initialize\_tmn\_SSS

**(**tpp\_handle, **&**tpp\_initialization\_struct **,&**tpp\_initialization\_output\_array\_struct**);**

**if(** tpp\_return\_status **!=** TPP\_SUCCESS **)**

**{**

printf**(**"[ERROR] tpp\_initialize\_tmn\_SSS() failed\n"**);**

**}**

After initialization, the user can free the allocated memory used in populating the flags and their values according the code sample below.

Code Sample 3 TMN Initialization (2)

int counter**;**

**for(**counter **=** 0**;** counter **<** flags\_counter **;** counter**++){**

**if(**flags\_struct\_pointer**[**counter**].**flag\_name\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_name\_**);**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_1\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_1\_**);**

**}**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_2\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_2\_**);**

**}**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_3\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_3\_**);**

**}**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_4\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_4\_**);**

**}**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_5\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_5\_**);**

**}**

**}**

**}**

free**(**flags\_struct\_pointer);

The following code sample shows an example where the initial position and heading is sent to TMN.

Code Sample TMN Initialization (2)

//Holds the number of initialization flags along with a pointer to the initialization structure array

TppInitializationStruct tpp\_initialization\_struct**;**

//Used as a short name instead of "tpp\_initialization\_struct.tpp\_initialization\_flag\_pointer\_"

TppInitializationFlagStruct**\*** flags\_struct\_pointer**;**

//Holds the actual number of flags that the developer initialized

unsigned short flags\_counter**;**

//Holds the number of maximum number of initialization flags the developer will allocate space for

const unsigned short MAXIMUM\_NUMBER\_OF\_FLAGS **=** 20**;**

//Holds the returned array from tpp\_initialize\_fff()and its size to be written to a file for

//Trusted Positioning debugging

TppOutputStruct tpp\_initialization\_output\_array\_struct**;**

TppReturnStatusEnum tpp\_return\_status\_enum**;**

tpp\_initialization\_output\_array\_struct**.**byte\_array\_pointer\_ **=**

**(**TppTypeUint8**\*)**malloc**(**TPP\_INIT\_BYTE\_ARRAY\_LENGTH**);**

flags\_counter **=** 0**;**

flags\_struct\_pointer **=**

**(**TppInitializationFlagStruct**\*)** malloc**(**MAXIMUM\_NUMBER\_OF\_FLAGS **\*** **sizeof(**TppInitializationFlagStruct**));**

memset**(**flags\_struct\_pointer**,** 0**,**MAXIMUM\_NUMBER\_OF\_FLAGS **\*** **sizeof(**TppInitializationFlagStruct**));**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"BAROMETER"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_**,**"on"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_**,**"1.0"**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"GYROSCOPE\_BIASES"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc **(**10**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc **(**10**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **=** **(**char**\*)** malloc **(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **,** "%.5f" **,** 0.0**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.5f" **,** 0.0**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **,** "%.5f" **,** 0.0**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"POSITION"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc **(**20**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc **(**20**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **=** **(**char**\*)** malloc **(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **,** "%.8f" **,** user\_initial\_latitude**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.8f" **,** user\_initial\_longitude**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **,** "%.2f" **,** user\_initial\_height**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"PLATFORM\_HEADING"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc **(**20**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **,** "%.8f" **,** user\_initial\_heading**);**

flags\_counter**++;**

tpp\_initialization\_structure**.**number\_of\_initialization\_flags\_ **=** flags\_counter**;**

tpp\_initialization\_struct**.**tpp\_initialization\_flag\_pointer\_ **=** flags\_struct\_pointer**;**

tpp\_return\_status **=** tpp\_initialize\_tmn\_SSS**(** **&**tpp\_initialization\_struct **,** **&**tpp\_initialization\_output\_array\_struct**);**

**if(** tpp\_return\_status **!=** TPP\_SUCCESS **)**

**{**

printf**(**"[ERROR] tpp\_initialize\_tmn\_SSS() failed\n"**);**

**}**

#### TPN Free Initialization

Following is a code sample to show how to do the initialization for TPN free. First, the developer creates a navigation session handle, and then specifies an array of flags where each entry in the array specifies the flag name and the corresponding value(s).

In this sample code, the developer specifies the first flag name to be “**MAGNETOMETER**” with the first and second values of this flag being “**on**” and “**1.0**” which corresponds to the navigator utilizing the magnetometer data with original weighting as provided in the standard deviation field.

The developer then specifies the second flag name to be “**BAROMETER**” with the first and second values of this flag being “**on**” and “**1.0**” which corresponds to the navigator utilizing the barometer data with original weighting as provided in the standard deviation field.

The third flag specified is “**GYROSCOPE\_BIASES**”. The biases’ values are zeros for the x-axis, y-axis and the z-axis gyroscopes.

The developer then specifies the number of initialization flags and the flags array in a variable of type tpp\_initialization\_structure and passes a pointer of the variable to tpp\_initialize\_tpn\_free(). As there are no other flags, TPN will use the default settings for the flags that were not initialized.

Code Sample TPN Free Initialization (1)

//Creating a navigation session handle

TppNavigationSessionHandle tpp\_handle = tpp\_create\_navigation\_session\_handle();

if(tpp\_handle **==** **NULL**)

{

**return;**

}

//Holds the number of initialization flags along with a pointer to the initialization structure //array

TppInitializationStruct tpp\_initialization\_struct**;**

//Used as a short name instead of "tpp\_initialization\_struct.tpp\_initialization\_flag\_pointer\_"

TppInitializationFlagStruct**\*** flags\_struct\_pointer**;**

//Holds the actual number of flags that the developer initialized

unsigned short flags\_counter**;**

//Holds the number of maximum number of initialization flags the developer will allocate space //for

const unsigned short MAXIMUM\_NUMBER\_OF\_FLAGS **=** 20**;**

//Holds the returned array from tpp\_initialize\_fff()and its size to be written to a file for

//Trusted Positioning debugging

TppOutputStruct tpp\_initialization\_output\_array\_struct**;**

TppReturnStatusEnum tpp\_return\_status\_enum**;**

tpp\_initialization\_output\_array\_struct**.**byte\_array\_pointer\_ **=**

**(**TppTypeUint8**\*)**malloc**(**TPP\_INIT\_BYTE\_ARRAY\_LENGTH**);**

flags\_counter **=** 0**;**

flags\_struct\_pointer **=**

**(**TppInitializationFlagStruct**\*)** malloc**(**MAXIMUM\_NUMBER\_OF\_FLAGS **\*** **sizeof(**TppInitializationFlagStruct**));**

memset**(**flags\_struct\_pointer**,** 0**,**MAXIMUM\_NUMBER\_OF\_FLAGS **\*** **sizeof(**TppInitializationFlagStruct**));**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"BAROMETER"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_**,**"on"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc**(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.8f" **,** 1.0**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"MAGNETOMETER"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_**,**"on"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc**(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.8f" **,** 1.0**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"GYROSCOPE\_BIASES"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**10**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc**(**10**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **=** **(**char**\*)** malloc**(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **,** "%.5f" **,** 0.0**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.5f" **,** 0.0**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **,** "%.5f" **,** 0.0**);**

flags\_counter**++;**

tpp\_initialization\_structure**.**number\_of\_initialization\_flags\_ **=** flags\_counter**;**

tpp\_initialization\_struct**.**tpp\_initialization\_flag\_pointer\_ **=** flags\_struct\_pointer**;**

tpp\_return\_status**=**tpp\_initialize\_tpn\_free

**(**tpp\_handle, **&**tpp\_initialization\_struct **,&**tpp\_initialization\_output\_array\_struct**);**

**if(** tpp\_return\_status **!=** TPP\_SUCCESS **)**

**{**

printf**(**"[ERROR] tpp\_initialize\_tpn\_free() failed\n"**);**

**}**

After initialization, the user can free the allocated memory used in populating the flags and their values according the code sample below.

Code Sample TPN Free Initialization (2)

int counter**;**

**for(**counter **=** 0**;** counter **<** flags\_counter **;** counter**++){**

**if(**flags\_struct\_pointer**[**counter**].**flag\_name\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_name\_**);**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_1\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_1\_**);**

**}**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_2\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_2\_**);**

**}**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_3\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_3\_**);**

**}**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_4\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_4\_**);**

**}**

**if(**flags\_struct\_pointer**[**counter**].**flag\_value\_5\_ **!=** **NULL){**

free**(**flags\_struct\_pointer**[**counter**].**flag\_value\_5\_**);**

**}**

**}**

**}**

free**(**flags\_struct\_pointer);

The following code sample shows an example where TPN free is initialized to work without GNSS and without magnetometer, and the developer provided the initial position and heading of the platform.

Code Sample TPN Free Initialization (3)

//Holds the number of initialization flags along with a pointer to the initialization structure array

TppInitializationStruct tpp\_initialization\_struct**;**

//Used as a short name instead of "tpp\_initialization\_struct.tpp\_initialization\_flag\_pointer\_"

TppInitializationFlagStruct**\*** flags\_struct\_pointer**;**

//Holds the actual number of flags that the developer initialized

unsigned short flags\_counter**;**

//Holds the number of maximum number of initialization flags the developer will allocate space for

const unsigned short MAXIMUM\_NUMBER\_OF\_FLAGS **=** 20**;**

//Holds the returned array from tpp\_initialize\_fff()and its size to be written to a file for

//Trusted Positioning debugging

TppOutputStruct tpp\_initialization\_output\_array\_struct**;**

TppReturnStatusEnum tpp\_return\_status\_enum**;**

tpp\_initialization\_output\_array\_struct**.**byte\_array\_pointer\_ **=**

**(**TppTypeUint8**\*)**malloc**(**TPP\_INIT\_BYTE\_ARRAY\_LENGTH**);**

flags\_counter **=** 0**;**

flags\_struct\_pointer **=**

**(**TppInitializationFlagStruct**\*)** malloc**(**MAXIMUM\_NUMBER\_OF\_FLAGS **\*** **sizeof(**TppInitializationFlagStruct**));**

memset**(**flags\_struct\_pointer**,** 0**,**MAXIMUM\_NUMBER\_OF\_FLAGS **\*** **sizeof(**TppInitializationFlagStruct**));**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"BAROMETER"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_**,**"on"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_**,**"1.0"**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"GYROSCOPE\_BIASES"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc **(**10**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc **(**10**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **=** **(**char**\*)** malloc **(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **,** "%.5f" **,** 0.0**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.5f" **,** 0.0**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **,** "%.5f" **,** 0.0**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"GNSS"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_**,**"off"**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"POSITION"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc **(**20**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **=** **(**char**\*)** malloc **(**20**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **=** **(**char**\*)** malloc **(**10**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **,** "%.8f" **,** user\_initial\_latitude**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_2\_ **,** "%.8f" **,** user\_initial\_longitude**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_3\_ **,** "%.2f" **,** user\_initial\_height**);**

flags\_counter**++;**

flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_ **=** **(**char**\*)** malloc**(**40**);**

strcpy**(**flags\_struct\_pointer**[**flags\_counter**].**flag\_name\_**,**"PLATFORM\_HEADING"**);**

flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **=** **(**char**\*)** malloc **(**20**);**

sprintf**(** flags\_struct\_pointer**[**flags\_counter**].**flag\_value\_1\_ **,** "%.8f" **,** user\_initial\_heading**);**

flags\_counter**++;**

tpp\_initialization\_structure**.**number\_of\_initialization\_flags\_ **=** flags\_counter**;**

tpp\_initialization\_struct**.**tpp\_initialization\_flag\_pointer\_ **=** flags\_struct\_pointer**;**

tpp\_return\_status **=** tpp\_initialize\_tpn\_free**(** **&**tpp\_initialization\_struct **,** **&**tpp\_initialization\_output\_array\_struct**);**

**if(** tpp\_return\_status **!=** TPP\_SUCCESS **)**

**{**

printf**(**"[ERROR] tpp\_initialize\_tpn\_free() failed\n"**);**

**}**

#### TPN Tethered Initialization

Following is an example of code to show how to do the initialization for TPN tethered. First, the developer creates a navigation session handle, and then specifies an array of flags where each entry in the array specifies the flag name and the corresponding value(s).

In this sample code, the developer specifies the first flag name to be “**MAGNETOMETER**” with the first and second values of this flag being “**on**” and “**1.0**” which corresponds to the navigator utilizing the magnetometer data with original weighting as provided in the standard deviation field.

The developer then specifies the second flag name to be “**BAROMETER**” with the first and second values of this flag being “**on**” and “**1.0**” which corresponds to the navigator utilizing the barometer data with original weighting as provided in the standard deviation field.

The developer then specifies the number of initialization flags and the flags array in a variable of type tpp\_initialization\_structure and passes a pointer of the variable to tpp\_initialize\_tpn\_tethered(). As there are no other flags, TPN will use the default settings for those flags.

Code Sample TPN Tethered Initialization (1)

//Creating a navigation session handle

TppNavigationSessionHandle tpp\_handle = tpp\_create\_navigation\_session\_handle();

if(tpp\_handle **==** **NULL**)

{

**return;**

}



The following code sample shows an example where TPN Tethered is initialized to work without GNSS and without magnetometer, and the developer provided the initial position and heading.

Code Sample TPN Tethered Initialization (2)



#### TVN Initialization

Following is a code sample to show how to do the initialization for TVN. First, the developer creates a navigation session handle, and then specifies an array of flags where each entry in the array specifies the flag name and the corresponding value(s).

In this sample code, the developer specifies the first flag name to be “**MAGNETOMETER**” with the first and second values of this flag being “**on**” and “**1.0**” which corresponds to the navigator utilizing the magnetometer data with original weighting as provided in the standard deviation field.

The developer then specifies the second flag name to be “**BAROMETER**” with the first and second values of this flag being “**on**” and “**1.0**” which corresponds to the navigator utilizing the barometer data with original weighting as provided in the standard deviation field.

The developer then specifies the number of initialization flags and the flags array in a variable of type tpp\_initialization\_structure and passes a pointer of the variable to tpp\_initialize\_tvn(). As there are no other flags, TVN will use the default settings for those flags.

Code Sample TVN Initialization (1)

//Creating a navigation session handle

TppNavigationSessionHandle tpp\_handle = tpp\_create\_navigation\_session\_handle();

if(tpp\_handle **==** **NULL**)

{

**return;**

}



The following code sample shows an example where the initial position and heading is sent to TVN.

Code Sample TVN Initialization (2)



### Data Processing

For each sensor, there is a specific function that the developer can use to process the data from the corresponding sensor. Code Sample 11 shows how to set the members of a structure of type TppBarometerMessageStruct before passing a pointer of this structure to the function tpp\_process\_barometer(). After calling the function tpp\_process\_barometer(), the developer checks that the function returns TPP\_SUCCESS which indicates that the call was successful. It is recommended to check on the return value when calling any of the API functions to ensure that the call is successful before proceeding.

Code Sample Processing the Barometer Data

TppReturnStatusEnum tpp\_return\_status**;**

tpp\_barometer\_message**.**height **=** user\_barometer\_height**;**

tpp\_barometer\_message**.**height\_standard\_deviation\_ **=** sqrt**(**user\_barometer\_height\_variance**);**

tpp\_return\_status **=** tpp\_process\_barometer**(**tpp\_handle,**&**tpp\_barometer\_message **);**

**if(** tpp\_return\_status **!=** TPP\_SUCCESS **)**

**{**

printf**(**"[ERROR] tpp\_process\_barometer() failed\n"**);**

**}**

The following code samples show how to use the library depending on the method by which the developer acquires the GNSS and sensors’ data.

In event-based applications, the developer registers specific events and the corresponding event handlers to get the GNSS and sensors’ data. When the data from any of the sensors is available, the developer will fill a variable of the corresponding sensor message and will pass it to the corresponding tpp\_process\_SSS()function. Code Sample 12 shows an example of using the library in an event-based application.

Code Sample Data processing using event handlers

user\_imu\_data\_event\_handler**(**event e**)**

**{**

**...**

tpp\_return\_status **=**

tpp\_advance\_navigation\_step**(**tpp\_handle, **&**tpp\_imu\_message **,** **&**solution\_structure**,**

**&**packet\_structure**);**

**...**

**}**

user\_gnss\_data\_event\_handler**(**event e**)**

**{**

**...**

tpp\_return\_status **=** tpp\_process\_gnss\_pvt**(**tpp\_handle, **&**tpp\_gnss\_pvt\_message **);**

**...**

**}**

user\_barometer\_data\_event\_handler**(**event e**)**

**{**

**...**

tpp\_return\_status **=** tpp\_process\_barometer**(**tpp\_handle, **&**tpp\_barometer\_message **);**

**...**

**}**

The same applies for multi-threaded applications that do not use events to acquire the sensors data which is the case when multiple threads exist where data is acquired from one sensor in each thread.

If one of the threads acquires multiple sensors data at the same epoch, then the order of calling the processing functions matter where the tpp\_advance\_navigation\_step() should be called last. This case is presented in Code Sample 13.

Code Sample Data processing when multiple sensors’ data is available at the same time

user\_imu\_barometer\_thread**(...)**

**{**

**...**

tpp\_return\_status **=** tpp\_process\_barometer**(**tpp\_handle, **&**tpp\_barometer\_message **);**

**...**

tpp\_return\_status **=**

tpp\_advance\_navigation\_step**(**tpp\_handle, **&**tpp\_imu\_message **,** **&**solution\_structure**,**

**&**packet\_structure**);**

**...**

**}**

user\_gnss\_data\_thread**(...)**

**{**

**...**

tpp\_return\_status **=** tpp\_process\_gnss\_pvt**(**tpp\_handle, **&**tpp\_gnss\_pvt\_message **);**

**...**

**}**

user\_magnetometer\_data\_thread**(...)**

**{**

**...**

tpp\_return\_status **=** tpp\_process\_magnetometer**(**tpp\_handle, **&**tpp\_magnetometer\_message **);**

**...**

**}**

### Output Array Parsing

When the developer calls tpp\_advance\_navigation\_step() and if TPP\_SUCCESS is returned. The developer can expect a valid output byte array which holds the navigation solution according to the structure specified in Section 7.23.1 that will be used to parse the output byte array. First, the developer can check that the three header bytes and the checksum are valid. The output packet is structured in this way, so that the developer can use this structure directly to write it to a file or send it over a communication channel.

Once the developer validated the byte array, the developer can start with parsing the packet according to the packet structure in Section 7.23.1.

Instead of parsing the output byte array, the developer can use the output of type TppSolutionStruct which provides direct access to the nine states of the integrated navigation solution along with the platform heading.

## Output Array Entities[[3]](#footnote-3)

The Ids for the different entities that can be specified with the ‘Output per Epoch’ packet along with the entities’ structures are defined below.

### Time

Entity Id: **0x00EA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | timetag\_ | DOUBLE64 | sec |  |

### Position

Entity Id: **0x0016**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | latitude\_ | DOUBLE64 | deg |  |
| 2 | longitude\_ | DOUBLE64 | deg |  |
| 3 | height\_ | FLOAT32 | m |  |

### Position Standard Deviation

Entity Id: **0x00BD**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 2 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 3 | height\_standard\_deviation\_ | FLOAT32 | m |  |

### Velocity

Entity Id: **0x00A2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 2 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 3 | velocity\_down\_ | FLOAT32 | m/sec |  |

### Velocity Standard Deviation

Entity Id: **0x005F**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 2 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 3 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |

### Attitude

Entity Id: **0x00E9**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | roll\_ | FLOAT32 | deg |  |
| 2 | pitch\_ | FLOAT32 | deg |  |
| 3 | heading\_ | FLOAT32 | deg | Device heading |

### Attitude Standard Deviation

Entity Id: **0x0060**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | roll\_standard\_deviation\_ | FLOAT32 | deg |  |
| 2 | pitch\_standard\_deviation\_ | FLOAT32 | deg |  |
| 3 | heading\_standard\_deviation\_ | FLOAT32 | deg | Device heading standard deviation |

### Accelerometer Bias

Entity Id: **0x006F**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | bias\_accelerometer\_x\_ | FLOAT32 | m/sec/sec |  |
| 2 | bias\_accelerometer\_y\_ | FLOAT32 | m/sec/sec |  |
| 3 | bias\_accelerometer\_z\_ | FLOAT32 | m/sec/sec |  |

### ­Gyroscope Bias

Entity Id: **0x002E**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | bias\_gyroscope\_x\_ | FLOAT32 | deg/sec |  |
| 2 | bias\_gyroscope\_y\_ | FLOAT32 | deg/sec |  |
| 3 | bias\_gyroscope\_z\_ | FLOAT32 | deg/sec |  |

### Heading Misalignment

Entity Id: **0x004D**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | heading\_misalignment\_ | FLOAT32 | deg |  |

### Platform Heading

Entity Id: **0x00B3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | platform\_heading\_ | FLOAT32 | deg |  |

### Stride Information

Entity Id: **0x005E**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | stride\_distance\_ | FLOAT32 | m |  |
| 2 | stride\_velocity\_ | FLOAT32 | m/sec |  |

### Number of Steps

Entity Id: **0x0099**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | number\_of\_steps\_ | UINT32 | N/A | Number of steps for the whole navigation session. |

### Floor Number

Entity Id: **0x00DD**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | floor\_number\_ | Unit16 | N/A |  |

### Converged Gyroscope Bias

Entity Id: **0x002F**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | is\_valid\_ | UINT8 | N/A | 0x00: Invalid bias values. The values should not be used.  0x01: Valid bias values. The values can be saved to be used later on in the initialization of the gyroscope bias values. |
| 2 | bias\_gyroscope\_x\_ | FLOAT32 | deg/sec |  |
| 3 | bias\_gyroscope\_y\_ | FLOAT32 | deg/sec |  |
| 4 | bias\_gyroscope\_z\_ | FLOAT32 | deg/sec |  |

### Flags - Magnetometer and Barometer

Entity Id: **0x0091**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | flags\_ | UINT8 | N/A | **Bit[0]**:Barometer Data Available Flag  0: Barometer data is not available from user  1: Barometer data is available from user  **Bit[1]**:Magnetometer Data Available Flag  0: Magnetometer data is not available from user  1: Magnetometer data is available from user  **Bit[2]**:Barometer Data In Use Flag  0: Barometer data is not used in navigator  1: Barometer data is used in navigator  **Bit[3]**:Magnetometer In Use Flag  0: Magnetometer data is not used in navigator  1: Magnetometer data is used in navigator  **Bit[7:4]**:Reserved |

### Mode of Transit

Entity Id: **0x00C8**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | mode\_ | UINT8 | N/A | Current mode of transit  0x00: Driving  0x01: Walking  0x02: Elevator  0x03: Stairs  0x04: Escalator walking  0x05: Escalator standing  0x06: Fidgeting  0x07: Conveyer walking  0x08: Conveyer standing  0x09: Running  0x0A: Cycling |

### Use Case

Entity Id: **0x00EE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | use\_case\_ | UINT8 | N/A | 0x00: Automatic  0x01: Others  0x02: Pocket  0x03: Hand dangling  0x04: Phone on ear  ~~0x05: Purse~~ |

### Flags – GNSS, Speed and Static Status

Entity Id: **0x002B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | flags\_ | UINT8 | N/A | **Bit[0]**:Static Status Flag  0: In motion  1: Static  **Bit[1]**:Multiple Antenna Available Flag  0:Multiple antenna data is not available from user  1: Multiple Antenna Data is available from user  **Bit[2]**:GNSS/PVT Available Flag  0: GNSS/PVT data is not available from user  1: GNSS/PVT data is available from user  **Bit[3]**:Multiple Antenna Data In Use Flag  0:Multiple antenna data is not used in navigator  1: Multiple antenna data is used in navigator  **Bit[4]**:GNSS/PVT Data In Use Flag  0: GNSS/PVT data is not used in navigator  1: GNSS/PVT Data is used in navigator  **Bit[5]**:GNSS Available for One Second Flag  0: GNSS is not available for the current second  1: GNSS is available for the current second  **Bit[6]**:Speed Data Available Flag  0: Speed data is not available from user  1: Speed data is available from user  **Bit[7]**:Speed Data In Use Flag  0: Speed data is not used in navigator  1: Speed data is used in navigator |

### Flags – Navigation Phase

Entity Id: **0x00E7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | navigation\_phase\_ | INT8 | N/A | -1: Pre-alignment  0: Alignment  +1: Navigation/Available  +2: Navigation/Reliable  +3: Navigation/Vertical Alignment  +4: Navigation/Drive to Walk |

### Orientation Based On Pitch

Entity Id: **0x00DF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | orientation\_ | INT8 | N/A | -1: Vertical Down  0: Horizontal  +1: Vertical Up |

### Distance Travelled

Entity Id: **0x0097**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | distance\_ | FLOAT32 | m |  |

### DSL Height (For TPP/DSL only)

Entity Id: **0x00BC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | height\_ | FLOAT32 | m |  |
| 2 | height\_standard\_deviation\_ | FLOAT32 | m |  |

### Raw IMU Data\*

This entity is present in the packet when the user passes data to tpp\_advance\_navigation\_step()

Entity Id: **0x00A1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | timetag\_ | DOUBLE64 | sec | time-tag |
| 2 | data\_gyroscope\_x\_ | DOUBLE64 | deg/sec |  |
| 3 | data\_gyroscope\_y\_ | DOUBLE64 | deg/sec |  |
| 4 | data\_gyroscope\_z\_ | DOUBLE64 | deg/sec |  |
| 5 | data\_accelerometer\_x\_ | DOUBLE64 | m/sec/sec |  |
| 6 | data\_accelerometer\_y\_ | DOUBLE64 | m/sec/sec |  |
| 7 | data\_accelerometer\_z\_ | DOUBLE64 | m/sec/sec |  |

### GNSS PVT Data\*

This entity is present in the packet when the user passes data to tpp\_process\_gnss\_pvt()

Entity Id: **0x001C**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_timetag\_ | DOUBLE64 | sec | An absolute time-tag that increments according to the rate of the GNSS data provided. |
| 2 | latitude\_ | DOUBLE64 | deg |  |
| 3 | longitude\_ | DOUBLE64 | deg |  |
| 4 | height\_ | FLOAT32 | m |  |
| 5 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 6 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 7 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 8 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 10 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 11 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 12 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 13 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 14 | dop\_data\_available\_ | UINT8 | N/A | 0x00: DOP data is not available  0x01: DOP data is available |
| 15 | horizontal\_dop\_ | FLOAT32 | N/A |  |
| 16 | vertical\_dop\_ | FLOAT32 | N/A |  |

### Barometer Data\*

This entity is present in the packet when the user passes data to tpp\_process\_barometer()

Entity Id: **0x0083**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | height\_ | FLOAT32 | m |  |
| 2 | height\_standard\_deviation\_ | FLOAT32 | m |  |

### Magnetometer Data\*

This entity is present in the packet when the user passes data to tpp\_process\_magnetometer()

Entity Id: **0x0062**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | raw\_data\_available\_ | UINT8 | N/A | 0x00: raw data is not available  0x01: raw data is available |
| 2 | raw\_data\_x\_ | FLOAT32 | mG |  |
| 3 | raw\_data\_y\_ | FLOAT32 | mG |  |
| 4 | raw\_data\_z\_ | FLOAT32 | mG |  |
| 5 | raw\_data\_accuracy\_flag\_ | UINT8 | N/A | 0x00: Flag not available  0x01: Invalid data  0x02: Valid data |
| 6 | calibrated\_data\_available\_ | UINT8 | N/A | 0x00: calibrated data is not available  0x01: calibrated data is available |
| 7 | calibrated\_data\_x\_ | FLOAT32 | mG |  |
| 8 | calibrated\_data\_y\_ | FLOAT32 | mG |  |
| 9 | calibrated\_data\_z\_ | FLOAT32 | mG |  |
| 10 | calibrated\_data\_accuracy\_flag\_ | UINT8 | N/A | 0x00: Flag not available  0x01:Unreliable  0x02: Low Accuracy  0x03: Medium Accuracy  0x04: High Accuracy |
| 11 | calibration\_status\_changed\_flag\_ | UINT8 | N/A | Specifies if the status of the calibration changed when the magnetometer providing the calibrated data is reset or when the the calibrated signals are saturated.  0x00: Flag not available  0x01: No change in status  0x02: Calibration status changed |
| 12 | heading\_available\_ | UINT8 | N/A | 0x00: heading data is not available  0x01: heading data is available |
| 13 | heading\_ | FLOAT32 | deg | The heading is with respect to the True North (i.e. The heading is corrected with the magnetic declination angle) and not the Magnetic North. |
| 14 | heading\_standard\_deviation\_ | FLOAT32 | deg |  |

### Speed Data\*

This entity is present in the packet when the user passes data to tpp\_process\_speed().

Entity Id: **0x00A7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | speed\_ | FLOAT32 | m/sec |  |
| 2 | reserved\_1\_ | UINT8 | N/A |  |
| 2 | reverse\_ | UINT8 | N/A | 0x00: Unavailable  0x01: Forward  0x02: Reverse |

### Operator 2D Position Data\*

This entity is present in the packet when the user passes data to tpp\_process\_2d\_position() with the source being operator.

Entity Id: **0x0043**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | latitude\_ | DOUBLE64 | deg |  |
| 2 | longitude\_ | DOUBLE64 | deg |  |

### Wireless Data\*

This entity is present in the packet when the user passes data to tpp\_process\_wireless() with the source being Wi-Fi.

Entity Id: **0x0056**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | source\_ | UINT8 | N/A | 0x01: Bluetooth  0x11: Wi-Fi Source (1)  0x12: Wi-Fi Source (2)  0x13: Wi-Fi Source (3) |
| 2 | latency\_available\_ | UINT8 | N/A | 0x00: Latency and the corresponding standard deviation are not available.  0x01: Latency and the corresponding standard deviation are available. |
| 3 | latency\_ | FLOAT32 | sec | Delay between the time the Wi-Fi 2D position is requested and the time the position is received and sent to the navigator.  If there is no delay, the user can use zero. |
| 4 | latency\_standard\_deviation\_ | FLOAT32 | sec |  |
| 5 | position\_2d\_available\_ | UINT8 | N/A | 0x00: 2d position and the corresponding standard deviation are not available  0x01: 2d position and the corresponding standard deviation are available |
| 6 | latitude\_ | DOUBLE64 | deg |  |
| 7 | longitude\_ | DOUBLE64 | deg |  |
| 8 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 10 | height\_available\_ | UINT8 | N/A | 0x00: Height and the corresponding standard deviation are not available  0x01: Height and the corresponding standard deviation are available |
| 11 | height\_ | FLOAT32 | m |  |
| 12 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 13 | velocity\_2d\_available\_ | UINT8 | N/A | 0x00: 2d velocity and the corresponding standard deviation are not available  0x01: 2d velocity and the corresponding standard deviation are available |
| 14 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 15 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 16 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 17 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 18 | velocity\_down\_available\_ | UINT8 | N/A | 0x00: Down velocity and the corresponding standard deviation are not available  0x01: Down velocity and the corresponding standard deviation are available |
| 19 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 20 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 21 | platform\_heading\_available\_ | UINT8 | N/A | 0x00: Platform heading and the corresponding standard deviation are not available  0x01: Platform heading and the corresponding standard deviation are available |
| 22 | platform\_heading\_ | FLOAT32 | deg |  |
| 23 | platform\_heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 24 | floor\_information\_available\_ | UINT8 | N/A | 0x00: Floor information is not available  0x01: Floor information is available |
| 24 | floor\_number\_ | INT16 | N/A |  |

### Floor Information Data\*

This entity is present in the packet when tpp\_process\_floor\_information() is called.

Entity Id: **0x004A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | floor\_number\_ | INT16 | N/A |  |
| 2 | height\_between\_floors\_ | FLOAT32 | m |  |

### Call Information Data\*

This entity is present in the packet when tpp\_process\_call\_information() is called.

Entity Id: **0x00AE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | is\_call\_active\_ | UINT8 | N/A | 0x00: Call Inactive  0x01: Call Active |
| 2 | is\_proximity\_detected\_ | UINT8 | N/A | 0x00: No Proximity Detected  0x01: Proximity Detected |
| 3 | is\_speaker\_active\_ | UINT8 | N/A | 0x00: Phone speaker Inactive  0x01: Phone speaker Active |
| 4 | headset\_type\_ | UINT8 | N/A | 0x00: None  0x01: Wired or wireless earphone |

### Set Mode of Transit Event Data\*

This entity is present in the packet when tpp\_set\_mode\_of\_transit() is called.

Entity Id: **0x001D**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | mode\_ | UINT8 | N/A | In-run change of user mode  0x00: Driving  0x01: Walking  0x02: Automatic  0x03: Running  0x04: Cycling |

### Platform Heading Data\*

This entity is present in the packet when tpp\_process\_platform\_heading() is called.

Entity Id: **0x00BA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | heading\_ | FLOAT32 | deg |  |
| 2 | heading\_standard\_deviation\_ | FLOAT32 | deg |  |

### Device Heading Data\*

This entity is present in the packet when tpp\_set\_device\_heading() is called.

Entity Id: **0x00BB**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | device\_heading\_available\_ | UINT8 | N/A | **0x00:** Device heading value is not available directly and will be computed from the **platform\_heading\_** and **misalignment\_angle\_** fields.  **0x01**: Device heading value is available directly from the **device\_heading\_** field. |
| 2 | device\_heading\_ | FLOAT32 | deg |  |
| 3 | platform\_heading\_ | FLOAT32 | deg |  |
| 4 | misalignment\_angle\_ | FLOAT32 | deg |  |

### 9-DOF Quaternions Data\*

This entity is present in the packet when tpp\_process\_9dof\_quaternions() is called.

Entity Id: **0x007A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | value\_0\_ | FLOAT32 | N/A | The scalar component of the quaternion. |
| 2 | value\_1\_ | FLOAT32 | N/A | The 1st vector component of the quaternion. |
| 3 | value\_2\_ | FLOAT32 | N/A | The 2nd vector component of the quaternion. |
| 4 | value\_3\_ | FLOAT32 | N/A | The 3rd vector component of the quaternion. |

### IMU Biases Data\*

This entity is present in the packet when tpp\_process\_imu\_biases() is called.

Entity Id: **0x0008**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | bias\_gyroscope\_source\_ | UINT8 | N/A | 0: Biases Not Available (Discard values)  **Other values to be defined by ISJ and ICA** |
| 2 | bias\_gyroscope\_accuracy\_ | FLOAT32 | deg/sec | Accuracy for gyroscope biases |
| 3 | bias\_gyroscope\_x\_ | FLOAT32 | deg/sec | Bias for gyroscope-x |
| 4 | bias\_gyroscope\_y\_ | FLOAT32 | deg/sec | Bias for gyroscope-y |
| 5 | bias\_gyroscope\_z\_ | FLOAT32 | deg/sec | Bias for gyroscope-z |
| 6 | bias\_accelerometer\_source\_ | UINT8 | N/A | 0: Biases Not Available (Discard values)  **Other values to be defined by ISJ and ICA** |
| 7 | bias\_accelerometer\_accuracy\_ | FLOAT32 | m/sec/sec | Accuracy for accelerometer biases |
| 8 | bias\_accelerometer\_x\_ | FLOAT32 | m/sec/sec | Bias for accelerometer-x |
| 9 | bias\_accelerometer\_y\_ | FLOAT32 | m/sec/sec | Bias for accelerometer-y |
| 10 | bias\_ accelerometer\_z\_ | FLOAT32 | m/sec/sec | Bias for accelerometer-z |

### Process Misalignment Data\*

This entity is present in the packet when tpp\_process\_misalignment() is called.

Entity Id: **0x00A6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | angle\_ | FLOAT32 | deg |  |
| 2 | angle\_standard\_deviation\_ | FLOAT32 | deg |  |

### Set ZUPT Mode Event Data\*

This entity is present in the packet when tpp\_set\_zupt\_mode() is called.

Entity Id: **0x009F**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | zupt\_mode\_ | UINT8 | N/A | 0x00: Automatic  0x01: On  0x02: Off |

### Set Misalignment Event Data\*

This entity is present in the packet when tpp\_set\_misalignment() is called.

Entity Id: **0x00A4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | estimation\_on\_ | UINT8 | N/A | 0x00: Automatic misalignment estimation turned off  0x01: Automatic misalignment estimation turned on |
| 2 | angle\_available\_ | UINT8 | N/A | 0x00: Misalignment angle is not available  0x01: Misalignment angle is available |
| 3 | angle\_ | FLOAT32 | deg |  |

### Set Magnetometer Calibration Status Event Data\*

This entity is present in the packet when tpp\_set\_magnetometer\_calibration() is called.

Entity Id: **0x00E4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | magnetometer\_calibration\_ | UINT8 | N/A | 0x00: calibration disabled  0x01: calibration enabled |

### Set Use Case Event Data\*

This entity is present in the packet when tpp\_set\_use\_case() is called.

Entity Id: **0x0098**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | use\_case\_ | UINT8 | N/A | 0x00: Automatic  0x01: Torso and hand viewing  0x02: Pocket  0x03: Hand swinging  0x04: Arm  ~~0x05: Purse~~ |

### 6-DOF Quaternions Data\*

This entity is present in the packet when tpp\_process\_6dof\_quaternions() is called.

Entity Id: **0x007B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | value\_0\_ | FLOAT32 | N/A | The scalar component of the quaternion. |
| 2 | value\_1\_ | FLOAT32 | N/A | The 1st vector component of the quaternion. |
| 3 | value\_2\_ | FLOAT32 | N/A | The 2nd vector component of the quaternion. |
| 4 | value\_3\_ | FLOAT32 | N/A | The 3rd vector component of the quaternion. |

### Venue Map Data\*

This entity is present in the packet when tpp\_process\_map\_information() is called.

Entity Id: **0x004B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | position\_2d\_available\_ | UINT8 | N/A | 0x00: 2d position and the corresponding standard deviation are **not available**  0x01: 2d position and the corresponding standard deviation are **available but not reliable**  0x02: 2d position and the corresponding standard deviation are **available and reliable** |
| 2 | latitude\_ | DOUBLE64 | deg |  |
| 3 | longitude\_ | DOUBLE64 | deg |  |
| 4 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 5 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 6 | height\_available\_ | UINT8 | N/A | 0x00: Height and the corresponding standard deviation are not available  0x01: Height and the corresponding standard deviation are available |
| 7 | height\_ | FLOAT32 | m |  |
| 8 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | platform\_heading\_available\_ | UINT8 | N/A | 0x00: Platform heading and the corresponding standard deviation are **not available**  0x01: Platform heading and the corresponding standard deviation are **available but not reliable**  0x01: Platform heading and the corresponding standard deviation are **available and reliable** |
| 10 | platform\_heading\_ | FLOAT32 | deg |  |
| 11 | platform\_heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 12 | map\_entity\_available\_ | UINT8 | N/A | 0x00: Map entity is not available  0x01: Map entity is available |
| 13 | map\_entity\_ | UINT16 | N/A | Specifies an entity in the map. Examples for map entities are stairs, elevators, escalators, types of rooms, and others. |
| 15 | step\_length\_scale\_available\_ | UINT8 | N/A | Flag to notify if both fields (#16) Step Length Scale and (#17) Use case for Step length scale are available or not.  0x00: Fields are not available  0x01: Fields are available |
| 16 | step\_length\_scale\_ | FLOAT32 | N/A |  |
| 17 | use\_case\_for\_step\_length\_scale\_ | UINT8 | N/A |  |
| 18 | mode\_of\_motion\_for\_step\_length\_scale\_ | UINT8 | N/A |  |

### Anchor Point Data\*

This entity is present in the packet when tpp\_add\_anchor\_point() is called.

Entity Id: **0x00FE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | type\_ | UINT8 | N/A | Anchor Point Type.  **0x01**: Start Anchor Point.  **0x02**: End Anchor Point. |
| 2 | position\_available\_ | UNIT8 | N/A |  |
| 3 | latitude\_ | DOUBLE64 | deg |  |
| 4 | longitude\_ | DOUBLE64 | deg |  |
| 5 | height\_ | FLOAT32 | m |  |
| 6 | heading\_available\_ | UNIT8 | N/A |  |
| 7 | heading\_ | FLOAT32 | deg |  |

### Synchronization Event Data\*

This entity is present in the packet when tpp\_add\_synchronization\_event() is called.

Entity Id: **0x00FF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | event\_number\_ | UINT64 | N/A |  |

### Orientation Based On Pitch Data\*

This entity is present in the packet when tpp\_set\_orientation\_based\_on\_pitch() is called.

Entity Id: **0x00FD**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | orientation\_ | INT8 | N/A |  |

### Multi-Device Position Velocity Data\*

When tpp\_multi\_device\_process\_position\_velocity() is called, the data passed to this function is constructed in the form of one common entity and 2 or more data entities. When parsing the packet the user can check the availability of the position and velocity common entity first. If it exists, the user can parse the number of devices. The number of position and velocity data entities that will exist in the packet is equal to the number of devices.

#### Common Entity

Entity Id: **0x0101**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | number\_of\_devices\_ | UINT8 | N/A |  |
| 2 | latency\_available\_ | UINT8 | N/A | 0x00: Latency is not available.  0x01: Latency is available |
| 3 | latency\_ | FLOAT32 | sec | Delay between the time the information is computed on the sender device and the time the information is processed on the receiver device.  If there is no delay, the user can use zero. |

#### Data per Device Entity

Entity Id: **0x0102**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | device\_identifier\_ | UINT64 | N/A | A unique identifier for the device that is sending this information.  When sending the device its own information, the value of the **device\_identifier\_** should be **zero**. |
| 2 | device\_type\_ | UINT8 | N/A | **0**: Phone  **1**: Tablet  **2**: Head-Mount  **3**: Watch |
| 3 | navigation\_phase\_ | INT8 | N/A | Equivalent to the values taken by the flag **navigation\_phase\_** in TppSolutionStruct |
| 4 | gnss\_in\_use\_ | UINT8 | N/A | 0x00: GNSS data was not used in computing the navigation solution within the last 2 seconds.  0x01: GNSS data was used in computing the navigation solution within the last 2 seconds. |
| 5 | magnetometer\_in\_use\_ | UINT8 | N/A | 0x00: Magnetometer data was not used in computing the navigation solution within the last 2 seconds.  0x01: Magnetometer data was used in computing the navigation solution within the last 2 seconds. |
| 6 | barometer\_in\_use\_ | UINT8 | N/A | 0x00: Barometer data was not used in computing the navigation solution within the last 2 seconds.  0x01: Barometer data was used in computing the navigation solution within the last 2 seconds. |
| 7 | speed\_in\_use\_ | UINT8 | N/A | 0x00: Speed data was not used in computing the navigation solution within the last 2 seconds.  0x01: Speed data was used in computing the navigation solution within the last 2 seconds. |
| 8 | use\_case\_ | UINT8 | N/A |  |
| 9 | mode\_of\_transit\_ | UINT8 | N/A |  |
| 10 | latitude\_ | DOUBLE64 | deg |  |
| 11 | longitude\_ | DOUBLE64 | deg |  |
| 12 | height\_ | FLOAT32 | m |  |
| 13 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 14 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 15 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 16 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 17 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 18 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 19 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 20 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 21 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 22 | number\_of\_steps\_ | UINT32 | N/A |  |

### Multi-Device Heading Data\*

When tpp\_multi\_device\_process\_heading() is called, the data passed to this function is constructed in the form of one common entity and 2 or more data entities. When parsing the packet the user can check the availability of the heading common entity first. If it exists, the user can parse the number of devices. The number of heading data entities that will exist in the packet is equal to the number of devices.

#### Common Entity

Entity Id: **0x0103**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | number\_of\_devices\_ | UINT8 | N/A |  |
| 2 | latency\_available\_ | UINT8 | N/A | 0x00: Latency is not available.  0x01: Latency is available |
| 3 | latency\_ | FLOAT32 | sec | Delay between the time the information is computed on the sender device and the time the information is processed on the receiver device.  If there is no delay, the user can use zero. |

#### Data per Device Entity

Entity Id: **0x0104**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | device\_identifier\_ | UINT64 | N/A | A unique identifier for the device that is sending this information.  When sending the device its own information, the value of the **device\_identifier\_** should be **zero**. |
| 2 | device\_type\_ | UINT8 | N/A | **0**: Phone  **1**: Tablet  **2**: Head-Mount  **3**: Watch |
| 3 | navigation\_phase\_ | INT8 | N/A | Equivalent to the values taken by the flag **navigation\_phase\_** in TppSolutionStruct |
| 4 | gnss\_in\_use\_ | UINT8 | N/A | 0x00: GNSS data was not used in computing the navigation solution within the last 2 seconds.  0x01: GNSS data was used in computing the navigation solution within the last 2 seconds. |
| 5 | magnetometer\_in\_use\_ | UINT8 | N/A | 0x00: Magnetometer data was not used in computing the navigation solution within the last 2 seconds.  0x01: Magnetometer data was used in computing the navigation solution within the last 2 seconds. |
| 6 | barometer\_in\_use\_ | UINT8 | N/A | 0x00: Barometer data was not used in computing the navigation solution within the last 2 seconds.  0x01: Barometer data was used in computing the navigation solution within the last 2 seconds. |
| 7 | speed\_in\_use\_ | UINT8 | N/A | 0x00: Speed data was not used in computing the navigation solution within the last 2 seconds.  0x01: Speed data was used in computing the navigation solution within the last 2 seconds. |
| 8 | use\_case\_ | UINT8 | N/A |  |
| 9 | mode\_of\_transit\_ | UINT8 | N/A |  |
| 10 | device\_heading\_ | FLOAT32 | deg |  |
| 11 | platform\_heading\_ | FLOAT32 | deg |  |
| 12 | heading\_misalignment\_ | FLOAT32 | deg |  |
| 13 | heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 14 | position\_north\_standard\_deviation | FLOAT32 | m |  |
| 15 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 16 | number\_of\_steps\_ | UINT32 | N/A |  |

### Multi-Device Secondary GNSS PVT Data\*

This entity is present in the packet when the user passes data to tpp\_multi\_device\_process\_secondary\_gnss\_pvt()

Entity Id: **0x0105**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_timetag\_ | DOUBLE64 | sec | An absolute time-tag that increments according to the rate of the GNSS data provided. |
| 2 | latitude\_ | DOUBLE64 | deg |  |
| 3 | longitude\_ | DOUBLE64 | deg |  |
| 4 | height\_ | FLOAT32 | m |  |
| 5 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 6 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 7 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 8 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 10 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 11 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 12 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 13 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 14 | dop\_data\_available\_ | UINT8 | N/A | 0x00: DOP data is not available  0x01: DOP data is available |
| 15 | horizontal\_dop\_ | FLOAT32 | N/A |  |
| 16 | vertical\_dop\_ | FLOAT32 | N/A |  |
| 17 | secondary\_device\_navigation\_phase\_ | INT8 | N/A | Equivalent to the values taken by the flag **navigation\_phase\_** in TppSolutionStruct |

### Multi-Device GNSS PVT Data Source\*

This entity is present in the packet to indicate the source of the GNSS PVT data when more than one source is available. The source can be the device’s own data (i.e. main device data which can be referenced from the entity in Section 9.25) or other device’s data (i.e. secondary device data, such as the master device in a multi-device system, which can be referenced from the entity in Section 9.50)

Entity Id: **0x0106**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | data\_source\_ | UINT8 | N/A | 0x00: Secondary Device  0x01: Main Device |

### In-run Magnetometer Calibration Information^ [Internal Debugging Only]

Entity Id: **0x000F**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | heading\_ | FLOAT32 | deg | The heading derived from the automatic magnetometer calibration algorithms. |
| 2 | used\_status\_ | UINT8 | N/A |  |
| 3 | calibration\_type\_ | UINT8 | N/A | Flag to indicate whether 2D or 3D calibration is used.  0x00 : No Calibration  0x01 : 2D  0x02 : 3D  0x03 : 3D gravity |
| 4 | used\_signals\_type\_ | UINT8 | N/A | Flag to indicate whether the raw or calibrated magnetometer signals are used in the calibration.  0x00: raw data  0x01: calibrated data |
| 5 | collection\_completed\_2d\_ | UINT8 | N/A | Flag to indicate if 2D collection is completed.  0x00: Not Completed  0x01: Completed |
| 6 | collection\_completed\_3d\_ | UINT8 | N/A | Flag to indicate if 3D collection is completed.  0x00: Not Completed  0x01: Completed |
| 7 | collection\_completed\_gravity\_3d\_ | UINT8 | N/A | Flag to indicate if 3D gravity collection is completed.  0x00: Not Completed  0x01: Completed |
| 8 | post\_calibration\_checks\_failure\_2d\_ | UINT8 | N/A | Flag to indicate that the 2D post-calibration checks failed.  0x00: No Failure  0x01:Scale Factor check failed 0x02:Heading check failed  0x03:Signal check failed |
| 9 | post\_calibration\_checks\_failure\_3d\_ | UINT8 | N/A | Flag to indicate that the 3D post-calibration check failed.  0x00: No Failure  0x01:Scale Factor Check Failed 0x02:Heading Check Failed  0x03:Signal Check Failed |
| 10 | post\_calibration\_checks\_failure\_gravity\_3d\_ | UINT8 | N/A | Flag to indicate that the 3D gravity post-calibration check failed.  0x00: No Failure  0x01:Scale Factor Check Failed 0x02:Heading Check Failed  0x03:Signal Check Failed |
| 11 | periodic\_checks\_failure\_2d\_ | UINT8 | N/A | Flag to indicate that the 2D periodic check failed.  0x00: No Failure  0x01: Signal Check Failed |
| 12 | periodic\_checks\_failure\_3d\_ | UINT8 | N/A | Flag to indicate that the 3D periodic check failed.  0x00: No Failure  0x01: Signal Check Failed |
| 13 | periodic\_checks\_failure\_gravity\_3d\_ | UINT8 | N/A | Flag to indicate that the 3D gravity periodic check failed.  0x00: No Failure  0x01: Signal Check Failed |
| 14 | mag\_bias\_x\_ | FLOAT32 | mG |  |
| 15 | mag\_bias\_y\_ | FLOAT32 | mG |  |
| 16 | mag\_bias\_z\_ | FLOAT32 | mG |  |
| 17 | mag\_scale\_factor\_x\_ | FLOAT32 | N/A |  |
| 18 | mag\_scale\_factor\_y\_ | FLOAT32 | N/A |  |
| 19 | mag\_scale\_factor \_z\_ | FLOAT32 | N/A |  |
| 20 | pitch\_roll\_sector\_id\_ | INT16 | N/A | Pitch-roll sector ID |
| 21 | pitch\_defining\_angle\_for\_sector\_ | FLOAT32 | deg | Pitch defining angle for pitch-roll sector |
| 22 | roll\_defining\_angle\_for\_sector\_ | FLOAT32 | deg | Roll defining angle for pitch-roll sector |

### Internal Debugging Information (1)^ [Internal Debugging Only]

Entity Id: **0x008C**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | imu\_sample\_processing\_time\_ | FLOAT32 | µSec | The time taken, in micro-seconds, by the core main function in the navigation algorithm to compute the integrated navigation solution. |
| 2 | radius\_of\_rotation\_ | DOUBLE64 | m |  |
| 3 | radius\_of\_rotation\_x\_ | DOUBLE64 | m |  |
| 4 | radius\_of\_rotation\_y\_ | DOUBLE64 | m |  |
| 5 | radius\_of\_rotation\_z\_ | DOUBLE64 | m |  |
| 6 | orientation\_based\_on\_pitch\_ | INT8 | N/A |  |
| 7 | automatic\_ear\_flag\_ | UINT8 | N/A |  |
| 8 | sensors\_ear\_flag\_ | UINT8 | N/A |  |
| 9 | walking\_fidgeting\_flag\_ | UINT8 | N/A |  |
| 10 | height\_changing\_modes\_ | INT16 | N/A |  |
| 11 | children\_info\_ | UINT8 | N/A |  |
| 12 | misalignement\_driving\_gps\_ | FLOAT32 | deg |  |
| 13 | misalignement\_driving\_radius\_ | FLOAT32 | deg |  |
| 14 | misalignement\_driving\_kalman\_filter\_ | FLOAT32 | deg |  |
| 15 | misalignment\_driving\_sensors\_ | FLOAT32 | deg |  |
| 16 | misalignment\_driving\_sensors\_combined\_ | FLOAT32 | deg |  |
| 17 | misalignment\_driving\_children\_ | FLOAT32 | deg |  |
| 18 | misalignment \_driving\_sensors\_align\_average\_ | FLOAT32 | deg |  |
| 19 | misalignment\_driving\_sensors\_ratio\_ | FLOAT32 | N/A |  |
| 20 | misalignment\_driving\_sensors\_count\_ | INT16 | N/A |  |
| 21 | misalignment\_walking\_pca\_p1\_ | FLOAT32 | deg |  |
| 22 | misalignment\_walking\_pca\_p2\_ | FLOAT32 | deg |  |
| 23 | misalginment\_walking\_outdoors\_ | FLOAT32 | deg |  |
| 24 | misalginment\_walking\_averaged\_ | FLOAT32 | deg |  |
| 25 | misalginment\_walking\_snap\_ | FLOAT32 | deg |  |
| 26 | primary\_device\_heading\_ | FLOAT32 | deg |  |
| 27 | roll\_misalignment\_ | FLOAT32 | deg |  |
| 28 | pitch\_misalignment\_ | FLOAT32 | deg |  |
| 29 | internal\_roll\_ | DOUBLE64 | deg |  |
| 30 | internal\_pitch\_ | DOUBLE64 | deg |  |
| 31 | accelerometer\_roll\_ | DOUBLE64 | deg |  |
| 32 | accelerometer\_pitch\_ | DOUBLE64 | deg |  |
| 33 | leveled\_data\_gyroscope\_x\_ | DOUBLE64 | deg/sec |  |
| 34 | leveled\_data\_gyroscope\_y\_ | DOUBLE64 | deg/sec |  |
| 35 | leveled\_data\_gyroscope\_z\_ | DOUBLE64 | deg/sec |  |
| 36 | leveled\_data\_accelerometer\_x\_ | FLOAT32 | m/sec/sec |  |
| 37 | leveled\_data\_accelerometer\_y\_ | FLOAT32 | m/sec/sec |  |
| 38 | leveled\_data\_accelerometer\_z\_ | FLOAT32 | m/sec/sec |  |
| 39 | magnitude\_compensated\_gyroscope\_ | DOUBLE64 | deg/sec |  |
| 40 | vertical\_speed\_from\_height\_ | DOUBLE64 | m/sec |  |
| 41 | step\_time\_ | DOUBLE64 | sec |  |
| 42 | step\_frequency\_ | DOUBLE64 | 1/sec |  |
| 43 | acceleration\_variance\_per\_step\_ | DOUBLE64 | m\*m/sec/sec/sec/sec |  |
| 44 | step\_leveled\_accelerometer\_value\_ | DOUBLE64 | m/sec/sec |  |
| 45 | leveled\_smoothed\_vertical\_accelerometer\_data\_ | FLOAT32 | m/sec/sec |  |
| 46 | leveled\_smoothed\_vertical\_acceleration\_ | FLOAT32 | m/sec/sec |  |
| 47 | pdr\_latitude\_ | DOUBLE64 | deg |  |
| 48 | pdr\_longitude\_ | DOUBLE64 | deg |  |
| 49 | attitude\_filter\_roll\_ | FLOAT32 | deg |  |
| 50 | attitude\_filter\_ptich\_ | FLOAT32 | deg |  |
| 51 | attitude\_filter\_heading\_ | FLOAT32 | deg |  |
| 52 | attitude\_filter\_roll\_standard\_deviation\_ | FLOAT32 | deg |  |
| 53 | attitude\_filter\_ptich\_ standard\_deviation\_ | FLOAT32 | deg |  |
| 54 | attitude\_filter\_heading\_ standard\_deviation\_ | FLOAT32 | deg |  |
| 55 | attitude\_filter\_bias\_gyroscope\_x\_ | FLOAT32 | deg/sec |  |
| 56 | attitude\_filter\_bias\_gyroscope\_y\_ | FLOAT32 | deg/sec |  |
| 57 | attitude\_filter\_bias\_gyroscope\_z\_ | FLOAT32 | deg/sec |  |
| 58 | attitude\_filter\_bias\_standard\_deviation\_gyroscope\_x\_ | FLOAT32 | deg/sec |  |
| 59 | attitude\_filter\_bias\_standard\_deviation\_gyroscope\_y\_ | FLOAT32 | deg/sec |  |
| 60 | attitude\_filter\_bias\_standard\_deviation\_gyroscope\_z\_ | FLOAT32 | deg/sec |  |
| 61 | declination\_angle\_ | FLOAT32 | deg |  |
| 62 | magnetometer\_derived\_heading\_from\_calibrated\_ | FLOAT32 | deg |  |
| 63 | magnetometer\_heading\_used\_ | FLOAT32 | deg |  |
| 64 | magnetometer\_heading\_source\_ | UINT8 | N/A | 0x00: Not Available  0x01: External heading available in the magnetometer input data.  0x02:Heading derived from quaternions  0x03: Heading derived from calibrated data.  0x04: Heading derived from in-run magnetometer calibration. |
| 65 | barometer\_filter\_height\_ | FLOAT32 | m |  |
| 66 | barometer\_filter\_height\_standard\_deviation\_ | FLOAT32 | m |  |
| 67 | quaternions\_derived\_roll\_ | FLOAT32 | deg |  |
| 68 | quaternions\_derived\_pitch\_ | FLOAT32 | deg |  |
| 69 | quaternions\_derived\_heading\_ | FLOAT32 | deg |  |
| 70 | secondary\_device\_roll\_ | FLOAT32 | deg |  |
| 71 | secondary\_device\_pitch\_ | FLOAT32 | deg |  |
| 72 | Secondary\_ bias\_gyroscope\_x\_ | FLOAT32 | deg/sec |  |
| 73 | Secondary\_ bias\_gyroscope\_y\_ | FLOAT32 | deg/sec |  |
| 74 | Secondary\_ bias\_gyroscope\_z\_ | FLOAT32 | deg/sec |  |
| 75 | Secondary\_ bias\_accelerometer\_x\_ | FLOAT32 | m/sec/sec |  |
| 76 | Secondary\_ bias\_accelerometer \_y\_ | FLOAT32 | m/sec/sec |  |
| 77 | Secondary\_ bias\_accelerometer \_z\_ | FLOAT32 | m/sec/sec |  |
| 78 | step\_distance\_sensors\_ | FLOAT32 | m |  |
| 79 | step\_speed\_sensors\_ | FLOAT32 | m/sec |  |
| 80 | step\_distance\_gnss\_ | FLOAT32 | m |  |
| 81 | step\_speed\_gnss\_ | FLOAT32 | m/sec |  |

## DEPRECATED ENTITIES

### Magnetometer Data\* [Deprecated – As of version 3.0.0]

This entity is present in the packet when the user passes data to tpp\_process\_magnetometer()

Entity Id: **0x00B7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | raw\_data\_available\_ | UINT8 | N/A | 0x00: raw data is not available  0x01: raw data is available |
| 2 | raw\_data\_x\_ | FLOAT32 | mG |  |
| 3 | raw\_data\_y\_ | FLOAT32 | mG |  |
| 4 | raw\_data\_z\_ | FLOAT32 | mG |  |
| 5 | raw\_data\_accuracy\_flag\_ | INT8 | N/A | 0x00: Flag not available  0x01: Invalid  0x02: Valid |
| 6 | calibrated\_data\_available\_ | UINT8 | N/A | 0x00: calibrated data is not available  0x01: calibrated data is available |
| 7 | calibrated\_data\_x\_ | FLOAT32 | mG |  |
| 8 | calibrated\_data\_y\_ | FLOAT32 | mG |  |
| 9 | calibrated\_data\_z\_ | FLOAT32 | mG |  |
| 10 | calibrated\_data\_accuracy\_flag\_ | UINT8 | N/A | 0x00: Flag not available  0x01:Unreliable  0x02: Low Accuracy  0x03: Medium Accuracy  0x04: High Accuracy |
| 11 | heading\_available\_ | UINT8 | N/A | 0x00: heading data is not available  0x01: heading data is available |
| 12 | heading\_ | FLOAT32 | deg |  |
| 13 | heading\_standard\_deviation\_ | FLOAT32 | deg |  |

### Wi-Fi 2D Position Data\* [Deprecated – As of version 4.1.0]

This entity is present in the packet when the user passes data to tpp\_process\_2d\_position() with the source being Wi-Fi.

Entity Id: **0x0051**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | latitude\_ | DOUBLE64 | deg |  |
| 2 | longitude\_ | DOUBLE64 | deg |  |
| 3 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 4 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |

### Operator 2D Position Data\* [Deprecated as of version 4.1.0]

This entity is present in the packet when the user passes data to tpp\_process\_2d\_position() with the source being operator.

Entity Id: **0x0042**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | latitude\_ | DOUBLE64 | deg |  |
| 2 | longitude\_ | DOUBLE64 | deg |  |
| 3 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 4 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |

### Wi-Fi Data\* [Deprecated]

This entity is present in the packet when the user passes data to tpp\_process\_wifi() with the source being Wi-Fi.

Entity Id: **0x0055**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member** | **Data Type** | **Unit** | **Description** |
| 1 | latency\_available\_ | UINT8 | N/A | 0x00: Latency and the corresponding standard deviation are not available.  0x01: Latency and the corresponding standard deviation are available. |
| 2 | latency\_ | FLOAT32 | sec | Delay between the time the Wi-Fi 2D position is requested and the time the position is received and sent to the navigator.  If there is no delay, the user can use zero. |
| 3 | latency\_standard\_deviation\_ | FLOAT32 | sec |  |
| 4 | position\_2d\_available\_ | UINT8 | N/A | 0x00: 2d position and the corresponding standard deviation are not available  0x01: 2d position and the corresponding standard deviation are available |
| 5 | latitude\_ | DOUBLE64 | deg |  |
| 6 | longitude\_ | DOUBLE64 | deg |  |
| 7 | position\_north\_standard\_deviation\_ | FLOAT32 | m |  |
| 8 | position\_east\_standard\_deviation\_ | FLOAT32 | m |  |
| 9 | height\_available\_ | UINT8 | N/A | 0x00: Height and the corresponding standard deviation are not available  0x01: Height and the corresponding standard deviation are available |
| 10 | height\_ | FLOAT32 | m |  |
| 11 | height\_standard\_deviation\_ | FLOAT32 | m |  |
| 12 | velocity\_2d\_available\_ | UINT8 | N/A | 0x00: 2d velocity and the corresponding standard deviation are not available  0x01: 2d velocity and the corresponding standard deviation are available |
| 13 | velocity\_north\_ | FLOAT32 | m/sec |  |
| 14 | velocity\_east\_ | FLOAT32 | m/sec |  |
| 15 | velocity\_north\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 16 | velocity\_east\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 17 | velocity\_down\_available\_ | UINT8 | N/A | 0x00: Down velocity and the corresponding standard deviation are not available  0x01: Down velocity and the corresponding standard deviation are available |
| 18 | velocity\_down\_ | FLOAT32 | m/sec |  |
| 19 | velocity\_down\_standard\_deviation\_ | FLOAT32 | m/sec |  |
| 20 | platform\_heading\_available\_ | UINT8 | N/A | 0x00: Platform heading and the corresponding standard deviation are not available  0x01: Platform heading and the corresponding standard deviation are available |
| 21 | platform\_heading\_ | FLOAT32 | deg |  |
| 22 | platform\_heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 23 | floor\_information\_available\_ | UINT8 | N/A | 0x00: Floor information is not available  0x01: Floor information is available |
| 24 | floor\_number\_ | INT16 | N/A |  |

## SUGGESTED ENTITIES

### Accelerometer Bias Standard Deviation

Entity Id: **0x0054**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | bias\_accelerometer\_standard\_deviation\_x\_ | FLOAT32 | m/sec/sec |  |
| 2 | bias\_accelerometer\_standard\_deviation\_y\_ | FLOAT32 | m/sec/sec |  |
| 3 | bias\_accelerometer\_standard\_deviation\_z\_ | FLOAT32 | m/sec/sec |  |

### Gyroscope Bias STD Outputs

Entity Id: **0x007E**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | BiasGyroXStdv | FLOAT32 | deg/sec |  |
| 2 | BiasGyroYStdv | FLOAT32 | deg/sec |  |
| 3 | BiasGyroZStdv | FLOAT32 | deg/sec |  |

### Corrected Accel Outputs:

Entity Id: **0x00E3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | CorrectedAccX | DOUBLE64 | m/sec/sec |  |
| 2 | CorrectedAccY | DOUBLE64 | m/sec/sec |  |
| 3 | CorrectedAccZ | DOUBLE64 | m/sec/sec |  |

### Corrected Gyro Outputs:

Entity Id: **0x00BE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | CorrectedGyroX | DOUBLE64 | deg/sec |  |
| 2 | CorrectedGyroY | DOUBLE64 | deg/sec |  |
| 3 | CorrectedGyroZ | DOUBLE64 | deg/sec |  |

### Declination Angle

Entity Id: **0x0050**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | declination\_angle\_ | FLOAT32 | deg |  |

### Mag Pre-Calibration Outputs:

Entity Id: **0x00DE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **DataType** | **Unit** | **Description** |
| 1 | CalibrationType | UINT8 | N/A |  |
| 2 | MagBiasX | DOUBLE64 | mG |  |
| 3 | MagBiasY | DOUBLE64 | mG |  |
| 4 | MagBiasZ | DOUBLE64 | mG |  |
| 5 | MagSfX | DOUBLE64 | N/A |  |
| 6 | MagSfY | DOUBLE64 | N/A |  |
| 7 | MagSfZ | DOUBLE64 | N/A |  |

### Mag Online Calibration Outputs:

Entity Id: **0x00B1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | CalibrationType | UINT8 | N/A |  |
| 2 | MagBiasX | DOUBLE64 | mG |  |
| 3 | MagBiasY | DOUBLE64 | mG |  |
| 4 | MagBiasZ | DOUBLE64 | mG |  |
| 5 | MagSfX | DOUBLE64 | N/A |  |
| 6 | MagSfY | DOUBLE64 | N/A |  |
| 7 | MagSfZ | DOUBLE64 | N/A |  |

### Magnetometer Derived Heading

Entity Id: **0x0077**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | magnetometer\_heading\_ | FLOAT32 | deg |  |

### Corrected Mag Outputs:

Entity Id: **0x0025**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | CorrectedMagX | DOUBLE64 | mG |  |
| 2 | CorrectedMagY | DOUBLE64 | mG |  |
| 3 | CorrectedMagZ | DOUBLE64 | mG |  |

### Barometer Height

Entity Id: **0x00E0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | barometer\_filter\_height\_ | FLOAT32 | m |  |
| 2 | barometer\_filter\_height\_standard\_deviation\_ | FLOAT32 | m |  |

### Odometer Scale Factor:

Entity Id: **0x0030**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | OdomF | DOUBLE64 | N/A |  |
| 2 | OdomFStdv | DOUBLE64 | N/A |  |

### Corrected Odometer Speed:

Entity Id: **0x00C9**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | CorrectedOdSpeed | DOUBLE64 | m/sec |  |

### Corrected odometer Speed STD:

Entity Id: **0x008A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | CorrectedOdSpeedStdv | DOUBLE64 | m/sec |  |

### Roll Misalignment Outputs:

Entity Id: **0x003D**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | RollMisalignment | DOUBLE64 | deg |  |

### Roll Misalignment STD Output:

Entity Id: **0x009A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | RollMisalignmentStdv | DOUBLE64 | deg |  |

### Pitch Misalignment Output:

Entity Id: **0x004E**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | PitchMisalignment | DOUBLE64 | deg |  |

### Pitch Misalignment STD Output:

Entity Id: **0x000A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | PitchMisalignmentStdv | DOUBLE64 | deg |  |

### Heading Misalignment STD Output:

Entity Id: **0x004F**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | misalignment\_standard\_deviation\_ | DOUBLE64 | deg |  |

### Resource Consumption

Entity Id: **0x00D5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | prediction\_time\_ | DOUBLE64 | milli-sec |  |
| 2 | update\_time\_ | DOUBLE64 | milli-sec |  |
| 3 | average\_epoch\_time\_ | DOUBLE64 | milli-sec |  |
| 4 | cpu\_percentage\_ | UINT8 | N/A |  |
| 5 | battery\_level\_ | UINT8 | N/A |  |
| 6 | **RESERVED** | DOUBLE64 | N/A |  |

### Flags - Events

Entity Id: **0x0004**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | flags\_ | UINT8 | N/A | **Bit[2:0]**: Alignment Info (3 bits)  **Bit[3]**: User Event Flag (1 bit)  **Bit[4]**: ~~Call On Flag (1 bit)~~  **Bit[5]**: ~~Proximity Sensor Reading (1 bit)~~  **Bit[7:6]**:~~Headset Reading (2 bits)~~ |

\*“Alignment info” indicates the Child Filters

### Flags – External Position

Entity Id: **0x00C0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | flags\_ | UINT8 | N/A | **Bit[0]**:  WiFi Valid Flag (1 bit)  **Bit[1]**:  WiFi In Use Flag (1 bit)  **Bit[2]**:  User Position Valid Flag (1 bit)  **Bit[3]**:  User Position In Use Flag (1 bit)  **Bit[4]**:  External Wireless Valid Flag (1 bit)  **Bit[5]**:  External Wireless In Use Flag (1 bit)  **Bit[6]**:  Map Position Valid Flag (1 bit)  **Bit[7]**:  Map Position In Use Flag (1 bit) |

### Flags – Vertical and Secondary Filter Info

Entity Id: **0x0040**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | flags\_ | UINT8 | N/A | **Bit[1:0]**:  Vertical Orientation (2 bits)  **Bit[5:2]**:  Secondary Info (4 bits)  **Bit[7:6]**:  Reserved (2 bits) |

### GNSS Raw Data\*

Entity Id: **0x0063**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_week\_ | UINT32 | N/A | Week |
| 2 | gnss\_tow\_ | DOUBLE64 | sec | Time of week |
| 3 | satellite\_num\_ | UINT32 | N/A | Number of satellites |
| 4 | gnss\_prn\_ | UINT32 | N/A | PRN |
| 5 | satellite\_system\_ | UINT32 | N/A | Satellite System:  0x00: GPS  0x01: GLONASS  0x02:COMPASS  0x03: GALILEO |
| 6 | signal\_type\_ | UINT32 | N/A | 0x00: L1/CA  0x01: L2  0x02: L5 |
| 7 | phase\_lock\_ | INT32 | N/A | 0x00: Lose Lock  0x01: Lock |
| 8 | code\_lock\_ | INT32 | N/A | 0x00: Lose Lock  0x01: Lock |
| 9 | lose\_lock\_indicator | UINT32 | N/A | Lose Lock Indicator |
| 10 | cn0\_ | DOUBLE64 | dB-Hz | Carrier to Noise density |
| 11 | lock\_time\_ | DOUBLE64 | sec | Time since the latest lock |
| 12 | pseudorange\_ | DOUBLE64 | m |  |
| 13 | pseudorange \_std\_ | DOUBLE64 | m |  |
| 14 | doppler\_ | DOUBLE64 | Hz |  |
| 15 | doppler\_std\_ | DOUBLE64 | Hz |  |
| 16 | phase\_ | DOUBLE64 | Cycles |  |

\* Packets are repeated <Num\_of\_Sat> times

### GPS Ephemeris Data\*

Entity Id: **0x0059**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_prn\_ | UINT32 | N/A | PRN |
| 2 | gnss\_week\_ | UINT32 | N/A | Week number |
| 3 | gnss\_tow\_ | DOUBLE64 | sec | Time of week |
| 4 | code\_ | UINT32 | N/A | Code on L2  0x00: Reserved  0x01: P code on  0x02: C/A code on |
| 5 | ura\_ | DOUBLE64 | m | Satellite accuracy |
| 6 | health\_ | UINT32 | N/A | Satellite health  0x00: all navigation date are ok  0x01: some navigation date are bad |
| 7 | iodc\_ | UINT32 | N/A |  |
| 8 | flag\_ | UINT32 | N/A | Data flag on L2P |
| 9 | tgd\_ | DOUBLE64 | sec |  |
| 10 | toc\_ | DOUBLE64 | sec |  |
| 11 | af2\_ | DOUBLE64 | sec/sec/sec |  |
| 12 | af1\_ | DOUBLE64 | sec/sec |  |
| 13 | af0\_ | DOUBLE64 | sec |  |
| 14 | iode2\_ | UINT32 | N/A |  |
| 15 | crs\_ | DOUBLE64 | m |  |
| 16 | deltan\_ | DOUBLE64 | semi-circles/sec |  |
| 17 | m0\_ | DOUBLE64 | semi-circles |  |
| 18 | cuc\_ | DOUBLE64 | rad |  |
| 19 | ecc\_ | DOUBLE64 | N/A |  |
| 20 | cus\_ | DOUBLE64 | rad |  |
| 21 | sqrta\_ | DOUBLE64 | √m |  |
| 22 | toe\_ | DOUBLE64 | sec |  |
| 23 | cic\_ | DOUBLE64 | rad |  |
| 24 | i0\_ | DOUBLE64 | semi-circles |  |
| 25 | crc\_ | DOUBLE64 | m |  |
| 26 | omega\_ | DOUBLE64 | semi-circles |  |
| 27 | omegadot\_ | DOUBLE64 | semi-circles/sec |  |
| 28 | iode3\_ | UINT32 | N/A |  |
| 29 | idot\_ | DOUBLE64 | semi-circles/sec |  |

### GLONASS Ephemeris Data\*

Entity Id: **0x00E8**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | gnss\_prn\_ | UINT32 | N/A |  |
| 2 | frequency\_ | INT32 | N/A | Frequency offset |
| 3 | toe\_ | DOUBLE64 | sec |  |
| 4 | toff\_ | DOUBLE64 | sec |  |
| 5 | iode\_ | UINT32 | N/A |  |
| 6 | health\_ | UINT32 | N/A | Satellite health  0x00: all navigation date are ok  0x01: some navigation date are bad |
| 7 | pos\_x\_ | DOUBLE64 | m |  |
| 8 | pos\_y\_ | DOUBLE64 | m |  |
| 9 | pos\_z\_ | DOUBLE64 | m |  |
| 10 | vel\_x\_ | DOUBLE64 | m/sec |  |
| 11 | vel\_y\_ | DOUBLE64 | m/sec |  |
| 12 | vel\_z\_ | DOUBLE64 | m/sec |  |
| 13 | acc\_x\_ | DOUBLE64 | m/sec/sec |  |
| 14 | acc\_y\_ | DOUBLE64 | m/sec/sec |  |
| 15 | acc\_z\_ | DOUBLE64 | m/sec/sec |  |
| 16 | taun\_ | DOUBLE64 | sec |  |
| 17 | gamman\_ | DOUBLE64 | sec/sec |  |
| 18 | tof\_ | DOUBLE64 | sec |  |
| 19 | age\_ | INT32 | N/A |  |

### Ionospheric Parameters\*

Entity Id: **0x00BF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | GPSTimeTag | DOUBLE64 | sec |  |
| 2 | A1 | DOUBLE64 |  |  |
| 3 | A2 | DOUBLE64 |  |  |
| 4 | A3 | DOUBLE64 |  |  |
| 5 | A4 | DOUBLE64 |  |  |
| 6 | B1 | DOUBLE64 |  |  |
| 7 | B2 | DOUBLE64 |  |  |
| 8 | B3 | DOUBLE64 |  |  |
| 9 | B4 | DOUBLE64 |  |  |

\* Packets are repeated <Num\_of\_Sat> times

### Satellite Parameters\*

Entity Id: **0x0052**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | Num\_of\_Sat | DOUBLE64 | N/A |  |
| 2 | GPSweek\* | UINT16 | N/A |  |
| 3 | GPSTimeTag\* | DOUBLE64 | sec |  |
| 4 | Num\_of\_Sat\* | UINT16 | N/A |  |
| 5 | SatelliteID\* | UINT8 | N/A |  |
| 6 | Satellite\_X \* | DOUBLE64 | m |  |
| 7 | Satellite\_Y\* | DOUBLE64 | m |  |
| 8 | Satellite\_Z\* | DOUBLE64 | m |  |
| 9 | Satellite\_Vx\* | DOUBLE64 | m/sec |  |
| 10 | Satellite\_Vy\* | DOUBLE64 | m/sec |  |
| 11 | Satellite\_Vz\* | DOUBLE64 | m/sec |  |
| 12 | Satellite\_Elev\* | DOUBLE64 | deg |  |
| 13 | Satellite\_Azi\* | DOUBLE64 | Deg |  |
| 14 | Sat\_Ion\_Corr\* | DOUBLE64 | Sec |  |

\* Packets are repeated <Num\_of\_Sat> times

### Multiple Antenna Data\*

Entity Id: **0x0075**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | heading\_ | FLOAT32 | deg |  |
| 2 | heading\_standard\_deviation\_ | FLOAT32 | deg |  |
| 3 | roll\_ | FLOAT32 | deg |  |
| 4 | roll\_standard\_deviation\_ | FLOAT32 | deg |  |
| 5 | pitch\_ | FLOAT32 | deg |  |
| 6 | pitch\_standard\_deviation\_ | FLOAT32 | deg |  |

### Temperature Sensor Data\*

Entity Id: **0x0071**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | GPSTimeTag | DOUBLE64 | Seconds |  |
| 2 | Temperature | FLOAT32 | K |  |

### Heading Data\*

Entity Id: **0x009B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | GPSTimeTag |  | Seconds |  |
| 2 | Heading | DOUBLE64 | Degrees |  |
| 3 | HeadingErr\_Std | FLOAT32 | Degrees |  |

### Map Matching Data\*

Entity Id: **0x00C7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | GPSTimeTag | DOUBLE64 | Seconds |  |
| 2 | Map\_ Derived\_Lat | DOUBLE64 | Degrees |  |
| 3 | Map\_ Derived\_Long | DOUBLE64 | Degrees |  |
| 4 | Map\_ Derived\_Head | DOUBLE64 | Degrees |  |

### Raw Map database Option1\*

Entity Id: **0x00C2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | num\_of\_map\_db\_entries |  |  |  |
| 2 | Seg\_Start\_Lat\* | DOUBLE64 | Degrees |  |
| 3 | Seg\_Start\_Long\* | DOUBLE64 | Degrees |  |
| 4 | Seg\_End\_Lat\* | DOUBLE64 | Degrees |  |
| 5 | Seg\_End\_Long\* | DOUBLE64 | Degrees |  |

### Raw Map database Option2\*

Entity Id: **0x00A0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | num\_of\_map\_db\_entries | DOUBLE64 | Degrees |  |
| 2 | Init\_Lat | DOUBLE64 | Degrees |  |
| 3 | Init\_Long | DOUBLE64 | Metres |  |
| 4 | Init\_Alt | FLOAT32 | m |  |
| 5 | Seg\_Start\_X\* | FLOAT32 | m |  |
| 6 | Seg\_Start\_Y\* | FLOAT32 | m |  |
| 7 | Seg\_End\_X\* | FLOAT32 | m |  |
| 8 | Seg\_End\_Y\* | FLOAT32 | m |  |
| 9 | Seg\_Slope\* | DOUBLE64 | N/A |  |

\* Packets are repeated < num\_of\_map\_db\_entries > times

### WiFi Beacon Message\*

Entity Id: **0x0018**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | GPSTimeTag | FLOAT32 | Sec |  |
| 2 | MAC | UINT8[6] | N/A |  |
| 3 | RSSI | FLOAT32 | dBm |  |
| 4 | Timestamp | ULONG | Ms |  |
| 5 | HostTimestamp | ULONG | 100ns |  |
| 6 | NetworkType | UNIT8 | N/A |  |
| 7 | SSID | BYTE[80] | N/A |  |
| 8 | Beacon Period | DOUBLE64 | Ms |  |
| 9 | Channel Frequency | ULONG | KHZ |  |
| 10 | Channel | INT32 | N/A |  |
| 11 | Lat | DOUBLE64 | Degrees |  |
| 12 | Long | DOUBLE64 | Degrees |  |
| 13 | Alt | FLOAT32 | Degrees |  |

### Height Data\*

Entity Id: **0x0081**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | height\_ | FLOAT32 | m |  |
| 2 | height\_standard\_deviation\_ | FLOAT32 | m |  |

### Heading Data\*

Entity Id: **0x0074**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | Heading | DOUBLE64 | Degrees |  |
| 2 | HeadingErr\_Std | FLOAT32 | Degrees |  |

### Other Wireless Position/Heading Data\*

Entity Id: **0x00CB**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Data Type** | **Unit** | **Description** |
| 1 | GPSTimeTag | DOUBLE64 | sec |  |
| 2 | Latitude | DOUBLE64 | deg |  |
| 3 | Longitude | DOUBLE64 | deg |  |
| 4 | Heading | DOUBLE64 | deg |  |
| 5 | Position std | FLOAT32 | m |  |
| 6 | Heading std | FLOAT32 | deg |  |

## Output Entities For Each Version

### 2.0.0-b

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Entity** | **Entity Id** | **Entity Type** |
| 1 | Time | 0x00ea | Standard |
| 2 | Position | 0x0016 | Standard |
| 3 | Position Standard Deviation | 0x00bd | Standard |
| 4 | Velocity | 0x00a2 | Standard |
| 5 | Velocity Standard Deviation | 0x005f | Standard |
| 6 | Attitude | 0x00e9 | Standard |
| 7 | Attitude Standard Deviation | 0x0060 | Standard |
| 8 | Accelerometer Bias | 0x006f | Standard |
| 9 | Gyroscope Bias | 0x002e | Standard |
| 10 | Heading Misalignment | 0x004d | TPN-Free |
| 11 | Platform Heading | 0x00b3 | TPN-Free |
| 12 | Number of Steps | 0x0099 | TPN-Free |
| 13 | Floor Number | 0x00dd | TPN-Free |
| 14 | Flags – Magnetometer and Barometer | 0x0091 | Standard |
| 15 | Mode of Transit | 0x00c8 | TPN-Free |
| 16 | Flags – GNSS, Speed and Static Status | 0x002b | Standard |
| 17 | Flags – Navigation Phase | 0x00e7 | Standard |
| 18 | Raw IMU Data\* | 0x00a1 | Standard / Debug |
| 19 | GNSS PVT Data\* | 0x001c | Standard / Debug |
| 20 | Barometer Data\* | 0x0083 | Standard / Debug |
| 21 | Magnetometer Data\* | 0x00b7 | Standard / Debug |
| 22 | Multiple Antenna Data\* | 0x0075 | Standard / Debug |
| 23 | Speed Data\* | 0x00a7 | Standard / Debug |
| 24 | Wi-Fi 2D Position Data\* | 0x0051 | TPN-Free / Debug |
| 25 | Operator 2D Position Data\* | 0x0042 | TPN-Free / Debug |
| 26 | Floor Information Data\* | 0x004a | TPN-Free / Debug |
| 27 | Call Information Data\* | 0x00ae | TPN-Free / Debug |
| 28 | Changed User Mode Event Data\* | 0x001d | TPN-Free / Debug |

### 2.1.0-rc/3.0.0

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Entity** | **Entity Id** | **Entity Type** |
| 1 | Time | 0x00ea | Standard |
| 2 | Position | 0x0016 | Standard |
| 3 | Position Standard Deviation | 0x00bd | Standard |
| 4 | Velocity | 0x00a2 | Standard |
| 5 | Velocity Standard Deviation | 0x005f | Standard |
| 6 | Attitude | 0x00e9 | Standard |
| 7 | Attitude Standard Deviation | 0x0060 | Standard |
| 8 | Accelerometer Bias | 0x006f | Standard |
| 9 | Gyroscope Bias | 0x002e | Standard |
| 10 | Heading Misalignment | 0x004d | TPN-Free |
| 11 | Platform Heading | 0x00b3 | TPN-Free |
| 12 | Number of Steps | 0x0099 | TPN-Free |
| 13 | Floor Number | 0x00dd | TPN-Free |
| 14 | Flags – Magnetometer and Barometer | 0x0091 | Standard |
| 15 | Mode of Transit | 0x00c8 | TPN-Free |
| 16 | Flags – GNSS, Speed and Static Status | 0x002b | Standard |
| 17 | Flags – Navigation Phase | 0x00e7 | Standard |
| 18 | Raw IMU Data\* | 0x00a1 | Standard / Debug |
| 19 | GNSS PVT Data\* | 0x001c | Standard / Debug |
| 20 | Barometer Data\* | 0x0083 | Standard / Debug |
|  | ~~Magnetometer Data\*~~ | ~~0x00b7~~ | ~~Standard / Debug~~ [Deprecated] |
| 21 | Magnetometer Data\* | 0x0062 | Standard / Debug |
|  | ~~Multiple Antenna Data\*~~ | ~~0x0075~~ | ~~Standard / Debug~~ [Removed in this release] |
| 22 | Speed Data\* | 0x00a7 | Standard / Debug |
| 23 | Wi-Fi 2D Position Data\* | 0x0051 | TPN-Free / Debug |
| 24 | Operator 2D Position Data\* | 0x0042 | TPN-Free / Debug |
| 25 | Floor Information Data\* | 0x004a | TPN-Free / Debug |
| 26 | Call Information Data\* | 0x00ae | TPN-Free / Debug |
| 27 | Changed User Mode Event Data\* | 0x001d | TPN-Free / Debug |

### 4.0.0-x

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Entity** | **Entity Id** | **Entity Type** |
| 1 | Time | 0x00ea | Standard |
| 2 | Position | 0x0016 | Standard |
| 3 | Position Standard Deviation | 0x00bd | Standard |
| 4 | Velocity | 0x00a2 | Standard |
| 5 | Velocity Standard Deviation | 0x005f | Standard |
| 6 | Attitude | 0x00e9 | Standard |
| 7 | Attitude Standard Deviation | 0x0060 | Standard |
| 8 | Accelerometer Bias | 0x006f | Standard |
| 9 | Gyroscope Bias | 0x002e | Standard |
| 10 | Heading Misalignment | 0x004d | TPN-Free |
| 11 | Platform Heading | 0x00b3 | TPN-Free |
| 12 | Number of Steps | 0x0099 | TPN-Free |
| 13 | Floor Number | 0x00dd | TPN-Free |
| 14 | Flags – Magnetometer and Barometer | 0x0091 | Standard |
| 15 | Mode of Transit | 0x00c8 | TPN-Free |
| 16 | Flags – GNSS, Speed and Static Status | 0x002b | Standard |
| 17 | Flags – Navigation Phase | 0x00e7 | Standard |
|  | ~~Use Case~~ | ~~0x00ee~~ | ~~TPN-Free~~ (Not output in this version) |
|  | ~~Stride Information~~ | ~~0x005e~~ | ~~TPN-Free~~ (Not output in this version) |
| 18 | Raw IMU Data\* | 0x00a1 | Standard / Debug |
| 19 | GNSS PVT Data\* | 0x001c | Standard / Debug |
| 20 | Barometer Data\* | 0x0083 | Standard / Debug |
| 21 | Magnetometer Data\* | 0x0062 | Standard / Debug |
| 22 | Speed Data\* | 0x00a7 | Standard / Debug |
| 23 | Wi-Fi 2D Position Data\* | 0x0051 | TPN-Free / Debug |
| 24 | Operator 2D Position Data\* | 0x0042 | TPN-Free / Debug |
| 25 | Floor Information Data\* | 0x004a | TPN-Free / Debug |
| 26 | Call Information Data\* | 0x00ae | TPN-Free / Debug |
| 27 | Set Mode of Transit Event Data\* | 0x001d | TPN-Free / Debug [Name Changed from “Changed User Mode Event Data”] |
| 28 | Platform Heading Data\* | 0x00ba | TPN-Free / Debug |
| 29 | Quaternions Data\* | 0x007a | TPN-Free / Debug |
| 30 | Gyroscope Delta Biases Data\* | 0x0008 | TPN-Free / Debug |
| 31 | Set ZUPT Mode Event Data\* | 0x009f | TPN-Free / Debug |
| 32 | Set Misalignment Event Data\* | 0x00a4 | TPN-Free / Debug |
| 33 | Set Magnetometer Calibration Status Event Data\* | 0x00e4 | TPN-Free / Debug |
| 34 | Set Use Case Event Data\* | 0x0098 | TPN-Free / Debug |
| 35 | 6 DOF Quaternions | 0x007B | TPN-Free/Debug |
| 36 | In-run Magnetometer Calibration Information ^ | 0x000f | TPN-Free / Internal [Engine Mode = 0] |
| 37 | Internal Debugging Information(1) ^ | 0x008c | TPN-Free / Internal [Engine Mode = 0] |

### 4.1.0-x

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Entity** | **Entity Id** | **Entity Type** |
| 1 | Time | 0x00ea | Standard |
| 2 | Position | 0x0016 | Standard |
| 3 | Position Standard Deviation | 0x00bd | Standard |
| 4 | Velocity | 0x00a2 | Standard |
| 5 | Velocity Standard Deviation | 0x005f | Standard |
| 6 | Attitude | 0x00e9 | Standard |
| 7 | Attitude Standard Deviation | 0x0060 | Standard |
| 8 | Accelerometer Bias | 0x006f | Standard |
| 9 | Gyroscope Bias | 0x002e | Standard |
| 10 | Heading Misalignment | 0x004d | TPN-Free |
| 11 | Platform Heading | 0x00b3 | TPN-Free |
| 12 | Number of Steps | 0x0099 | TPN-Free |
| 13 | Floor Number | 0x00dd | TPN-Free |
| 14 | Converged Gyroscope Bias | 0x002f | TPN-Free [Added in 4.1.0-a.5] |
| 15 | Flags – Magnetometer and Barometer | 0x0091 | Standard |
| 16 | Mode of Transit | 0x00c8 | TPN-Free |
| 17 | Flags – GNSS, Speed and Static Status | 0x002b | Standard |
| 18 | Flags – Navigation Phase | 0x00e7 | Standard |
| 19 | Use Case | 0x00ee | TPN-Free |
| 20 | Stride Information | 0x005e | TPN-Free |
| 21 | Raw IMU Data\* | 0x00a1 | Standard / Debug |
| 22 | GNSS PVT Data\* | 0x001c | Standard / Debug |
| 23 | Barometer Data\* | 0x0083 | Standard / Debug |
| 24 | Magnetometer Data\* | 0x0062 | Standard / Debug |
| 25 | Speed Data\* | 0x00a7 | Standard / Debug |
|  | ~~Wi-Fi 2D Position Data\*~~ | ~~0x0051~~ | ~~TPN-Free / Debug~~ |
|  | ~~Operator 2D Position Data\*~~ | ~~0x0042~~ | ~~TPN-Free / Debug~~ |
| 26 | Operator 2D Position Data\* | 0x0043 | TPN-Free / Debug |
| 27 | Wi-Fi Data\* | 0x0055 | TPN-Free / Debug |
| 28 | Floor Information Data\* | 0x004a | TPN-Free / Debug |
| 29 | Call Information Data\* | 0x00ae | TPN-Free / Debug |
| 30 | Set Mode of Transit Event Data\* | 0x001d | TPN-Free / Debug [Name Changed from “Changed User Mode Event Data”] |
| 31 | Platform Heading Data\* | 0x00ba | TPN-Free / Debug |
| 32 | Quaternions Data\* | 0x007a | TPN-Free / Debug |
| 33 | Gyroscope Delta Biases Data\* | 0x0008 | TPN-Free / Debug |
| 34 | Process Misalignment Data\* | 0x00a6 | TPN-Free/Debug [Added in 4.1.0-a.5] |
| 35 | Set ZUPT Mode Event Data\* | 0x009f | TPN-Free / Debug |
| 36 | Set Misalignment Event Data\* | 0x00a4 | TPN-Free / Debug |
| 37 | Set Magnetometer Calibration Status Event Data\* | 0x00e4 | TPN-Free / Debug |
| 38 | Set Use Case Event Data\* | 0x0098 | TPN-Free / Debug |
| 39 | 6 DOF Quaternions | 0x007B | TPN-Free/Debug |
| 40 | Synchronization Event Data\* | 0x00ff | TPN-Free/Debug [Added in 4.1.0-a.5] |
| 41 | In-run Magnetometer Calibration Information ^ | 0x000f | TPN-Free / Internal [Engine Mode = 0] |
| 42 | Internal Debugging Information(1) ^ | 0x008c | TPN-Free / Internal [Engine Mode = 0] |

## Appendix A: Predefined Configurations

### Predefined GNSS Configurations

|  |  |
| --- | --- |
| **PREDEFINED\_STRING** | **Examples of GNSS Receivers** |
| “gnss\_high\_sensitivity” | Qualcomm-GNSS(GPS+GLONASS) (SIII), Default High Sensitivity |
| “gnss\_high\_sensitivity\_1” | SiRFstar (Nexus) |
| “gnss\_high\_sensitivity\_2” | Broadcom-GNSS(GPS+GLONASS) (Note1) |
| “gnss\_high\_sensitivity\_3” | u-blox (when used with low grade gyroscopes with bias instability ≥ 100 deg/hr) |
| “gnss\_high\_sensitivity\_4” | Low cost GNSS receivers that provide unusable GPS velocities |
| “gnss\_high\_sensitivity\_5” | u-blox (when used with high grade gyroscopes with bias instability < 100 deg/hr) |
| “gnss\_high\_sensitivity\_6” | BlackBerry Playbook GNSS Receiver |
| “gnss\_high\_sensitivity\_7” | Not Specified |
| “gnss\_high\_sensitivity\_8” | Samsung Note 3 |
| “gnss\_precision” | NovAtel, Hemisphere, Default Precision Receivers |
| “gnss\_precision\_1” | Trimble |

### Predefined Accelerometer Configurations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PREDEFINED\_STRING** | **Bias Instability (mGal)** | **VRW**  **(m/sec/√hr)** | **Correlation time (hr)** | **Examples of Triad accelerometers** |
| “accelerometer\_1” | 1700 | 2.4 | 2.0 | MPU3050(Nexus), MPU6050(SIII), MPU9150, Note(STMICRO) |
| “accelerometer\_2” | 100  Turning: 5000 | 0.09 | 4.0  Turning:  0.5 | ADIS16485,ADIS16385 |
| “accelerometer\_3” | 3700 | 3.4 | 2.0 | BlackBerry Playbook |
| “accelerometer\_4” | 1000 | 0.5 | 1.0 | Murata SCC1300-D04 |
| “accelerometer\_5” | 500 | 0.1 | 1.0 | **TMN/Airplane[[4]](#footnote-4)**  Murata SCC1300-D04 |

### Predefined Gyroscope Configurations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PREDEFINED\_STRING** | **Bias Instability (deg/hr)** | **ARW (deg/√hr)** | **Correlation time (hr)** | **Examples of triad gyroscopes** |
| “gyroscope\_1” | 100 | 1.8 | 1.0 | MPU6050(SIII), MPU9150 |
| “gyroscope\_2” | 150 | 1.8 | 1.0 | MPU3050(Nexus), STMICRO (Note) |
| “gyroscope\_3” | 4 | 0.15 | 4.0 | ADIS16485 |
| “gyroscope\_4” | 10 | 1.8 | 1.0 | Embedded library |
| “gyroscope\_5” | 20  Turning:  35 | 1.0 | 1.0 | Murata SCR1100-D04 |
| “gyroscope\_6” | 1 | 0.15 | 4.0 | NG Litef uFORS-6U |
| “gyroscope\_7” | 21 | 1.9 | 1.0 | ADIS16385(X,Y GYROS) |
| “gyroscope\_8” | 6 | 0.75 | 1.0 | ADIS16385(Z GYRO) |
| “gyroscope\_9” | 15 | 2.0 | 1.0 | **TMN/Airplane**  Murata SCR1100-D04 |
| “gyroscope\_10” | 350 | 5 | 1.0 | Lumia820 Gyros for walk |

### Predefined Motion Constraints Configurations

|  |  |  |  |
| --- | --- | --- | --- |
| **PREDEFINED\_STRING** | **Dynamic Velocity Constraint (m/sec)** | **Static Velocity Constraint (m/sec)** | **Examples of Triad Accelerometers** |
| “motion\_constraints\_1” | 0.1 | 0.1 | MPU6050(SIII),MPU9150 |
| “motion\_constraints\_2” | 0.3 | 0.1 | Embedded library |
| “motion\_constraints\_3” | 0.4 | 0.5 | Murata SCC1300-D04 |
| “motion\_constraints\_4” | 0.15 | 0.5 | ADIS16485, ADIS16385 |
| “motion\_constraints\_5” | 50.0 | 0.1 | **TMN/Airplane**  MPU3050(Nexus), STMicro (Note),MPU6050(SIII), MPU9150 |
| “motion\_constraints\_6” | 35.0 | 0.5 | **TMN/Airplane**  Murata SCC1300-D04, ADIS16385, ADIS16485 |

## Change Log

Table Change Log

|  |  |
| --- | --- |
| **Version** | **Changes** |
| v4.5.0-alpha | 1. Adding functions to create and delete a navigation session handle. This navigation session handle needs to be passed to all the functions. The use of the navigation session handle would allow the developer to use multiple navigation sessions in the same process that loads the TPP library.  2. Adding the multi-device API functions, structures, return values.  v4.5.0-alpha.11  1. Adding the initialization flag DECLINATION\_ANGLE  2. Adding the function **tpp\_multi\_device\_process\_secondary\_gnss\_pvt()**  4. Adding temporary use-cases for testing  5. Adding the “running” mode of transit  v4.5.0-alpha.16  1. Adding the function **tpp\_process\_venue\_map(),** the function return types, the corresponding structure **TppVenueMapMessageStructure**, and the corresponding entity.  v4.5.0-alpha.20  1. Updating the type of **solution\_reliable\_** field in TppSolutionStruct to INT8 instead of UINT8. The reason is to use the value (-1) to indicate a new phase which is pre-alignment. In this phase, the solution shall be discarded entirely in plotting or as a reference.  2. In the entity ‘**Flags – Navigation Phase**’, the reserved 4 bits are removed, and the whole byte is allocated for the navigation phase. The type of the byte also changed from INT8 to UINT8 to reflect the change related to the point above,  3. Updating the name of the flag **solution\_reliable\_** in TppSolutionStruct to **navigation\_phase\_** to be consistent in what both values represent.  v4.5.0-alpha.21  1. Adding values 0x07 and 0x08 for ‘Conveyer Walking’ and ‘Conveyer Standing’  v4.5.0-alpha.30  1. Updating the structure definition for **TppApiVersionStruct**.  2. Renaming the name of the function **tpp\_process\_gyroscope\_delta\_biases()** to **tpp\_process\_imu\_biases()**. The corresponding structures and enumerations’ names are changed as well.  3. Updating the structure definition for **TppGyroscopeDeltaBiasesMessageStructure** accordingly.  v4.5.0-alpha.  1. Adding the function **tpp\_add\_anchor\_point (),** the function return types, the corresponding structure **TppAnchorPoint**, and the corresponding entity.  2. Adding the three fields related to the “step length scale” to **TppVenueMapMessageStruct** and the corresponding entity.  3. Adding the function **tpp\_set\_orientation\_based\_on\_pitch()**, the corresponding enumeration **TppOrientationBasedOnPitch**, return types and entity “**Orientation Based On Pitch Data**”.  4. Adding the entity **Orientation Based On Pitch**  5. Adding the initialization flag **PROCESSING\_MODE**  v4.5.0-alpha.60  1. Replacing tpp\_process\_wifi() with tpp\_process\_wireless() and updating the structures accordingly.  2. Adding the “cycling” mode of transit  3. Adding the entities “Distance Travelled” and “DSL Height” |
| v4.2.0-beta | 1. Beta release of v4.2.0 of the library  2. Updating the **solution\_reliable\_** flag definition in **TppSolutionStruct**  3. Adding the function **tpp\_set\_device\_heading()** |
| v4.1.0-alpha | 1. Alpha release of v4.1.0 of the library  2. For “**MISALIGNMENT**” initialization flag**, flag\_value\_2\_** now takes a value when **flag\_value\_1\_** is “**automatic**”. The value for **flag\_value\_2\_** can be “**misalignment\_normal**” or “**misalignment\_snap**”  3. **tpp\_process\_2d\_position()** is no more available. To process operator 2d position updates, the user can use **tpp\_process\_operator\_2d\_position().**  4. To process Wi-Fi data, **tpp\_process\_wifi()** is added.  5. **tpp\_process\_quaternions()** is renamed to **tpp\_process\_9dof\_quaternions()**  6. The name of the structure **TppMisalignmentStruct** is changed to **TppSetMisalignmentStruct**  7. **tpp\_process\_misalignment()**, and its corresponding structure, entity, and return value is added. |
| v4.0.0-alpha | 1. Alpha release of v4.0.0 of the library  2. Added Initialization Flags:  a. “**OUTPUT\_POSITION\_ERROR\_WEIGHTING**”  b. “**OUTPUT\_VELOCITY\_ERROR\_WEIGHTING**”  c. “**OUTPUT\_ATTITUDE\_ERROR\_WEIGHTING**”  d. “**ACCELEROMETER\_BIASES**”  3. Added Functions. The corresponding structures/enumerators and data/debug entities are added as well:  a. **tpp\_process\_platform\_heading**  b. **tpp\_process\_quaternions**  c. **tpp\_process\_6dof\_quaternions**  d. **tpp\_process\_gyroscope\_delta\_biases**  e. **tpp\_set\_zupt\_mode**  f. **tpp\_set\_misalignment**  g. **tpp\_set\_magnetometer\_calibration**  h. **tpp\_set\_use\_case**  4. Added output entities:  a. **Stride Information** – entity Id: **0x005E**  b. **Use Case** – entity Id: **0x00EE**  5. Additional values are used for “**Mode of Transit**” entity – entity Id: **0x00C8** |
| v3.0.0 | 1. Official release of the ready for production version of the library  2. “**START\_OPTION**” initialization flag is added to the list of initialization flags  3. The name of the function **tpp\_change\_mode\_of\_transit**() changed to **tpp\_set\_mode\_of\_transit()**  4. The raw magnetometer entity structure is revised and the entity ID changed.  5. The behavior of **tpp\_process\_2d\_position()** is changed. If the **source\_** field is equal to 0x00 (i.e. Operator position update), the standard deviation will not take effect. |
| v2.0.0-beta | 1. Official release of the beta version of the library |

## Revision History

Table Document Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision** | **Changes** |
| 2012-November-10 | 1 | Initial release. |
| 2012-December-06 | 2  (1.5.1) | 1. Adding a couple of functions to process (1) the 2D position (2) height from a source other than GPS or barometer (3) multiple antenna data (4) floor information  2. Correcting the measurement units  3. Adding a table for the list of definitions  4. Adding pseudo-code for checksum computation |
| 2013-February-28 | 3  (1.5.2) | 1. Adding an API function to get the major, minor and patch versions for the API.  2. Removing the default values that specific options will take if the application developer used a flag name without a flag value.  3. Revising the code samples provided.  4. Changing the order of the first three sections.  5. Changing the patch number due to fixing bugs, and adding a new function.  6. Removing the offset column in the output array structure table.  7. Updating the footer of the document.  8. Adding a specific title on the cover page to specify whether the API is for TPN/Free, TPN/Tethered, TMN or TVN.  9. TMN initialization in 7.1. is revised.  10. Tightly coupled entities added by Tao. |
| 2013-April-23  2013-May-13  2013-May-28 | 4  (2.0.0 Beta) | 1. Adding a function to initialize TMN/Aerial.  2. Changing the name of the function used to initialize TMN system **tpp\_initialize\_tmn()** to **tpp\_intialize\_tmn\_land()**  3. Adding **tpp\_process\_call\_information()** function and the corresponding input structure  4. Removing the **timetag\_** member/field from four input structures   * Tpp2dPositionMessageStruct * TppHeightMessageStruct * TppFloorInformationMessageStruct * TppCallInformationMessageStruct   5. Error codes enumeration values are updated to match more logical and consistent values.  6. Some entities are revised such as   * Wi-Fi Data * 2D Position Data * Height Data * Floor Information Data   6. Tightly coupled functions **tpp\_process\_gnss\_observations(),** **tpp\_process\_gps\_ephemeris()**, **tpp\_process\_glonass\_ephemeris()** and the corresponding input structures along with the error codes are defined.  7. **TppPacketStruct** replaces **TppOutputStruct** (just a name replacement)  8. **TppSolutionStruct** is added for customers  9. **sec** is used instead of **s** in some areas.  10. **Height** definition is refined. **Developer** and **operator** definitions are added.  11. **tpp\_advance\_navigation\_step**() is restructured to accommodate for another parameter of type **TppSolutionStruct**  12. **tpp\_change\_mode\_of\_transit**() is added along with corresponding enum and error code to return.  13. For every initialization flag, we have now up to 9 flag values that can be set instead of 4.  14. **GNSS** initialization flag: adding **scalefactor**, and receiver type  15. **WIFI** initialization flag added: adding **scalefactor**  16. **IMU\_CONFIGURATION**, **ACCELEROMETER\_CONFIGURATION**, **GYROSCOPE\_CONFIGURATION** and **DEBUG\_DATA** flags are added  17. **MAGNETOMETER**,**BAROMETER**, and **SPEED** flags: adding **scalefactor**  18. **HEADING**: on/off flag removed  19. **message\_id\_**, and **message\_sub\_id\_** members are removed from the input structures.  20. GNSS/PVT input structure: members **dop\_data\_available\_** added & **pdop\_,gdop\_,gnss\_velocity\_timetag** are removed  21. Restructuring the barometer and magnetometer structures  22. Barometer, Magnetometer, Raw IMU, GNSS entities are restructured  23. Stars (\*) are added to identify data/debug entities  24. Names of some entities were refined  25. Call Information Data entity is added  26. Error codes are added to make it easier for the developer to identify the problems  27. The word ‘developer’ is used instead of ‘user’  28. **sensor\_status\_** member is removed from RAW IMU structure  29. **tpp\_change\_mode\_of\_transit**() is added  30. Adding “**MOTION\_CONSTRAINTS**” flag  31. No default values for “**GYROSCOPE\_BIASES**”. Has to be set by the user, or an error will be returned. Adding corresponding error code.  32. Wired and wireless earphones events are combined in one flag instead of two.  33. Changes to the types of some variables/members in the structures and the entities.  34. Heading is used instead of azimuth in all locations  35. All flag values use non-capital letters  36. **platform\_heading\_** is added to **TppSolutionStructure**  37. **HEADING** flag is now **PLATFORM\_HEADING**  38. Changing the name of the initialize TPN/Free function to **tpp\_initialize\_tpn\_free()** instead of **tpp\_initialize\_tpn\_free\_motion()**  39. The configurations table is refined  40. “**GNSS**” flag values are refined  41. Correcting the default GNSS flag values for TPN  42. “Examples of GNSS Receivers” is used instead of “Included Receivers” |
| 2013-June-24 | 5  (Customized Library) | 1. Customized API functions are added |
| 2013-July-19 | 6 | 1. Adding an API function, initialization flag, and return enumerations to run the backward smoothing functionality. |
| 2013-August-27 | 7 | 1. Changing the word ‘scaling’ to ‘weighting’. Other wording is changed accordingly. |
| 2013-September-30 | 8 | Refer to the change log for v4.0.0-alpha |
| 2013-November-14 | 9 | Refer to the change log for v4.1.0-alpha |
| 2014-January-22 | 10 | Changes related to the TMN and TVN Libraries  1. **START\_OPTION** initialization flag is now supported for TMN  2. **flag\_value\_1\_** for **GNSS** can be set to “on” or “off” where previously it was reserved |
| 2014-July-25 | 11 | Adding the initialization flag “**DECLINATION\_ANGLE**” |

1. “F.F” specifies that the value that should be inserted in this field is a string representing a floating number. There is no limit to the number of digits, so as an example the developer can insert “123.456”. [↑](#footnote-ref-1)
2. “N” specifies that the value that should be inserted in this field is a string representing an integer. If a floating number is used in the corresponding field, the number will be implicitly casted to an integer. [↑](#footnote-ref-2)
3. Entities with a star (\*) next to their names correspond to entities that hold the input data and are mainly used for Trusted Positioning Debugging. Those entities can be generated by initializing the “DEBUG\_DATA” flag accordingly. [↑](#footnote-ref-3)
4. Not applicable for the TPN/Free navigator. [↑](#footnote-ref-4)