



GENIVI GNSSService

Component Specification

Release 4.0.0
Status: Released
25.01.2016

Accepted for release by:

Approved by the GENIVI expert group Location Based Services (LBS) and the GENIVI system architecture team (SAT).

Abstract:

This document describes the API 4.0.0 of the **GNSSService** Abstract Component.

Keywords:

GNSSService, GNSS, GPS, Positioning API.

SPDX-License-Identifier: CC-BY-SA-4.0

Copyright © 2012, BMW Car IT GmbH, Continental Automotive GmbH, PCA Peugeot Citroën, XS Embedded GmbH

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License

To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

Table of Contents

Change History	4
1. Introduction	5
2. Terminology	6
3. Requirements	7
1. Requirements Diagram	7
AGPS Support	9
Forward a Set of Sensor Values	9
Provides Data via IPC.....	9
Support of different Global Navigation Satellite Systems (GNSS) to calculate the current position.	9
Accelerator Sensor.....	9
Access to Sensor Services	10
Car Configuration Data	10
Data Latency for GNSS and DR Signals.....	10
Enhanced Position	11
Extended Acceleration Sensor.....	11
Extended GNSS Service	11
Extended Gyroscope Sensor Service.....	12
GNSS Service	12
PPS Signal	12
Inclination Sensor	13
Odometer Sensor	13
ReverseGear Sensor.....	13
Sensor Directory	13
Sensor Meta-Data	14
Sensor Signal Timestamp	14
Signal Measurement Units.....	14
Signal Values Type Compatibility	15
Simple Gyroscope Sensor Service.....	15
Slip Angle Sensor	15
SteeringAngle Sensor	15
Vehicle State Sensor	16
VehicleSpeed Sensor	16
Wheel Tick/Speed Sensor Service.....	16
4. Architecture	17
1. GNSSService	17
2. GNSSService Diagram	17
3. Traceability Diagram	18
4. Context Diagram.....	19

Change History

Version	Date	Author	Change
0.1	27.08.2013	M. Residori	Document Created
0.2	18.11.2013	M. Residori	Document generated from the GENIVI Enterprise Architect model
0.3	27.03.2014	M. Residori	Added copyright notes
3.0.0-alpha	24.04.2014	M. Residori	Changed license version from 3.0 to 4.0
3.0.0-alpha	10.12.2014	M. Residori	Updated API description
3.0.0	20.01.2015	M. Residori	Changed document status to “Released” (after System Architecture Team approval)
3.0.0	01.04.2015	H. Schmidt	Updated approval note on title page
4.0.0-alpha	16.12.2015	M. Residori	Updated API description
4.0.0	25.01.2016	M. Residori	Release 4.0.0

1. Introduction

This document describes the API of the GNSSService component.

The GNSSService is a component that abstracts the access to GNSS devices (e.g. GPS receivers).

It hides hardware and software dependencies on specific GNSS devices and their drivers.

In systems that implements the EnhancedPositionService component, the GNSSService is typically provided as a C library that is dynamically linked by the EnhancedPositionService.

2. Terminology

<i>Term</i>	<i>Description</i>
GNSS	Global Navigation Satellite System

3. Requirements

1. Requirements Diagram

This diagram shows an overview of all requirements in the area of positioning.

The requirements are organized in four groups:

1. SW-POS: general requirements
2. SW-GNSS: requirements related to the GNSS receiver
3. SW-SNS: requirements related to the vehicle sensors
4. SW-ENP: requirements related to enhanced positioning

req Requirements

SW-POS

Sensor Meta-Data <i>tags</i> Originator = Thomas Himbacher (BMW) Priority = P1 Rationale =	Sensor Signal Timestamp <i>tags</i> Originator = Thomas Bader (BMW) Priority = P1 Rationale =	Signal Measurement Units <i>tags</i> Originator = Thomas Bader (BMW) Priority = P1 Rationale =	Car Configuration Data <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =
Signal Values Type Compatibility <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =	Access to Sensor Services <i>tags</i> Originator = Thomas Himbacher (BMW) Priority = P2 Rationale =	Sensor Directory <i>tags</i> Originator = Thomas Himbacher (BMW) Priority = P2 Rationale =	Support of different Global Navigation Satellite Systems (GNSS) to calculate the current position. <i>tags</i> Originator = Philippe Colliot (PSA) Priority = P2 Rationale = <memo>

SW-POS-GNSS

GNSS Service <i>tags</i> Originator = Thomas Himbacher (BMW) Priority = P1 Rationale =	Extended GNSS Service <i>tags</i> Originator = Thomas Himbacher (BMW) Priority = P2 Rationale =	PPS Signal <i>tags</i> Originator = Carsten Iserl (BMW) Priority = P2 Rationale =	AGPS Support <i>tags</i> Originator = Thomas Bader (BMW) Priority = P3 Rationale =
---	--	--	---

SW-POS-SNS

VehicleSpeed Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =	Odometer Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =	Wheel Tick/Speed Sensor Service <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =	Simple Gyroscope Sensor Service <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =
Extended Gyroscope Sensor Service <i>tags</i> Originator = Thomas Bader (BMW) Priority = P3 Rationale =	ReverseGear Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =	Accelerator Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =	Extended Acceleration Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P3 Rationale =
Inclination Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P3 Rationale =	Slip Angle Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P3 Rationale =	Vehicle State Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P3 Rationale =	SteeringAngle Sensor <i>tags</i> Originator = Thomas Bader (BMW) Priority = P3 Rationale =

SW-POS-ENP

Enhanced Position <i>tags</i> Originator = Thomas Himbacher (BMW) Priority = P1 Rationale =	Provides Data via IPC <i>tags</i> Originator = Thomas Bader (BMW) Priority = P1 Rationale =	Forward a Set of Sensor Values <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =	Data Latency for GNSS and DR Signals <i>tags</i> Originator = Thomas Bader (BMW) Priority = P2 Rationale =
--	--	---	---

Figure: 1

AGPS Support		
«GFunctionalRequirement»	Priority: Medium	
Description: The software platform provides the possibility to inject AGPS "Assisted GPS" data to the GPS device.		
Rationale: This allows to speed up the time to get a valid (fixed) GPS position.		

Forward a Set of Sensor Values		
«GFunctionalRequirement»	Priority: Medium	
Description: The Enhanced Position contains in addition to the Position and Course values as well a set of sensor data. <ul style="list-style-type: none"> - yawRate in degrees per second - filter status - accuracy information in form of sigma values for every direction [m] and the covariance between latitude and longitude in m². - number of used, tracked and visible satellites. 		
Rational: Some clients (e.g. Map Matcher) needs the basic DR filtered position specific sensor values as additional input for the decision algorithm.		

Provides Data via IPC		
«GFunctionalRequirement»	Priority: Medium	
Description: The enhanced position is accessible for multiple clients on the platform at the same time. An IPC is used to deliver to the clients the Enhanced Position data fields.		
Rational: Several SW components in the system are clients for the result of the filtered position and need to access the data.		

Support of different Global Navigation Satellite Systems (GNSS) to calculate the current position.		
«GFunctionalRequirement»	Priority: Medium	
The interfaces are defined in such a way that client applications don't need to know the details of the GNSS in use (e.g. GPS, Galileo, GLONASS, Compass).		

Accelerator Sensor

«GFunctionalRequirement»	Priority: Medium	
Description: The software platform provides a sensor, which delivers the vehicle acceleration in the driving direction (x Axis, see reference system). The sensor value is delivered in m/s^2. Sensor value of temperature near the sensor is optional. Configuration data about placement and orientation of the sensor can be provided optionally.		
Rational: Used for optimizing the dead reckoning solution.		

Access to Sensor Services		
«GFunctionalRequirement»	Priority: Medium	
Description: The software platform delivers signals to multiple client applications concurrently by the Sensor Service.		
Rational: This allows for multiple Client Applications to share a single Sensor.		

Car Configuration Data		
«GFunctionalRequirement»	Priority: Medium	
Description: The software platform provides car configuration data, that contains general vehicle details (e.g. physical dimensions of car, distance of axis, driven axis, etc). Sensor related configuration data depends on the specific sensor requirements (e.g. position of sensor) and is included with the specific sensors. <ul style="list-style-type: none"> - Position of center of gravity - Position of front and rear axle - driven axles - seat count - vehicle mass - vehicle width - track width 		
Rational: DR module needs the detailed information for more accurate calculations.		

Data Latency for GNSS and DR Signals		
«GNonFunctionalRequirement»	Priority: Medium	
Description: The software platform provides the signals of the GNSS, Extended GNSS and enhanced position in less than 300 ms after acquisition.		
Rational: This guarantees that the tracked current position does not deviate much from the actual position.		

Enhanced Position		
«GFunctionalRequirement»	Priority: Medium	
Description: The software platform delivers the filtered (i.e. combined GNSS and vehicle sensor) position as the Enhanced Position, which is the result of the dead reckoning calculation. The Enhanced Position contains: <ul style="list-style-type: none"> - Position expressed as WGS 84 longitude and latitude (unit is tenth of microdegree (degree x 10⁻⁷)) - the Altitude 'above mean sea level' in meters (corrected by GeoID) - Heading in degrees relative to the true north - Climb - Speed in meters per seconds, positive in the forward direction Rational: Other SW-components on the same platform want to access the improved GNSS position, which is calculated by a dead reckoning algorithm.		

Extended Acceleration Sensor		
«GFunctionalRequirement»	Priority: Low	
Description: The software platform provides a sensor, which provides the acceleration on the additional axis y (left-side) and z (up). The position of the sensor in 3D space in relation to the reference point is given. The angles of the sensor can be specified in the car configuration data. The standard deviations for the sensors can be specified for each axis. Rational: Used for optimizing the dead reckoning solution.		

Extended GNSS Service		
«GFunctionalRequirement»	Priority: Medium	
Description: The software platform provides an extension to the GNSS Service with optional information. <p>Accuracy:</p> <ul style="list-style-type: none"> - fixStatus - hdop, pdop, vdop - numberOfSatellites - sigmaLatitude, sigmaLongitude, sigmaAltitude <p>Satellite Details:</p> <ul style="list-style-type: none"> - Information per satellite: azimuth, elevation, inUse, SatelliteId, signalNoiseRatio <p>Course Details:</p> <ul style="list-style-type: none"> - speed for 3-axis <p>Antenna:</p> <ul style="list-style-type: none"> - Antenna Position in 3D coordinates in relation to the reference point (see reference system). <p>Updated at least with 1Hz frequency additionally to the Signals provided by GNSS-Only Service.</p>		

The GNSS Service should provide the capability to switch between different GNSS-Devices (e.g. Galileo, GPS, etc)

Rational:

These data are used for improved positioning based on GNSS.

Extended Gyroscope Sensor Service

«GFunctionalRequirement»

Priority: Low

Description:

The software platform includes the sensor that delivers

- pitch rate
- roll rate

This sensor values extend the simple gyroscope sensor.

Sign of is defined by rule of right hand (thumb direction: left and front, see reference system).

Car configuration data need to provide position angles according to vehicle reference system.

Rational:

This Sensor Service is used in Dead Reckoning calculations of the vehicle position.

GNSS Service

«GFunctionalRequirement»

Priority: High

Description:

The software platform includes a service that provides the following GNSS Signals updated at least with 1Hz frequency:

Position:

- position expressed as WGS 84 altitude, longitude and latitude in tenth of microdegree (degree x 10^{-7})

Course:

- speed in meters per second
- climb
- heading relative to true north expressed in degrees

Timestamp and date as UTC.

Rational:

These data are contained in NMEA 0183 \$GPGGA and \$GPRMC messages and provide the minimum information required for GNSS-only vehicle positioning.

PPS Signal

«GFunctionalRequirement»

Priority: Medium

Description:

1) For accurate timing the 1 PPS (pulse per second) signal from the GPS receiver is provided within the positioning framework.

The PPS is a hardware signal which is a UTC synchronized pulse.

The duration between the pulses is 1s +/- 40ns and the duration of the pulse is configurable (e.g. it could be

100ms or 200ms).

The pulses occur exactly at the UTC full second timeslots.

2) One option is to provide this signal in the positioning framework as an interrupt service routine and the difference to the system time can be accessed by a getter. This provides a synchronization of the system time to UTC.

Rationale:

Used for synchronizing the timing of the ECU.

Inclination Sensor

«GFunctionalRequirement» Priority: Low

Description:

The software platform provides the inclination of the road in longitudinal direction, i.e. in the direction of movement [°]. Estimated gradient of the road in transverse direction [°]. In unstable driving situations this value might not be available.

Rational:

This Sensor is used for optimizations in Dead Reckoning calculations of the vehicle position.

Odometer Sensor

«GFunctionalRequirement» Priority: Medium

Description:

The software platform includes a Sensor that delivers the traveled distance.

Distance in [cm] with at least 5Hz as a running counter with overflow to support multiple clients.

Rational:

Odometer is sometimes the only speed related Signal available to the head unit.

ReverseGear Sensor

«GFunctionalRequirement» Priority: Medium

Description:

The software platform includes a Sensor that delivers the information if the reverse gear is enabled or not.

Rational:

The direction of movement is included in the vehicle speed. This information is only used to detect reverse gear or not.

Sensor Directory

«GFunctionalRequirement» Priority: Medium

Description:

Client Applications are able to query what Sensors are currently available.

Rational:

This allows for development of flexible applications that do not know what sensor data are available in the vehicle a priori. Client shall check first this directory to find out which ones are available; use meta-data to choose one of interest and use provided data to connect to necessary services.

Sensor Meta-Data

«GFunctionalRequirement»

Priority: High

Description:

The software platform provides the following information about the Sensor and the related output Signals:

- Sensor Identifier that is unique within the system
- Sensor Category (Physical/Logical)
- Sensor Type (GPS, Odometer, Map Matching, etc.)
- Sensor Sub-Type (ordinary GPS, differential GPS, etc.)
- Output Signals (Longitude, Latitude, Course, Speed, etc.)
- Output Signal Sampling Frequency (1 Hz, 10 Hz, irregular, etc.)
- Output Signal Measurement Units (kilometers per hour; meters per second; etc.)

Rational:

Sensor clients need that information in order to correctly handle data provided by sensor service and to adapt to the variation in the signal data delivery.

Sensor Signal Timestamp

«GFunctionalRequirement»

Priority: High

Description:

The software platform provides for each sample returned by the Sensor Service the timestamp, when it is accompanied. The timestamp corresponds to the time point of the sample acquisition or calculation. Timestamps are derived from the same clock that is accessible to the Client Applications. Timestamp is delivered with a accuracy of milliseconds.

Rational:

Measurement timestamps are important for proper functioning of most processing algorithms. For instance, algorithms for sensor calibration and dead reckoning typically use data from multiple sensors in conjunction, e.g. logical sensor.

Signal Measurement Units

«GFunctionalRequirement»

Priority: High

Description:

The software platform delivers signal values in universal, implementation independent units. It's preferred to use SI-units.

For example, a gyroscope signal should be measured in millidegrees per second instead of A/D converter counts.

Rational:

This decouples the client applications from the implementation details of individual sensor devices.

Signal Values Type Compatibility		
«GFunctionalRequirement»	Priority: Medium	
Description: All Sensor Services that provide Signals referring to the same physical quantity deliver their data in the same format (including API signatures, data type and measurement units). However, sampling frequency, accuracy etc. can differ.		
Rational: Sensor service clients are able to use multiple Sensor Services without changes in the interfaces.		

Simple Gyroscope Sensor Service		
«GFunctionalRequirement»	Priority: Medium	
Description: The software platform includes the Sensor that delivers <ul style="list-style-type: none"> - yaw rate: the rate of the vehicle heading change -temperature - status:(temperature compensated or not, etc) at the frequency of at least 5Hz. Unit of yaw rate is "degrees per second". Sign of yaw rate is defined by rule of right hand (thumb direction: up) (see reference system)		
Rational: This Sensor Service is used in Dead Reckoning calculations of the vehicle position.		

Slip Angle Sensor		
«GFunctionalRequirement»	Priority: Low	
Description: Platform provides a sensor, which delivers the value slip angle in degrees [°]. It is defined as the angle between the fixed car axis (direction of driving) and the real direction of vehicle movement. The direction and sign is defined equal to the yaw rate (See reference system).		
Rational: This Sensor is used for optimizations in Dead Reckoning calculations of the vehicle position.		

SteeringAngle Sensor		
«GFunctionalRequirement»	Priority: Low	
Description: This sensor provides the angles of the front and rear wheels and the steering wheel in degrees. Configuration values can be provided for sigmas and steering ratio.		
Rational: Is used as additional element for plausibilisation of the yaw rate in the dead reckoning module.		

Vehicle State Sensor		
«GFunctionalRequirement»	Priority: Low	
Description: The software platform provides a sensor, giving the state of certain vehicle systems: ABS: on/off ESP: on/off ASC: on/off (stability control) breaks: on/off Rational: This Sensor is used for optimizations in Dead Reckoning calculations of the vehicle position.		

VehicleSpeed Sensor		
«GFunctionalRequirement»	Priority: Medium	
Description: The software platform includes a Sensor that delivers the vehicle speed. Filtered vehicle speed in [m/s] with a frequency of at least 5Hz. Direction is given by the sign of this value. Rational: Vehicle speed is sometimes the only speed related signal available to the head unit.		

Wheel Tick/Speed Sensor Service		
«GFunctionalRequirement»	Priority: Medium	
Description: The software platform provides a Sensor that delivers the running counter of partial wheel revolutions at the frequency of at least 5Hz or the already calculated wheelspeed (speed in [m/s] or angular speed). The resolution of a single wheel revolution (i.e. the number of ticks per revolution) is included with the Sensor Service meta-data. This identifiers specify the wheel of measurement: 0: Average of non driven axle 1: Left front wheel 2: Right front wheel 3: Left rear wheel 4: Right rear wheel Unit: [ticks]. Rational: This Sensor typically registers ‘ticks’ from a wheel, adds them up and sends to the vehicle bus with a certain interval. The number of ‘ticks’ per complete wheel revolution is known in advance. In some cases, the data from multiple wheels are averaged. Other implementations send the already precalculated speed per wheel or axle, which is a valid replacement for most use cases.		

4. Architecture

1. GNSSService

The GNSSService is a component that abstracts the access to GNSS devices (e.g. GPS receivers).

It hides hardware and software dependencies on specific GNSS devices and their drivers.

2. *GNSSService Diagram*

This diagram shows the GNSSService and its interfaces.

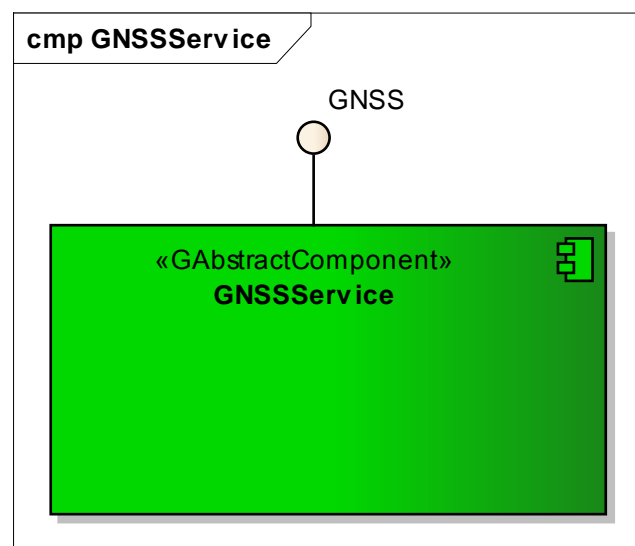


Figure: 2

3. Traceability Diagram

This diagram shows the software platform requirements and the use case realizations associated to the GNSSService.

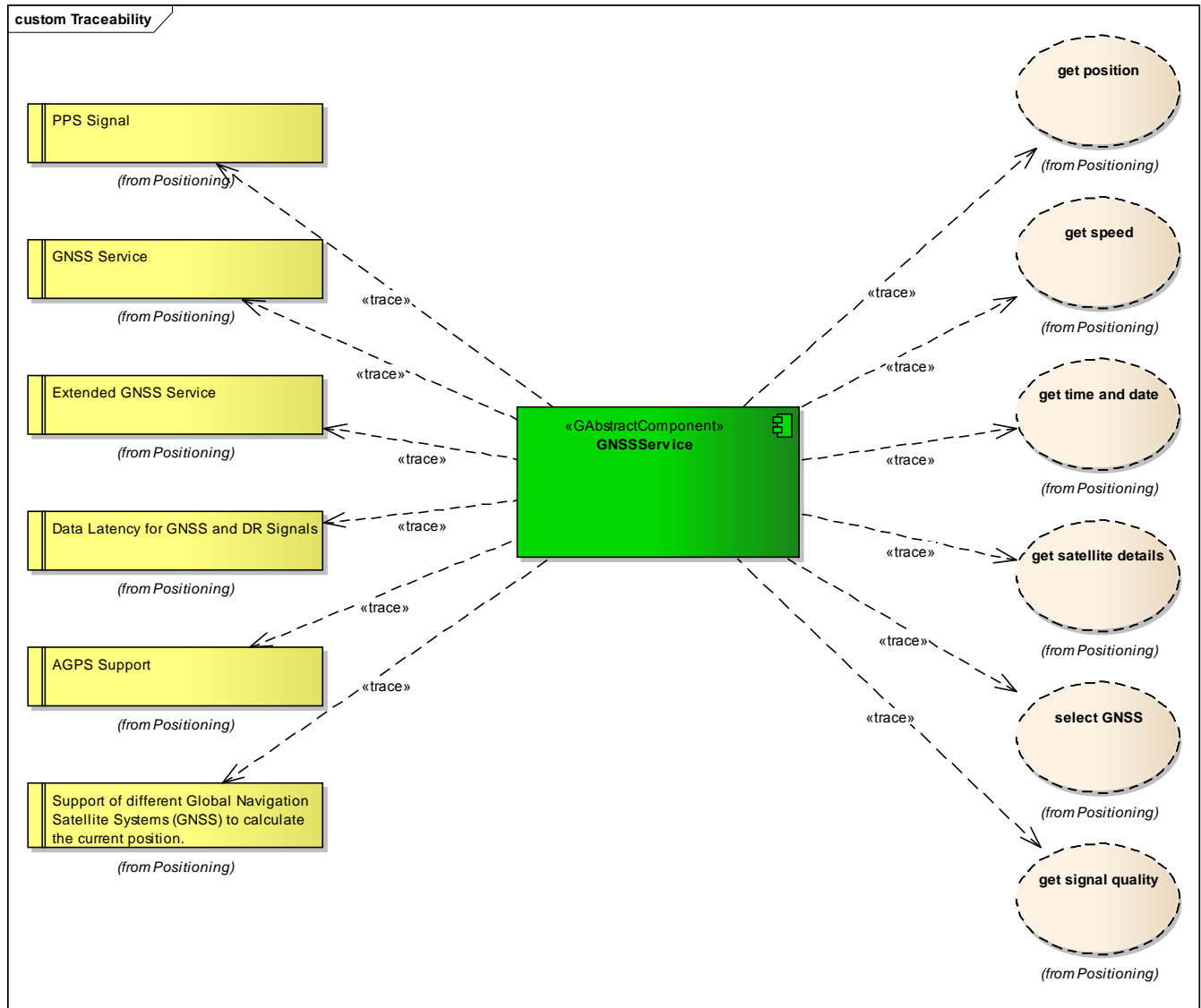


Figure: 3

4. Context Diagram

This diagram shows how the GNSSService component interacts with the SensorsService and the EnhancedPositionService.

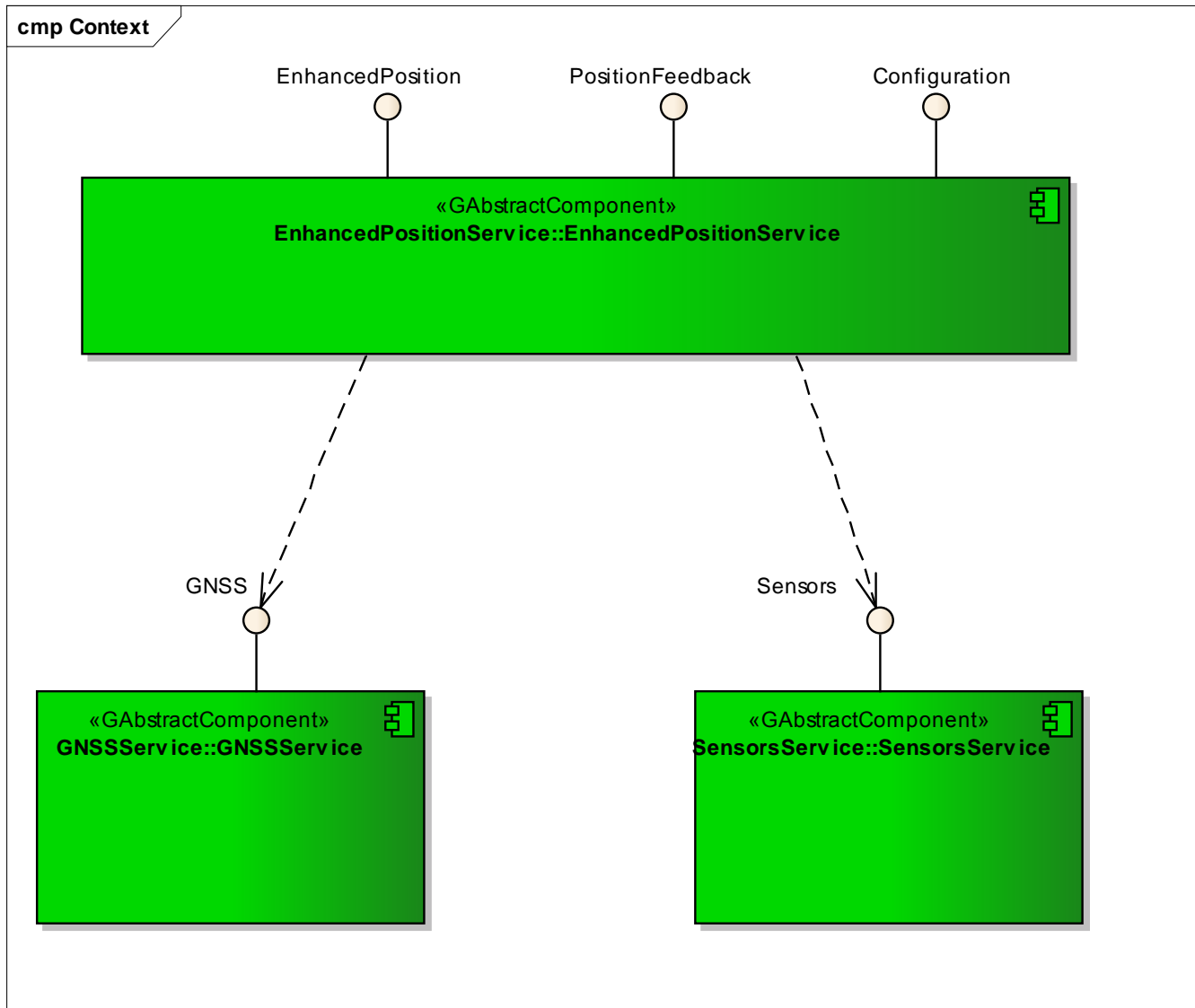


Figure: 4

GNSSService

Generated by Doxygen 1.8.6

Mon Jan 25 2016 17:25:06

Contents

1	Class Documentation	2
1.1	TGNSSConfiguration Struct Reference	2
1.1.1	Detailed Description	2
1.1.2	Member Data Documentation	2
1.2	TGNSSDistance3D Struct Reference	2
1.2.1	Detailed Description	2
1.2.2	Member Data Documentation	3
1.3	TGnssMetaData Struct Reference	3
1.3.1	Detailed Description	3
1.3.2	Member Data Documentation	3
1.4	TGNSSPosition Struct Reference	4
1.4.1	Detailed Description	4
1.4.2	Member Data Documentation	5
1.5	TGNSSSatelliteDetail Struct Reference	6
1.5.1	Detailed Description	7
1.5.2	Member Data Documentation	7
1.6	TGNSSStatus Struct Reference	7
1.6.1	Detailed Description	8
1.6.2	Member Data Documentation	8
1.7	TGNSSTime Struct Reference	8
1.7.1	Detailed Description	8
1.7.2	Member Data Documentation	9
2	File Documentation	9
2.1	gnss-init.h File Reference	9
2.1.1	Macro Definition Documentation	10
2.1.2	Function Documentation	10
2.2	gnss-meta-data.h File Reference	10
2.2.1	Enumeration Type Documentation	11
2.2.2	Function Documentation	11
2.3	gnss-status.h File Reference	12
2.3.1	Typedef Documentation	12
2.3.2	Enumeration Type Documentation	12
2.3.3	Function Documentation	13
2.4	gnss.h File Reference	14
2.4.1	Typedef Documentation	15
2.4.2	Enumeration Type Documentation	17
2.4.3	Function Documentation	20

1 Class Documentation

1.1 TGNSSConfiguration Struct Reference

```
#include <gnss.h>
```

Public Attributes

- [TGNSSDistance3D antennaPosition](#)
- [uint32_t supportedSystems](#)
- [uint32_t validityBits](#)

1.1.1 Detailed Description

Static configuration data related to the GNSS service.

1.1.2 Member Data Documentation

1.1.2.1 TGNSSDistance3D TGNSSConfiguration::antennaPosition

GNSS antenna position relative to the vehicle reference point.

1.1.2.2 [uint32_t](#) TGNSSConfiguration::supportedSystems

Bit mask indicating the satellite systems which are supported by the GNSS hardware [bitwise or'ed [EGNSSSystem](#) values].

1.1.2.3 [uint32_t](#) TGNSSConfiguration::validityBits

Bit mask indicating the validity of each corresponding value. [bitwise or'ed [EGNSSConfigValidityBits](#) values]. Must be checked before usage.

The documentation for this struct was generated from the following file:

- [gnss.h](#)

1.2 TGNSSDistance3D Struct Reference

```
#include <gnss.h>
```

Public Attributes

- [float](#) [x](#)
- [float](#) [y](#)
- [float](#) [z](#)

1.2.1 Detailed Description

3 dimensional distance used for description of geometric descriptions within the vehicle reference system.

The vehicle axis system as defined in ISO 8855:2011(E). In this system, the axes (Xv, Yv, Zv) are oriented as follows

- Xv is in the horizontal plane, pointing forwards
- Yv is in the horizontal plane, pointing to the left
- Zv is perpendicular to the horizontal plane, pointing upwards For an illustration, see <https://collab.-genivi.org/wiki/display/genivi/LBSSensorServiceRequirementsBorg#LBS-SensorServiceRequirementsBorg-ReferenceSystem>

The reference point of the vehicle lies underneath the center of the rear axle on the surface of the road.

1.2.2 Member Data Documentation

1.2.2.1 float TGNSSDistance3D::x

Distance in x direction in [m] according to the reference coordinate system.

1.2.2.2 float TGNSSDistance3D::y

Distance in y direction in [m] according to the reference coordinate system.

1.2.2.3 float TGNSSDistance3D::z

Distance in z direction in [m] according to the reference coordinate system.

The documentation for this struct was generated from the following file:

- [gnss.h](#)

1.3 TGnssMetaData Struct Reference

```
#include <gnss-meta-data.h>
```

Public Attributes

- uint32_t [version](#)
- [EGnssCategory](#) category
- uint32_t [typeBits](#)
- uint32_t [cycleTime](#)
- uint16_t [numChannels](#)

1.3.1 Detailed Description

The software platform provides the following information about the GNSS output signals. GNSS clients need the meta data information in order to correctly handle data provided by GNSS service and to adapt to the variation in the signal data delivery.

1.3.2 Member Data Documentation

1.3.2.1 [EGnssCategory](#) TGnssMetaData::category

GNSS Category (Physical/Logical).

1.3.2.2 uint32_t TGnssMetaData::cycleTime

GNSS cycle time (update interval) in ms. 0 for irregular updates

1.3.2.3 uint16_t TGnssMetaData::numChannels

Number of GNSS receiver channels for satellite signal reception.

1.3.2.4 uint32_t TGnssMetaData::typeBits

GNSS Type: combination of bits defined in [EGnssTypeBits](#).

1.3.2.5 uint32_t TGnssMetaData::version

Version of the GNSS service.

The documentation for this struct was generated from the following file:

- [gnss-meta-data.h](#)

1.4 TGNSSPosition Struct Reference

```
#include <gnss.h>
```

Public Attributes

- uint64_t [timestamp](#)
- double [latitude](#)
- double [longitude](#)
- float [altitudeMSL](#)
- float [altitudeEll](#)
- float [hSpeed](#)
- float [vSpeed](#)
- float [heading](#)
- float [pdop](#)
- float [hdop](#)
- float [vdop](#)
- uint16_t [usedSatellites](#)
- uint16_t [trackedSatellites](#)
- uint16_t [visibleSatellites](#)
- float [sigmaHPosition](#)
- float [sigmaAltitude](#)
- float [sigmaHSpeed](#)
- float [sigmaVSpeed](#)
- float [sigmaHeading](#)
- [EGNSSFixStatus](#) [fixStatus](#)
- uint32_t [fixTypeBits](#)
- uint32_t [activatedSystems](#)
- uint32_t [usedSystems](#)
- uint32_t [validityBits](#)

1.4.1 Detailed Description

GNSS position data including velocity, status and accuracy. This data structure provides all GNSS information which is typically needed for positioning applications such as GNSS/Dead Reckoning sensor fusion.

1.4.2 Member Data Documentation

1.4.2.1 uint32_t TGNSSPosition::activatedSystems

Bit mask indicating the satellite systems that are activated for use [bitwise or'ed [EGNSSSystem](#) values].

1.4.2.2 float TGNSSPosition::altitudeEll

Altitude above WGS84 ellipsoid in [m].

1.4.2.3 float TGNSSPosition::altitudeMSL

Altitude above mean sea level (geoid) in [m].

1.4.2.4 EGNSSFixStatus TGNSSPosition::fixStatus

Value representing the GNSS mode.

1.4.2.5 uint32_t TGNSSPosition::fixTypeBits

Bit mask indicating the sources actually used for the GNSS calculation. [bitwise or'ed [EGNSSFixType](#) values].

1.4.2.6 float TGNSSPosition::hdop

The horizontal (2D) dilution of precision.

1.4.2.7 float TGNSSPosition::heading

GNSS course angle [degree] (0 => north, 90 => east, 180 => south, 270 => west, no negative values).

1.4.2.8 float TGNSSPosition::hSpeed

Horizontal speed [m/s].

1.4.2.9 double TGNSSPosition::latitude

Latitude in WGS84 in [degree].

1.4.2.10 double TGNSSPosition::longitude

Longitude in WGS84 in [degree].

1.4.2.11 float TGNSSPosition::pdop

The positional (3D) dilution of precision. [Note: $pdop^2 = hdop^2 + vdop^2$]

1.4.2.12 float TGNSSPosition::sigmaAltitude

Standard error estimate of altitude in [m].

1.4.2.13 float TGNSSPosition::sigmaHeading

Standard error estimate of horizontal heading/course in [degree].

1.4.2.14 float TGNSSPosition::sigmaHPosition

Standard error estimate of the horizontal position in [m].

1.4.2.15 float TGNSSPosition::sigmaHSpeed

Standard error estimate of horizontal speed in [m/s].

1.4.2.16 float TGNSSPosition::sigmaVSpeed

Standard error estimate of vertical speed in [m/s].

1.4.2.17 uint64_t TGNSSPosition::timestamp

Timestamp of the acquisition of the GNSS data [ms]. All sensor/GNSS timestamps must be based on the same time source.

1.4.2.18 uint16_t TGNSSPosition::trackedSatellites

Number of satellites from which a signal is received.

1.4.2.19 uint16_t TGNSSPosition::usedSatellites

Number of satellites used for the GNSS fix.

1.4.2.20 uint32_t TGNSSPosition::usedSystems

Bit mask indicating the satellite systems that are actually used for the position fix [bitwise or'ed [EGNSSSystem](#) values].

1.4.2.21 uint32_t TGNSSPosition::validityBits

Bit mask indicating the validity of each corresponding value. [bitwise or'ed [EGNSSPositionValidityBits](#) values]. Must be checked before usage.

1.4.2.22 float TGNSSPosition::vdop

The vertical (altitude) dilution of precision.

1.4.2.23 uint16_t TGNSSPosition::visibleSatellites

Number of satellites expected to be receivable, i.e. above horizon or elevation mask.

1.4.2.24 float TGNSSPosition::vSpeed

Vertical speed [m/s].

The documentation for this struct was generated from the following file:

- [gnss.h](#)

1.5 TGNSSSatelliteDetail Struct Reference

```
#include <gnss.h>
```

Public Attributes

- uint64_t [timestamp](#)
- [EGNSSSystem](#) [system](#)
- uint16_t [satelliteId](#)
- uint16_t [azimuth](#)
- uint16_t [elevation](#)
- uint16_t [SNR](#)
- uint32_t [statusBits](#)
- int16_t [posResidual](#)
- uint32_t [validityBits](#)

1.5.1 Detailed Description

Detailed data from one GNSS satellite.

1.5.2 Member Data Documentation

1.5.2.1 uint16_t TGNSSSatelliteDetail::azimuth

Satellite Azimuth in degrees. Value range 0..359

1.5.2.2 uint16_t TGNSSSatelliteDetail::elevation

Satellite Elevation in degrees. Value range 0..90

1.5.2.3 int16_t TGNSSSatelliteDetail::posResidual

Residual in m of position calculation. Range -999 to +999, 0 if not tracking

1.5.2.4 uint16_t TGNSSSatelliteDetail::satelliteId

Satellite ID. Satellite IDs are only unique within one satellite system. Satellites of different systems can be distinguished by [TGNSSSatelliteDetail::system](#). Ranges: 1..32: GPS satellites (by PRN) 33..64: SBAS/WAAS satellites 65..96: GLONASS satellites 1..64: GALILEO satellites, Galileo OS SIS ICD, <http://www.gsc-europa.eu/gnss-markets/segments-applications/os-sis-icd>.

1.5.2.5 uint16_t TGNSSSatelliteDetail::SNR

SNR (C/No) in dBHz. Range 0 to 99, null when not tracking

1.5.2.6 uint32_t TGNSSSatelliteDetail::statusBits

Bit mask of additional status flags. [bitwise or'ed [EGNSSSatelliteFlag](#) values].

1.5.2.7 EGNSSSystem TGNSSSatelliteDetail::system

Value representing the GNSS system.

1.5.2.8 uint64_t TGNSSSatelliteDetail::timestamp

Timestamp of the acquisition of the satellite detail data [ms]. All sensor/GNSS timestamps must be based on the same time source.

1.5.2.9 uint32_t TGNSSSatelliteDetail::validityBits

Bit mask indicating the validity of each corresponding value. [bitwise or'ed [EGNSSSatelliteDetailValidityBits](#) values]. Must be checked before usage.

The documentation for this struct was generated from the following file:

- [gnss.h](#)

1.6 TGNSSStatus Struct Reference

```
#include <gnss-status.h>
```

Public Attributes

- uint64_t [timestamp](#)
- [EGNSSStatus](#) [status](#)

- [EGNSSAntennaStatus antStatus](#)
- [uint32_t validityBits](#)

1.6.1 Detailed Description

Container for GNSS status information

1.6.2 Member Data Documentation

1.6.2.1 [EGNSSAntennaStatus](#) [TGNSSStatus::antStatus](#)

Status of the GNSS antenna

1.6.2.2 [EGNSSStatus](#) [TGNSSStatus::status](#)

Status of the GNSS receiver

1.6.2.3 [uint64_t](#) [TGNSSStatus::timestamp](#)

Timestamp of the GNSS status transition [ms]. All sensor/GNSS timestamps must be based on the same time source.

1.6.2.4 [uint32_t](#) [TGNSSStatus::validityBits](#)

Bit mask indicating the validity of each corresponding value. [bitwise or'ed [EGNSSStatusValidityBits](#) values]. Must be checked before usage.

The documentation for this struct was generated from the following file:

- [gnss-status.h](#)

1.7 TGNSSTime Struct Reference

```
#include <gnss.h>
```

Public Attributes

- [uint64_t](#) [timestamp](#)
- [uint16_t](#) [year](#)
- [uint8_t](#) [month](#)
- [uint8_t](#) [day](#)
- [uint8_t](#) [hour](#)
- [uint8_t](#) [minute](#)
- [uint8_t](#) [second](#)
- [uint16_t](#) [ms](#)
- [EGNSTimeScale](#) [scale](#)
- [int8_t](#) [leapSeconds](#)
- [uint32_t](#) [validityBits](#)

1.7.1 Detailed Description

Provides the current date and time according UTC (Coordinated Universal Time) Note: the uncommon numbering of day and month is chosen to be compatible with the struct tm from the standard C-Library

1.7.2 Member Data Documentation

1.7.2.1 `uint8_t TGNSSTime::day`

Day of month fraction of the UTC time. Unit: [day]. Number between 1 and 31

1.7.2.2 `uint8_t TGNSSTime::hour`

Hour fraction of the UTC time. Unit: [hour] Number between 0 and 23

1.7.2.3 `int8_t TGNSSTime::leapSeconds`

Number of leap seconds, i.e. difference between GPS time and UTC. Unit: [seconds]. Note: value before 01-July-2015: 16; from 01-July-2015: 17; further changes possible.

1.7.2.4 `uint8_t TGNSSTime::minute`

Minute fraction of the UTC time. Unit: [minutes] Number between 0 and 59

1.7.2.5 `uint8_t TGNSSTime::month`

Month fraction of the UTC time. Unit: [month] Number between 0 and 11

1.7.2.6 `uint16_t TGNSSTime::ms`

Millisecond fraction of the UTC time. Unit: [milliseconds] Number between 0 and 999

1.7.2.7 `EGNSSTimeScale TGNSSTime::scale`

Time scale used: UTC or GPS.

1.7.2.8 `uint8_t TGNSSTime::second`

Second fraction of the UTC time. Unit: [seconds] Number between 0 and 59. In case of a leap second this value is 60.

1.7.2.9 `uint64_t TGNSSTime::timestamp`

Timestamp of the acquisition of the UTC date/time [ms]. All sensor/GNSS timestamps must be based on the same time source.

1.7.2.10 `uint32_t TGNSSTime::validityBits`

Bit mask indicating the validity of each corresponding value. [bitwise or'ed [EGNSSTimeValidityBits](#) values]. Must be checked before usage.

1.7.2.11 `uint16_t TGNSSTime::year`

Year fraction of the UTC time. Unit: [year] Number equivalent to the year (4 digits)

The documentation for this struct was generated from the following file:

- [gnss.h](#)

2 File Documentation

2.1 `gnss-init.h` File Reference

```
#include <stdbool.h>
```

Macros

- `#define GENIVI_GNSS_API_MAJOR 4`
- `#define GENIVI_GNSS_API_MINOR 0`
- `#define GENIVI_GNSS_API_MICRO 0`

Functions

- `bool gnssInit ()`
- `bool gnssDestroy ()`
- `void gnssGetVersion (int *major, int *minor, int *micro)`

2.1.1 Macro Definition Documentation

2.1.1.1 `#define GENIVI_GNSS_API_MAJOR 4`

2.1.1.2 `#define GENIVI_GNSS_API_MICRO 0`

2.1.1.3 `#define GENIVI_GNSS_API_MINOR 0`

2.1.2 Function Documentation

2.1.2.1 `bool gnssDestroy ()`

Destroy the GNSS service. Must be called after using the GNSS service to shut down the service.

Returns

True if shutdown has been successful.

2.1.2.2 `void gnssGetVersion (int * major, int * minor, int * micro)`

GNSS services version information. This information is for the GNSS services system structure. The version information for each specific GNSS component can be obtained via the metadata.

Parameters

<i>major</i>	Major version number. Changes in this number are used for incompatible API change.
<i>minor</i>	Minor version number. Changes in this number are used for compatible API change.
<i>micro</i>	Micro version number. Changes in this number are used for minor changes.

2.1.2.3 `bool gnssInit ()`

Initialization of the GNSS service. Must be called before using the GNSS service to set up the service.

Returns

True if initialization has been successful.

2.2 gnss-meta-data.h File Reference

```
#include <stdint.h>
```

Classes

- struct `TGnssMetaData`

Enumerations

- enum `EGnssCategory` { `GNSS_CATEGORY_UNKNOWN`, `GNSS_CATEGORY_LOGICAL`, `GNSS_CATEGORY_PHYSICAL` }
- enum `EGnssTypeBits` { `GNSS_TYPE_GNSS` = 0x00000001, `GNSS_TYPE_ASSISTED` = 0x00000002, `GNSS_TYPE_SBAS` = 0x00000004, `GNSS_TYPE_DGPS` = 0x00000008, `GNSS_TYPE_DR` = 0x00000010 }

Functions

- bool `gnssGetMetaData` (`TGnssMetaData` *data)

2.2.1 Enumeration Type Documentation

2.2.1.1 enum `EGnssCategory`

The GNSS category introduces the concept that sensor information can also be derived information computed by combining several signals.

Enumerator

`GNSS_CATEGORY_UNKNOWN` Unknown category. Should not be used.

`GNSS_CATEGORY_LOGICAL` A logical GNSS service can combine the signal of a GNSS receiver with additional sources.

`GNSS_CATEGORY_PHYSICAL` A physical GNSS service, i.e. a stand-alone GNSS receiver.

2.2.1.2 enum `EGnssTypeBits`

`TGnssMetaData`:typeBits provides information about the sources used for the GNSS calculation It is a or'ed bitmask of the `EGnssTypeBits` values.

Enumerator

`GNSS_TYPE_GNSS` GNSS receiver. Should always be set.

`GNSS_TYPE_ASSISTED` GNSS receiver with support for Assisted GNSS. E.g. ephemeris or clock data can be provided over network for faster TTFF

`GNSS_TYPE_SBAS` GNSS receiver with support for SBAS (satellite based augmentation system), such as WAAS, EGNOS, ...

`GNSS_TYPE_DGPS` GNSS receiver with support for differential GPS

`GNSS_TYPE_DR` GNSS receiver with built in dead reckoning sensor fusion

2.2.2 Function Documentation

2.2.2.1 bool `gnssGetMetaData` (`TGnssMetaData` * data)

Provide meta information about GNSS service. The meta data of a service provides information about it's name, version, type, subtype, sampling frequency etc.

Parameters

<i>data</i>	Meta data content about the sensor service.
-------------	---

Returns

True if meta data is available.

2.3 gnss-status.h File Reference

Classes

- struct [TGNSSStatus](#)

Typedefs

- typedef void(* [GNSSStatusCallback](#))(const [TGNSSStatus](#) *status)

Enumerations

- enum [EGNSSStatus](#) {
[GNSS_STATUS_NOTAVAILABLE](#) = 0, [GNSS_STATUS_INITIALIZING](#) = 1, [GNSS_STATUS_AVAILABLE](#) = 2, [GNSS_STATUS_RESTARTING](#) = 3,
[GNSS_STATUS_FAILURE](#) = 4, [GNSS_STATUS_OUTOFSERVICE](#) = 5 }
- enum [EGNSSAntennaStatus](#) {
[GNSS_ANT_STATUS_NORMAL](#) = 0, [GNSS_ANT_STATUS_OVERCURRENT](#) = 1, [GNSS_ANT_STATUS_OPEN](#) = 2, [GNSS_ANT_STATUS_SHORT_GND](#) = 3,
[GNSS_ANT_STATUS_SHORT_BATT](#) = 4, [GNSS_ANT_STATUS_OUTOFSERVICE](#) = 5 }
- enum [EGNSSStatusValidityBits](#) { [GNSS_STATUS_STATUS_VALID](#) = 0x00000001, [GNSS_STATUS_ANT_STATUS_VALID](#) = 0x00000002 }

Functions

- bool [gnssGetStatus](#) ([TGNSSStatus](#) *status)
- bool [gnssRegisterStatusCallback](#) ([GNSSStatusCallback](#) callback)
- bool [gnssDeregisterStatusCallback](#) ([GNSSStatusCallback](#) callback)

2.3.1 Typedef Documentation

2.3.1.1 typedef void(* GNSSStatusCallback)(const TGNSSStatus *status)

Callback type for gnss status. Use this type of callback if you want to register for gnss status updates data.

Parameters

<i>status</i>	the gnss status
---------------	-----------------

2.3.2 Enumeration Type Documentation

2.3.2.1 enum EGNSSAntennaStatus

Enumeration to describe the status of the GNSS antenna

Enumerator

GNSS_ANT_STATUS_NORMAL GNSS antenna is working in normal operation.

GNSS_ANT_STATUS_OVERCURRENT GNSS antenna is working but the antenna current is higher than expected.

GNSS_ANT_STATUS_OPEN GNSS antenna is not working because not connected (antenna current too low).

GNSS_ANT_STATUS_SHORT_GND GNSS antenna is not working due to short-circuit to ground.

GNSS_ANT_STATUS_SHORT_BATT GNSS antenna is not working due to short-circuit to battery.

GNSS_ANT_STATUS_OUTOFSERVICE GNSS antenna is temporarily not available, due to some known external condition.

2.3.2.2 enum EGNSSStatus

Enumeration to describe the status of the GNSS receiver

Enumerator

GNSS_STATUS_NOTAVAILABLE GNSS is not available at all, based on configuration data.

GNSS_STATUS_INITIALIZING Initial status when the connection to the GNSS is set up for the first time.

GNSS_STATUS_AVAILABLE GNSS is available and running as expected.

GNSS_STATUS_RESTARTING GNSS is restarted, i.e. due to communication loss.

GNSS_STATUS_FAILURE GNSS is not operating properly. Restarting did not help.

GNSS_STATUS_OUTOFSERVICE GNSS is temporarily not available, due to some known external condition, e.g. firmware update or switch off for antenna supervision.

2.3.2.3 enum EGNSSStatusValidityBits

[TGNSSStatus::validityBits](#) provides information about the currently valid signals of the [TGNSSStatus](#) struct. It is a or'ed bitmask of the EGNSSStatusValidityBits values.

Enumerator

GNSS_STATUS_STATUS_VALID Validity bit for field [TGNSSStatus::status](#).

GNSS_STATUS_ANT_STATUS_VALID Validity bit for field [TGNSSStatus::antStatus](#).

2.3.3 Function Documentation

2.3.3.1 bool gnssDeregisterStatusCallback ([GNSSStatusCallback](#) *callback*)

Deregister gnss status callback. After calling this method no new gnss status updates will be delivered to the client.

Parameters

<i>callback</i>	The callback which should be deregistered.
-----------------	--

Returns

True if callback has been deregistered successfully.

2.3.3.2 bool gnssGetStatus ([TGNSSStatus](#) * *status*)

Method to get the gyroscope status at a specific point in time.

Parameters

<i>status</i>	After calling the method the current gyroscope status is written into status
---------------	--

Returns

Is true if data can be provided and false otherwise, e.g. missing initialization

2.3.3.3 bool gnssRegisterStatusCallback ([GNSSStatusCallback](#) *callback*)

Register gnss status callback. This is the recommended method for continuously monitoring the gyroscope status. The callback will be invoked when new gyroscope status data is available.

Parameters

<i>callback</i>	The callback which should be registered.
-----------------	--

Returns

True if callback has been registered successfully.

2.4 gnss.h File Reference

```
#include "gnss-meta-data.h"
#include <stdint.h>
#include <stdbool.h>
```

Classes

- struct [TGNSSDistance3D](#)
- struct [TGNSSConfiguration](#)
- struct [TGNSSTime](#)
- struct [TGNSSSatelliteDetail](#)
- struct [TGNSSPosition](#)

Typedefs

- typedef void(* [GNSSTimeCallback](#))(const [TGNSSTime](#) time[], uint16_t numElements)
- typedef void(* [GNSSSatelliteDetailCallback](#))(const [TGNSSSatelliteDetail](#) satelliteDetail[], uint16_t numElements)
- typedef void(* [GNSSPositionCallback](#))(const [TGNSSPosition](#) position[], uint16_t numElements)

Enumerations

- enum [EGNSSFixStatus](#) { [GNSS_FIX_STATUS_NO](#), [GNSS_FIX_STATUS_TIME](#), [GNSS_FIX_STATUS_2D](#), [GNSS_FIX_STATUS_3D](#) }
- enum [EGNSSConfigValidityBits](#) { [GNSS_CONFIG_ANTPOS_VALID](#) = 0x00000001, [GNSS_CONFIG_SATSYS_VALID](#) = 0x00000002 }
- enum [EGNSSFixType](#) { [GNSS_FIX_TYPE_SINGLE_FREQUENCY](#) = 0x00000001, [GNSS_FIX_TYPE_MULTI_FREQUENCY](#) = 0x00000002, [GNSS_FIX_TYPE_MULTI_CONSTELLATION](#) = 0x00000004, [GNSS_FIX_TYPE_PPP](#) = 0x00000010, [GNSS_FIX_TYPE_INTEGRITY_CHECKED](#) = 0x00000020, [GNSS_FIX_TYPE_SBAS](#) = 0x00001000, [GNSS_FIX_TYPE_DGNSS](#) = 0x00002000, [GNSS_FIX_TYPE_RTK_FIXED](#) = 0x00004000, [GNSS_FIX_TYPE_RTK_FLOAT](#) = 0x00008000, [GNSS_FIX_TYPE_SSR](#) = 0x00010000, [GNSS_FIX_TYPE_ESTIMATED](#) = 0x00100000, [GNSS_FIX_TYPE_DEAD_RECKONING](#) = 0x00200000, [GNSS_FIX_TYPE_MANUAL](#) = 0x10000000, [GNSS_FIX_TYPE_SIMULATOR_MODE](#) = 0x20000000 }
- enum [EGNSSTimeScale](#) { [GNSS_TIME_SCALE_UTC](#) = 0, [GNSS_TIME_SCALE_GPS](#) = 1 }
- enum [EGNSSTimeValidityBits](#) { [GNSS_TIME_TIME_VALID](#) = 0x00000001, [GNSS_TIME_DATE_VALID](#) = 0x00000002, [GNSS_TIME_SCALE_VALID](#) = 0x00000004, [GNSS_TIME_LEAPSEC_VALID](#) = 0x00000008 }
- enum [EGNSSSystem](#) { [GNSS_SYSTEM_GPS](#) = 0x00000001, [GNSS_SYSTEM_GLOPASS](#) = 0x00000002, [GNSS_SYSTEM_GALILEO](#) = 0x00000004, [GNSS_SYSTEM_BEIDOU](#) = 0x00000008, [GNSS_SYSTEM_GPS_L2](#) = 0x00000010, [GNSS_SYSTEM_GPS_L5](#) = 0x00000020, [GNSS_SYSTEM_GLOPASS_L2](#) = 0x00000040, [GNSS_SYSTEM_BEIDOU_B2](#) = 0x00000080, [GNSS_SYSTEM_SBAS_WAAS](#) = 0x00010000, [GNSS_SYSTEM_SBAS_EGNOS](#) = 0x00020000, [GNSS_SYSTEM_SBAS_MSAS](#) = 0x00040000, [GNSS_SYSTEM_SBAS_QZSS_SAIF](#) = 0x00080000, [GNSS_SYSTEM_SBAS_SDCM](#) = 0x00100000, [GNSS_SYSTEM_SBAS_GAGAN](#) = 0x00200000 }

- enum [EGNSSatelliteFlag](#) { [GNSS_SATELLITE_USED](#) = 0x00000001, [GNSS_SATELLITE_EPHEMERIS_AVAILABLE](#) = 0x00000002 }
- enum [EGNSSatelliteDetailValidityBits](#) {
[GNSS_SATELLITE_SYSTEM_VALID](#) = 0x00000001, [GNSS_SATELLITE_ID_VALID](#) = 0x00000002, [GNSS_SATELLITE_AZIMUTH_VALID](#) = 0x00000004, [GNSS_SATELLITE_ELEVATION_VALID](#) = 0x00000008,
[GNSS_SATELLITE_SNR_VALID](#) = 0x00000010, [GNSS_SATELLITE_USED_VALID](#) = 0x00000020, [GNSS_SATELLITE_EPHEMERIS_AVAILABLE_VALID](#) = 0x00000040, [GNSS_SATELLITE_RESIDUAL_VALID](#) = 0x00000080 }
- enum [EGNSSPositionValidityBits](#) {
[GNSS_POSITION_LATITUDE_VALID](#) = 0x00000001, [GNSS_POSITION_LONGITUDE_VALID](#) = 0x00000002,
[GNSS_POSITION_ALTITUDEMSL_VALID](#) = 0x00000004, [GNSS_POSITION_ALTITUDEELL_VALID](#) = 0x00000008,
[GNSS_POSITION_HSPEED_VALID](#) = 0x00000010, [GNSS_POSITION_VSPEED_VALID](#) = 0x00000020,
[GNSS_POSITION_HEADING_VALID](#) = 0x00000040, [GNSS_POSITION_PDOP_VALID](#) = 0x00000080,
[GNSS_POSITION_HDOP_VALID](#) = 0x00000100, [GNSS_POSITION_VDOP_VALID](#) = 0x00000200, [GNSS_POSITION_USAT_VALID](#) = 0x00000400, [GNSS_POSITION_TSAT_VALID](#) = 0x00000800,
[GNSS_POSITION_VSAT_VALID](#) = 0x00001000, [GNSS_POSITION_SHPOS_VALID](#) = 0x00002000, [GNSS_POSITION_SALT_VALID](#) = 0x00004000, [GNSS_POSITION_SHSPEED_VALID](#) = 0x00008000,
[GNSS_POSITION_SVSPEED_VALID](#) = 0x00010000, [GNSS_POSITION_SHEADING_VALID](#) = 0x00020000,
[GNSS_POSITION_STAT_VALID](#) = 0x00040000, [GNSS_POSITION_TYPE_VALID](#) = 0x00080000,
[GNSS_POSITION_ASYS_VALID](#) = 0x00100000, [GNSS_POSITION_USYS_VALID](#) = 0x00200000 }

Functions

- bool [gnssGetConfiguration](#) ([TGNSSConfiguration](#) *gnssConfig)
- bool [gnssGetTime](#) ([TGNSSTime](#) *utc)
- bool [gnssRegisterTimeCallback](#) ([GNSSTimeCallback](#) callback)
- bool [gnssDeregisterTimeCallback](#) ([GNSSTimeCallback](#) callback)
- bool [gnssGetSatelliteDetails](#) ([TGNSSSatelliteDetail](#) *satelliteDetails, [uint16_t](#) count, [uint16_t](#) *numSatelliteDetails)
- bool [gnssRegisterSatelliteDetailCallback](#) ([GNSSSatelliteDetailCallback](#) callback)
- bool [gnssDeregisterSatelliteDetailCallback](#) ([GNSSSatelliteDetailCallback](#) callback)
- bool [gnssGetPosition](#) ([TGNSSPosition](#) *position)
- bool [gnssRegisterPositionCallback](#) ([GNSSPositionCallback](#) callback)
- bool [gnssDeregisterPositionCallback](#) ([GNSSPositionCallback](#) callback)
- bool [gnssGetPrecisionTimingOffset](#) ([int32_t](#) *delta)
- bool [gnssSetGNSSSystems](#) ([uint32_t](#) activateSystems)

2.4.1 Typedef Documentation

2.4.1.1 typedef void(* GNSSPositionCallback)(const TGNSSPosition position[], [uint16_t](#) numElements)

Callback type for GNSS position data Use this type of callback if you want to register for GNSS position data. This callback may return buffered data (`numElements > 1`) for different reasons for (large) portions of data buffered at startup for data buffered during regular operation e.g. for performance optimization (reduction of callback invocation frequency) If the array contains (`numElements > 1`), the elements will be ordered with rising timestamps

Parameters

<i>position</i>	pointer to an array of TGNSSPosition with size <code>numElements</code>
<i>numElements</i>	allowed range: ≥ 1 . If <code>numElements > 1</code> , buffered data are provided.

2.4.1.2 typedef void(* GNSSSatelliteDetailCallback)(const TGNSSSatelliteDetail satelliteDetail[], [uint16_t](#) numElements)

Callback type for GNSS satellite details. Use this type of callback if you want to register for GNSS satellite detail data. This callback may return buffered data (`numElements > 1`) for different reasons for (large) portions of data

buffered at startup for data buffered during regular operation e.g. for performance optimization (reduction of callback invocation frequency) If the array contains (`numElements > 1`), the elements will be ordered with rising timestamps

Parameters

<i>time</i>	pointer to an array of TGNSSSatelliteDetail with size numElements
<i>numElements</i>	allowed range: ≥ 1 . If numElements > 1 , buffered data are provided.

2.4.1.3 typedef void(* GNSSTimeCallback)(const TGNSSTime time[], uint16_t numElements)

Callback type for GNSS UTC date and time. Use this type of callback if you want to register for GNSS UTC time data. This callback may return buffered data (numElements > 1) for different reasons for (large) portions of data buffered at startup for data buffered during regular operation e.g. for performance optimization (reduction of callback invocation frequency) If the array contains (numElements > 1), the elements will be ordered with rising timestamps

Parameters

<i>time</i>	pointer to an array of TGNSSTime with size numElements
<i>numElements</i>	allowed range: ≥ 1 . If numElements > 1 , buffered data are provided.

2.4.2 Enumeration Type Documentation

2.4.2.1 enum EGNSSConfigValidityBits

[TGNSSConfiguration::validityBits](#) provides information about the currently valid signals of the GNSS configuration data. It is a or'ed bitmask of the EGNSSConfigValidityBits values.

Enumerator

GNSS_CONFIG_ANTPOS_VALID Validity bit for field [TGNSSConfiguration::antennaPosition](#).

GNSS_CONFIG_SATSYS_VALID Validity bit for field [TGNSSConfiguration::supportedSystems](#).

2.4.2.2 enum EGNSSFixStatus

Description of the fix status of the GNSS receiver.

Enumerator

GNSS_FIX_STATUS_NO GNSS has no fix, i.e. position, velocity, time cannot be determined

GNSS_FIX_STATUS_TIME GNSS can only determine the time, but not position and velocity

GNSS_FIX_STATUS_2D GNSS has a 2D fix, i.e. the horizontal position can be determined but not the altitude. This implies that also velocity and time are available.

GNSS_FIX_STATUS_3D GNSS has a 3D fix, i.e. position can be determined including the altitude. This implies that also velocity and time are available.

2.4.2.3 enum EGNSSFixType

[TGNSSAccuracy::fixTypeBits](#) provides GNSS Fix Type indication. I.e. it identifies the sources actually used for the GNSS calculation It is a or'ed bitmask of the EGNSSFixType values. The bit values have been grouped logically with gaps where future extensions can be foreseen Within one group, not all combinations make necessarily sense Between different groups, all combinations should make sense

Enumerator

GNSS_FIX_TYPE_SINGLE_FREQUENCY GNSS satellite data are received on a single frequency. A typical example is GPS using only the C/A code on the L1 frequency. It e.g. also applies to a combined GPS(-L1)/Galileo(E1) fix since L1 and E1 share the same frequency.

GNSS_FIX_TYPE_MULTI_FREQUENCY GNSS satellite data are received on a multiple frequencies. This enables the receiver to correct frequency-dependent errors such as for ionospheric delays. An example could be a GPS receiver receiving on the L1 and L2C band.

GNSS_FIX_TYPE_MULTI_CONSTELLATION GNSS satellite data are received and used for the fix from more than one GNSS system. For example, the fix could be calculated from GPS and GLONASS. This is also possible for single frequency as several GNSS systems share the same frequencies.

GNSS_FIX_TYPE_PPP PPP = Precise Point Positioning An improved precision is achieved without differential corrections. This is possible even for single frequency receivers, e.g. by using carrier phase tracking

GNSS_FIX_TYPE_INTEGRITY_CHECKED Additional integrity checks have been done to ensure the correctness of the fix.

GNSS_FIX_TYPE_SBAS SBAS = Satellite Based Augmentation System Correction data from an SBAS system such as WAAS, EGNOS, ... are taken into account

GNSS_FIX_TYPE_DGNSS DGNSS = Differential GNSS Correction data from Differential GNSS is taken into account

GNSS_FIX_TYPE_RTK_FIXED RTK = Real Time Kinematic Correction data from a RTK fixed solution is taken into account

GNSS_FIX_TYPE_RTK_FLOAT RTK = Real Time Kinematic Correction data from a RTK floating solution is taken into account

GNSS_FIX_TYPE_SSR SSR = State Space Representation Correction data according the SSR standard from RTCM SC104 or similar are taken into account

GNSS_FIX_TYPE_ESTIMATED The position is propagated without additional sensor input

GNSS_FIX_TYPE_DEAD_RECKONING The position is propagated with support of additional sensor input, e.g. from inertial and/or vehicle sensors

GNSS_FIX_TYPE_MANUAL Position is set by manual input

GNSS_FIX_TYPE_SIMULATOR_MODE Position is simulated

2.4.2.4 enum EGNSSPositionValidityBits

[TGNSSPosition::validityBits](#) provides information about the currently valid signals of the GNSS position and velocity including status and accuracy data. It is a or'ed bitmask of the EGNSSPositionValidityBits values.

Enumerator

GNSS_POSITION_LATITUDE_VALID Validity bit for field [TGNSSPosition::latitude](#).

GNSS_POSITION_LONGITUDE_VALID Validity bit for field [TGNSSPosition::longitude](#).

GNSS_POSITION_ALTITUDEMSL_VALID Validity bit for field [TGNSSPosition::altitudeMSL](#).

GNSS_POSITION_ALTITUDEELL_VALID Validity bit for field [TGNSSPosition::altitudeEll](#).

GNSS_POSITION_HSPEED_VALID Validity bit for field [TGNSSPosition::hSpeed](#).

GNSS_POSITION_VSPEED_VALID Validity bit for field [TGNSSPosition::vSpeed](#).

GNSS_POSITION_HEADING_VALID Validity bit for field [TGNSSPosition::heading](#).

GNSS_POSITION_PDOP_VALID Validity bit for field [TGNSSPosition::pdop](#).

GNSS_POSITION_HDOP_VALID Validity bit for field [TGNSSPosition::hdop](#).

GNSS_POSITION_VDOP_VALID Validity bit for field [TGNSSPosition::vdop](#).

GNSS_POSITION_USAT_VALID Validity bit for field [TGNSSPosition::usedSatellites](#).

GNSS_POSITION_TSAT_VALID Validity bit for field [TGNSSPosition::trackedSatellites](#).

GNSS_POSITION_VSAT_VALID Validity bit for field [TGNSSPosition::visibleSatellites](#).

GNSS_POSITION_SHPOS_VALID Validity bit for field [TGNSSPosition::sigmaHPosition](#).

GNSS_POSITION_SALT_VALID Validity bit for field [TGNSSPosition::sigmaAltitude](#).

GNSS_POSITION_SHSPEED_VALID Validity bit for field [TGNSSPosition::sigmaHSpeed](#).

GNSS_POSITION_SVSPEED_VALID Validity bit for field [TGNSSPosition::sigmaVSpeed](#).

GNSS_POSITION_SHEADING_VALID Validity bit for field [TGNSSPosition::sigmaHeading](#).

GNSS_POSITION_STAT_VALID Validity bit for field [TGNSSPosition::fixStatus](#).

GNSS_POSITION_TYPE_VALID Validity bit for field [TGNSSPosition::fixTypeBits](#).

GNSS_POSITION_ASYS_VALID Validity bit for field [TGNSSPosition::activatedSystems](#).

GNSS_POSITION_USYS_VALID Validity bit for field [TGNSSPosition::usedSystems](#).

2.4.2.5 enum EGNSSSatelliteDetailValidityBits

[TGNSSSatelliteDetail::validityBits](#) provides information about the currently valid values of GNSS satellite data. It is a or'ed bitmask of the EGNSSSatelliteDetailValidityBits values.

Enumerator

- GNSS_SATELLITE_SYSTEM_VALID** Validity bit for field [TGNSSSatelliteDetail::system](#).
- GNSS_SATELLITE_ID_VALID** Validity bit for field [TGNSSSatelliteDetail::satelliteId](#).
- GNSS_SATELLITE_AZIMUTH_VALID** Validity bit for field [TGNSSSatelliteDetail::azimuth](#).
- GNSS_SATELLITE_ELEVATION_VALID** Validity bit for field [TGNSSSatelliteDetail::elevation](#).
- GNSS_SATELLITE_SNR_VALID** Validity bit for field [TGNSSSatelliteDetail::SNR](#).
- GNSS_SATELLITE_USED_VALID** Validity bit for field [TGNSSSatelliteDetail::statusBits::GNSS_SATELLITE_USED](#).
- GNSS_SATELLITE_EPHEMERIS_AVAILABLE_VALID** Validity bit for field [TGNSSSatelliteDetail::statusBits::GNSS_SATELLITE_EPHEMERIS_AVAILABLE](#).
- GNSS_SATELLITE_RESIDUAL_VALID** Validity bit for field [TGNSSSatelliteDetail::posResidual](#).

2.4.2.6 enum EGNSSSatelliteFlag

[TGNSSSatelliteDetail::statusBits](#) provides additional status information about a GNSS satellite. It is a or'ed bitmask of the EGNSSSatelliteFlag values.

Enumerator

- GNSS_SATELLITE_USED** Bit is set when satellite is used for fix.
- GNSS_SATELLITE_EPHEMERIS_AVAILABLE** Bit is set when ephemeris is available for this satellite.

2.4.2.7 enum EGNSSSystem

Enumeration to describe the type of GNSS system to which a particular GNSS satellite belongs. For GNSS systems providing different signals (frequencies), separate values are provided for each signal. The enumeration values can be used in bitmasks to represent combinations of satellite systems, e.g. in case of multiconstellation GNSS or GNSS + augmentation systems

Enumerator

- GNSS_SYSTEM_GPS** GPS (L1 signal)
- GNSS_SYSTEM_GLONASS** GLONASS (L1 signal)
- GNSS_SYSTEM_GALILEO** GALILEO (E1 signal)
- GNSS_SYSTEM_BEIDOU** BeiDou aka COMPASS (B1 signal)
- GNSS_SYSTEM_GPS_L2** GPS (L2 signal)
- GNSS_SYSTEM_GPS_L5** GPS (L5 signal)
- GNSS_SYSTEM_GLONASS_L2** GLONASS (L2 signal)
- GNSS_SYSTEM_BEIDOU_B2** BeiDou aka COMPASS (B2 signal)
- GNSS_SYSTEM_SBAS_WAAS** WAAS (North America)
- GNSS_SYSTEM_SBAS_EGNOS** EGNOS (Europe)
- GNSS_SYSTEM_SBAS_MSAS** MSAS (Japan)
- GNSS_SYSTEM_SBAS_QZSS_SAIF** QZSS-SAIF (Japan)
- GNSS_SYSTEM_SBAS_SDCM** SDCM (Russia)
- GNSS_SYSTEM_SBAS_GAGAN** GAGAN (India)

2.4.2.8 enum EGNSSTimeScale

Description of the time scale used.

Enumerator

GNSS_TIME_SCALE_UTC GNSS time is provided according UTC time scale (with leap seconds). This is the preferred time scale.

GNSS_TIME_SCALE_GPS GNSS time is provided according GPS time scale (no leap seconds since 06--Jan-1980). This time scale will only be used if UTC is not available.

2.4.2.9 enum EGNSSTimeValidityBits

[TGNSSTime::validityBits](#) provides information about the currently valid parts of UTC date/time. It is a or'ed bitmask of the EGNSSUTCValidityBits values. There are separate validity bits for date and time since a GPS receiver may be able to provide time earlier than date.

Enumerator

GNSS_TIME_TIME_VALID Validity bit for field [TGNSSTime](#) fields hour, minute, second, ms.

GNSS_TIME_DATE_VALID Validity bit for field [TGNSSTime](#) fields year, month, day.

GNSS_TIME_SCALE_VALID Validity bit for field [TGNSSTime](#) field scale.

GNSS_TIME_LEAPSEC_VALID Validity bit for field [TGNSSTime](#) field leapSeconds.

2.4.3 Function Documentation

2.4.3.1 bool gnssDeregisterPositionCallback (GNSSPositionCallback callback)

Deregister GNSS position callback. After calling this method no new data will be delivered to the client.

Parameters

<i>callback</i>	The callback which should be deregistered.
-----------------	--

Returns

True if callback has been deregistered successfully.

2.4.3.2 bool gnssDeregisterSatelliteDetailCallback (GNSSSatelliteDetailCallback callback)

Deregister GNSS satellite detail callback. After calling this method no new data will be delivered to the client.

Parameters

<i>callback</i>	The callback which should be deregistered.
-----------------	--

Returns

True if callback has been deregistered successfully.

2.4.3.3 bool gnssDeregisterTimeCallback (GNSSTimeCallback callback)

Deregister GNSS UTC time callback. After calling this method no new time will be delivered to the client.

Parameters

<i>callback</i>	The callback which should be deregistered.
-----------------	--

Returns

True if callback has been deregistered successfully.

2.4.3.4 bool gnssGetConfiguration (TGNSSConfiguration * *gnssConfig*)

Accessing static configuration data related to the GNSS service.

Parameters

<i>gnssConfig</i>	After calling the method the currently available GNSS configuration data is written into gnss-Config.
-------------------	---

Returns

Is true if data can be provided and false otherwise, e.g. missing initialization

2.4.3.5 bool gnssGetPosition (TGNSSPosition * *position*)

Method to get the GNSS position data at a specific point in time. All valid flags are updated. The data is only guaranteed to be updated when the valid flag is true.

Parameters

<i>position</i>	After calling the method current GNSS position, velocity, accuracy are written into this parameter.
-----------------	---

Returns

Is true if data can be provided and false otherwise, e.g. missing initialization

2.4.3.6 bool gnssGetPrecisionTimingOffset (int32_t * *delta*)

Provides the precision timing information as signaled by the GNSS PPS signal. For accurate timing the 1 PPS (pulse per second) signal from the GNSS receiver is used within the positioning framework. The PPS is a hardware signal which is a UTC synchronized pulse. The duration between the pulses is 1s +/- 40ns and the duration of the pulse is configurable (about 100-200ms). The PPS signal can be provided in the positioning framework as an interrupt service routine and this method provides the access to the delta from UTC to system time. If you really need precision timing you have to have the system time set within a range of +/-2s of UTC.

Parameters

<i>delta</i>	The result is provided in this parameter in nanoseconds. It gives the deviation of the system time (+/-) in respect to the PPS pulse and UTC. If the deviation is greater than a value that can be represented with 32 Bits (i.e. more or less than about 2s) the maximum values are written to this parameter and the return value will be false.
--------------	--

Returns

True if the precision timing is available and fits in the range which can be represented by the delta parameter.

2.4.3.7 bool gnssGetSatelliteDetails (TGNSSSatelliteDetail * *satelliteDetails*, uint16_t *count*, uint16_t * *numSatelliteDetails*)

Method to get the GNSS satellite details at a specific point in time. All valid flags are updated. The data is only guaranteed to be updated when the valid flag is true.

Parameters

<i>satelliteDetails</i>	After calling the method current GNSS satellite details are written into this array with size count.
<i>count</i>	Number of elements of the array *satelliteDetails. This should be at least TGnssMetaData::numChannels .
<i>numSatellite-Details</i>	Number of elements written to the array *satelliteDetails.

Returns

Is true if data can be provided and false otherwise, e.g. missing initialization

2.4.3.8 bool gnssGetTime (TGNSSTime * utc)

Method to get the UTC date / time data of the GNSS receiver at a specific point in time. The valid flags is updated. The data is only guaranteed to be updated when the valid flag is true.

Parameters

<i>time</i>	After calling the method the current GNSS UTC date / time is written into this parameter.
-------------	---

Returns

Is true if data can be provided and false otherwise, e.g. missing initialization

2.4.3.9 bool gnssRegisterPositionCallback (GNSSPositionCallback callback)

Register GNSS position callback. The callback will be invoked when new position data data is available from the GNSS receiver. The valid flags is updated. The data is only guaranteed to be updated when the valid flag is true.

Parameters

<i>callback</i>	The callback which should be registered.
-----------------	--

Returns

True if callback has been registered successfully.

2.4.3.10 bool gnssRegisterSatelliteDetailCallback (GNSSSatelliteDetailCallback callback)

Register GNSS satellite detail callback. The callback will be invoked when new date data is available from the GNSS receiver. The valid flags is updated. The data is only guaranteed to be updated when the valid flag is true.

Parameters

<i>callback</i>	The callback which should be registered.
-----------------	--

Returns

True if callback has been registered successfully.

2.4.3.11 bool gnssRegisterTimeCallback (GNSSTimeCallback callback)

Register GNSS UTC time callback. The callback will be invoked when new time data is available from the GNSS receiver. The valid flags is updated. The data is only guaranteed to be updated when the valid flag is true.

Parameters

<i>callback</i>	The callback which should be registered.
-----------------	--

Returns

True if callback has been registered successfully.

2.4.3.12 bool gnssSetGNSSSystems (uint32_t *activateSystems*)

Send a configuration request to use a specific set of GNSS satellite systems

No immediate confirmation is provided as the configuration request is typically executed asynchronously by the GNSS receiver. To verify when the configuration change has been executed, the corresponding fields `activated_systems` and `used_systems` in [TGNSSPosition](#) updates have to be monitored

Parameters

<i>activateSystems</i>	Bit mask indicating the satellite systems which shall be activated for use [bitwise or'ed EGN-SSSystem values].
------------------------	---

Returns

True if the configuration request has been accepted.

False if the configuration request has not been accepted or is not supported at all.

Index

- activatedSystems
 - TGNSSPosition, [5](#)
- altitudeEll
 - TGNSSPosition, [5](#)
- altitudeMSL
 - TGNSSPosition, [5](#)
- antStatus
 - TGNSSStatus, [8](#)
- antennaPosition
 - TGNSSConfiguration, [2](#)
- azimuth
 - TGNSSSatelliteDetail, [7](#)
- category
 - TGnssMetaData, [3](#)
- cycleTime
 - TGnssMetaData, [3](#)
- day
 - TGNSSTime, [9](#)
- EGNSSAntennaStatus
 - gnss-status.h, [12](#)
- EGNSSConfigValidityBits
 - gnss.h, [17](#)
- EGNSSFixStatus
 - gnss.h, [17](#)
- EGNSSFixType
 - gnss.h, [17](#)
- EGNSSPositionValidityBits
 - gnss.h, [18](#)
- EGNSSSatelliteDetailValidityBits
 - gnss.h, [18](#)
- EGNSSSatelliteFlag
 - gnss.h, [19](#)
- EGNSSStatus
 - gnss-status.h, [12](#)
- EGNSSStatusValidityBits
 - gnss-status.h, [13](#)
- EGNSSSystem
 - gnss.h, [19](#)
- EGNSSTimeScale
 - gnss.h, [19](#)
- EGNSSTimeValidityBits
 - gnss.h, [20](#)
- EGnssCategory
 - gnss-meta-data.h, [11](#)
- EGnssTypeBits
 - gnss-meta-data.h, [11](#)
- elevation
 - TGNSSSatelliteDetail, [7](#)
- fixStatus
 - TGNSSPosition, [5](#)
- fixTypeBits
 - TGNSSPosition, [5](#)
- GNSS_ANT_STATUS_NORMAL
 - gnss-status.h, [12](#)
- GNSS_ANT_STATUS_OPEN
 - gnss-status.h, [12](#)
- GNSS_ANT_STATUS_OUTOFSERVICE
 - gnss-status.h, [12](#)
- GNSS_ANT_STATUS_OVERCURRENT
 - gnss-status.h, [12](#)
- GNSS_ANT_STATUS_SHORT_BATT
 - gnss-status.h, [12](#)
- GNSS_ANT_STATUS_SHORT_GND
 - gnss-status.h, [12](#)
- GNSS_CATEGORY_LOGICAL
 - gnss-meta-data.h, [11](#)
- GNSS_CATEGORY_PHYSICAL
 - gnss-meta-data.h, [11](#)
- GNSS_CATEGORY_UNKNOWN
 - gnss-meta-data.h, [11](#)
- GNSS_CONFIG_ANTPOS_VALID
 - gnss.h, [17](#)
- GNSS_CONFIG_SATSYS_VALID
 - gnss.h, [17](#)
- GNSS_FIX_STATUS_2D
 - gnss.h, [17](#)
- GNSS_FIX_STATUS_3D
 - gnss.h, [17](#)
- GNSS_FIX_STATUS_NO
 - gnss.h, [17](#)
- GNSS_FIX_STATUS_TIME
 - gnss.h, [17](#)
- GNSS_FIX_TYPE_DEAD_RECKONING
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_DGNSS
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_ESTIMATED
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_INTEGRITY_CHECKED
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_MANUAL
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_MULTI_CONSTELLATION
 - gnss.h, [17](#)
- GNSS_FIX_TYPE_MULTI_FREQUENCY
 - gnss.h, [17](#)
- GNSS_FIX_TYPE_PPP
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_RTK_FIXED
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_RTK_FLOAT
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_SBAS
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_SIMULATOR_MODE
 - gnss.h, [18](#)
- GNSS_FIX_TYPE_SINGLE_FREQUENCY

gnss.h, [17](#)
GNSS_FIX_TYPE_SSR
gnss.h, [18](#)
GNSS_POSITION_ALTITUDEELL_VALID
gnss.h, [18](#)
GNSS_POSITION_ALTITUDEMSL_VALID
gnss.h, [18](#)
GNSS_POSITION_ASYS_VALID
gnss.h, [18](#)
GNSS_POSITION_HDOP_VALID
gnss.h, [18](#)
GNSS_POSITION_HEADING_VALID
gnss.h, [18](#)
GNSS_POSITION_HSPEED_VALID
gnss.h, [18](#)
GNSS_POSITION_LATITUDE_VALID
gnss.h, [18](#)
GNSS_POSITION_LONGITUDE_VALID
gnss.h, [18](#)
GNSS_POSITION_PDOP_VALID
gnss.h, [18](#)
GNSS_POSITION_SALT_VALID
gnss.h, [18](#)
GNSS_POSITION_SHEADING_VALID
gnss.h, [18](#)
GNSS_POSITION_SHPOS_VALID
gnss.h, [18](#)
GNSS_POSITION_SHSPEED_VALID
gnss.h, [18](#)
GNSS_POSITION_STAT_VALID
gnss.h, [18](#)
GNSS_POSITION_SVSPEED_VALID
gnss.h, [18](#)
GNSS_POSITION_TSAT_VALID
gnss.h, [18](#)
GNSS_POSITION_TYPE_VALID
gnss.h, [18](#)
GNSS_POSITION_USAT_VALID
gnss.h, [18](#)
GNSS_POSITION_USYS_VALID
gnss.h, [18](#)
GNSS_POSITION_VDOP_VALID
gnss.h, [18](#)
GNSS_POSITION_VSAT_VALID
gnss.h, [18](#)
GNSS_POSITION_VSPEED_VALID
gnss.h, [18](#)
GNSS_SATELLITE_AZIMUTH_VALID
gnss.h, [19](#)
GNSS_SATELLITE_ELEVATION_VALID
gnss.h, [19](#)
GNSS_SATELLITE_EPHEMERIS_AVAILABLE
gnss.h, [19](#)
GNSS_SATELLITE_EPHEMERIS_AVAILABLE_VALID
gnss.h, [19](#)
GNSS_SATELLITE_ID_VALID
gnss.h, [19](#)
GNSS_SATELLITE_RESIDUAL_VALID
gnss.h, [19](#)
GNSS_SATELLITE_SNR_VALID
gnss.h, [19](#)
GNSS_SATELLITE_SYSTEM_VALID
gnss.h, [19](#)
GNSS_SATELLITE_USED
gnss.h, [19](#)
GNSS_SATELLITE_USED_VALID
gnss.h, [19](#)
GNSS_STATUS_ANT_STATUS_VALID
gnss-status.h, [13](#)
GNSS_STATUS_AVAILABLE
gnss-status.h, [13](#)
GNSS_STATUS_FAILURE
gnss-status.h, [13](#)
GNSS_STATUS_INITIALIZING
gnss-status.h, [13](#)
GNSS_STATUS_NOTAVAILABLE
gnss-status.h, [13](#)
GNSS_STATUS_OUTOFSERVICE
gnss-status.h, [13](#)
GNSS_STATUS_RESTARTING
gnss-status.h, [13](#)
GNSS_STATUS_STATUS_VALID
gnss-status.h, [13](#)
GNSS_SYSTEM_BEIDOU
gnss.h, [19](#)
GNSS_SYSTEM_BEIDOU_B2
gnss.h, [19](#)
GNSS_SYSTEM_GALILEO
gnss.h, [19](#)
GNSS_SYSTEM_GLONASS
gnss.h, [19](#)
GNSS_SYSTEM_GLONASS_L2
gnss.h, [19](#)
GNSS_SYSTEM_GPS
gnss.h, [19](#)
GNSS_SYSTEM_GPS_L2
gnss.h, [19](#)
GNSS_SYSTEM_GPS_L5
gnss.h, [19](#)
GNSS_SYSTEM_SBAS_EGNOS
gnss.h, [19](#)
GNSS_SYSTEM_SBAS_GAGAN
gnss.h, [19](#)
GNSS_SYSTEM_SBAS_MSAS
gnss.h, [19](#)
GNSS_SYSTEM_SBAS_QZSS_SAIF
gnss.h, [19](#)
GNSS_SYSTEM_SBAS_SDCM
gnss.h, [19](#)
GNSS_SYSTEM_SBAS_WAAS
gnss.h, [19](#)
GNSS_TIME_DATE_VALID
gnss.h, [20](#)
GNSS_TIME_LEAPSEC_VALID
gnss.h, [20](#)
GNSS_TIME_SCALE_GPS

- gnss.h, 20
- GNSS_TIME_SCALE_UTC
 - gnss.h, 20
- GNSS_TIME_SCALE_VALID
 - gnss.h, 20
- GNSS_TIME_TIME_VALID
 - gnss.h, 20
- GNSS_TYPE_ASSISTED
 - gnss-meta-data.h, 11
- GNSS_TYPE_DGPS
 - gnss-meta-data.h, 11
- GNSS_TYPE_DR
 - gnss-meta-data.h, 11
- GNSS_TYPE_GNSS
 - gnss-meta-data.h, 11
- GNSS_TYPE_SBAS
 - gnss-meta-data.h, 11
- GNSSPositionCallback
 - gnss.h, 15
- GNSSSatelliteDetailCallback
 - gnss.h, 15
- GNSSStatusCallback
 - gnss-status.h, 12
- GNSSTimeCallback
 - gnss.h, 17
- gnss-init.h, 9
 - gnssDestroy, 10
 - gnssGetVersion, 10
 - gnssInit, 10
- gnss-meta-data.h
 - GNSS_CATEGORY_LOGICAL, 11
 - GNSS_CATEGORY_PHYSICAL, 11
 - GNSS_CATEGORY_UNKNOWN, 11
 - GNSS_TYPE_ASSISTED, 11
 - GNSS_TYPE_DGPS, 11
 - GNSS_TYPE_DR, 11
 - GNSS_TYPE_GNSS, 11
 - GNSS_TYPE_SBAS, 11
- gnss-meta-data.h, 10
 - EGnssCategory, 11
 - EGnssTypeBits, 11
 - gnssGetMetaData, 11
- gnss-status.h
 - GNSS_ANT_STATUS_NORMAL, 12
 - GNSS_ANT_STATUS_OPEN, 12
 - GNSS_ANT_STATUS_OUTOFSERVICE, 12
 - GNSS_ANT_STATUS_OVERCURRENT, 12
 - GNSS_ANT_STATUS_SHORT_BATT, 12
 - GNSS_ANT_STATUS_SHORT_GND, 12
 - GNSS_STATUS_ANT_STATUS_VALID, 13
 - GNSS_STATUS_AVAILABLE, 13
 - GNSS_STATUS_FAILURE, 13
 - GNSS_STATUS_INITIALIZING, 13
 - GNSS_STATUS_NOTAVAILABLE, 13
 - GNSS_STATUS_OUTOFSERVICE, 13
 - GNSS_STATUS_RESTARTING, 13
 - GNSS_STATUS_STATUS_VALID, 13
- gnss-status.h, 12
- EGNSSAntennaStatus, 12
- EGNSSStatus, 12
- EGNSSStatusValidityBits, 13
- GNSSStatusCallback, 12
- gnssDeregisterStatusCallback, 13
- gnssGetStatus, 13
- gnssRegisterStatusCallback, 13
- gnss.h
 - GNSS_CONFIG_ANTPOS_VALID, 17
 - GNSS_CONFIG_SATSYS_VALID, 17
 - GNSS_FIX_STATUS_2D, 17
 - GNSS_FIX_STATUS_3D, 17
 - GNSS_FIX_STATUS_NO, 17
 - GNSS_FIX_STATUS_TIME, 17
 - GNSS_FIX_TYPE_DEAD_RECKONING, 18
 - GNSS_FIX_TYPE_DGNSS, 18
 - GNSS_FIX_TYPE_ESTIMATED, 18
 - GNSS_FIX_TYPE_INTEGRITY_CHECKED, 18
 - GNSS_FIX_TYPE_MANUAL, 18
 - GNSS_FIX_TYPE_MULTI_CONSTELLATION, 17
 - GNSS_FIX_TYPE_MULTI_FREQUENCY, 17
 - GNSS_FIX_TYPE_PPP, 18
 - GNSS_FIX_TYPE_RTK_FIXED, 18
 - GNSS_FIX_TYPE_RTK_FLOAT, 18
 - GNSS_FIX_TYPE_SBAS, 18
 - GNSS_FIX_TYPE_SIMULATOR_MODE, 18
 - GNSS_FIX_TYPE_SINGLE_FREQUENCY, 17
 - GNSS_FIX_TYPE_SSR, 18
 - GNSS_POSITION_ALTITUDEELL_VALID, 18
 - GNSS_POSITION_ALTITUDEMSL_VALID, 18
 - GNSS_POSITION_ASYS_VALID, 18
 - GNSS_POSITION_HDOP_VALID, 18
 - GNSS_POSITION_HEADING_VALID, 18
 - GNSS_POSITION_HSPEED_VALID, 18
 - GNSS_POSITION_LATITUDE_VALID, 18
 - GNSS_POSITION_LONGITUDE_VALID, 18
 - GNSS_POSITION_PDOP_VALID, 18
 - GNSS_POSITION_SALT_VALID, 18
 - GNSS_POSITION_SHEADING_VALID, 18
 - GNSS_POSITION_SHPOS_VALID, 18
 - GNSS_POSITION_SHSPEED_VALID, 18
 - GNSS_POSITION_STAT_VALID, 18
 - GNSS_POSITION_SVSPEED_VALID, 18
 - GNSS_POSITION_TSAT_VALID, 18
 - GNSS_POSITION_TYPE_VALID, 18
 - GNSS_POSITION_USAT_VALID, 18
 - GNSS_POSITION_USYS_VALID, 18
 - GNSS_POSITION_VDOP_VALID, 18
 - GNSS_POSITION_VSAT_VALID, 18
 - GNSS_POSITION_VSPEED_VALID, 18
 - GNSS_SATELLITE_AZIMUTH_VALID, 19
 - GNSS_SATELLITE_ELEVATION_VALID, 19
 - GNSS_SATELLITE_EPHEMERIS_AVAILABLE, 19
 - GNSS_SATELLITE_EPHEMERIS_AVAILABLE - VALID, 19
 - GNSS_SATELLITE_ID_VALID, 19
 - GNSS_SATELLITE_RESIDUAL_VALID, 19

- GNSS_SATELLITE_SNR_VALID, 19
- GNSS_SATELLITE_SYSTEM_VALID, 19
- GNSS_SATELLITE_USED, 19
- GNSS_SATELLITE_USED_VALID, 19
- GNSS_SYSTEM_BEIDOU, 19
- GNSS_SYSTEM_BEIDOU_B2, 19
- GNSS_SYSTEM_GALILEO, 19
- GNSS_SYSTEM_GLONASS, 19
- GNSS_SYSTEM_GLONASS_L2, 19
- GNSS_SYSTEM_GPS, 19
- GNSS_SYSTEM_GPS_L2, 19
- GNSS_SYSTEM_GPS_L5, 19
- GNSS_SYSTEM_SBAS_EGNOS, 19
- GNSS_SYSTEM_SBAS_GAGAN, 19
- GNSS_SYSTEM_SBAS_MSAS, 19
- GNSS_SYSTEM_SBAS_QZSS_SAIF, 19
- GNSS_SYSTEM_SBAS_SDCM, 19
- GNSS_SYSTEM_SBAS_WAAS, 19
- GNSS_TIME_DATE_VALID, 20
- GNSS_TIME_LEAPSEC_VALID, 20
- GNSS_TIME_SCALE_GPS, 20
- GNSS_TIME_SCALE_UTC, 20
- GNSS_TIME_SCALE_VALID, 20
- GNSS_TIME_TIME_VALID, 20
- gnss.h, 14
 - EGNSSConfigValidityBits, 17
 - EGNSSFixStatus, 17
 - EGNSSFixType, 17
 - EGNSSPositionValidityBits, 18
 - EGNSSSatelliteDetailValidityBits, 18
 - EGNSSSatelliteFlag, 19
 - EGNSSSystem, 19
 - EGNSSTimeScale, 19
 - EGNSSTimeValidityBits, 20
 - GNSSPositionCallback, 15
 - GNSSSatelliteDetailCallback, 15
 - GNSSTimeCallback, 17
 - gnssDeregisterPositionCallback, 20
 - gnssDeregisterSatelliteDetailCallback, 20
 - gnssDeregisterTimeCallback, 20
 - gnssGetConfiguration, 21
 - gnssGetPosition, 21
 - gnssGetPrecisionTimingOffset, 21
 - gnssGetSatelliteDetails, 21
 - gnssGetTime, 22
 - gnssRegisterPositionCallback, 22
 - gnssRegisterSatelliteDetailCallback, 22
 - gnssRegisterTimeCallback, 22
 - gnssSetGNSSSystems, 23
- gnssDeregisterPositionCallback
 - gnss.h, 20
- gnssDeregisterSatelliteDetailCallback
 - gnss.h, 20
- gnssDeregisterStatusCallback
 - gnss-status.h, 13
- gnssDeregisterTimeCallback
 - gnss.h, 20
- gnssDestroy
 - gnss-init.h, 10
- gnssGetConfiguration
 - gnss.h, 21
- gnssGetMetaData
 - gnss-meta-data.h, 11
- gnssGetPosition
 - gnss.h, 21
- gnssGetPrecisionTimingOffset
 - gnss.h, 21
- gnssGetSatelliteDetails
 - gnss.h, 21
- gnssGetStatus
 - gnss-status.h, 13
- gnssGetTime
 - gnss.h, 22
- gnssGetVersion
 - gnss-init.h, 10
- gnssInit
 - gnss-init.h, 10
- gnssRegisterPositionCallback
 - gnss.h, 22
- gnssRegisterSatelliteDetailCallback
 - gnss.h, 22
- gnssRegisterStatusCallback
 - gnss-status.h, 13
- gnssRegisterTimeCallback
 - gnss.h, 22
- gnssSetGNSSSystems
 - gnss.h, 23
- hSpeed
 - TGNSSPosition, 5
- hdop
 - TGNSSPosition, 5
- heading
 - TGNSSPosition, 5
- hour
 - TGNSSTime, 9
- latitude
 - TGNSSPosition, 5
- leapSeconds
 - TGNSSTime, 9
- longitude
 - TGNSSPosition, 5
- minute
 - TGNSSTime, 9
- month
 - TGNSSTime, 9
- ms
 - TGNSSTime, 9
- numChannels
 - TGnssMetaData, 3
- pdop
 - TGNSSPosition, 5
- posResidual

- TGNSSSatelliteDetail, 7
- SNR
 - TGNSSSatelliteDetail, 7
- satelliteId
 - TGNSSSatelliteDetail, 7
- scale
 - TGNSSTime, 9
- second
 - TGNSSTime, 9
- sigmaAltitude
 - TGNSSPosition, 5
- sigmaHPosition
 - TGNSSPosition, 5
- sigmaHSpeed
 - TGNSSPosition, 5
- sigmaHeading
 - TGNSSPosition, 5
- sigmaVSpeed
 - TGNSSPosition, 5
- status
 - TGNSSStatus, 8
- statusBits
 - TGNSSSatelliteDetail, 7
- supportedSystems
 - TGNSSConfiguration, 2
- system
 - TGNSSSatelliteDetail, 7
- TGNSSConfiguration, 2
 - antennaPosition, 2
 - supportedSystems, 2
 - validityBits, 2
- TGNSSDistance3D, 2
 - x, 3
 - y, 3
 - z, 3
- TGNSSPosition, 4
 - activatedSystems, 5
 - altitudeEll, 5
 - altitudeMSL, 5
 - fixStatus, 5
 - fixTypeBits, 5
 - hSpeed, 5
 - hdop, 5
 - heading, 5
 - latitude, 5
 - longitude, 5
 - pdop, 5
 - sigmaAltitude, 5
 - sigmaHPosition, 5
 - sigmaHSpeed, 5
 - sigmaHeading, 5
 - sigmaVSpeed, 5
 - timestamp, 6
 - trackedSatellites, 6
 - usedSatellites, 6
 - usedSystems, 6
 - vSpeed, 6
- validityBits, 6
 - vdop, 6
 - visibleSatellites, 6
- TGNSSSatelliteDetail, 6
 - azimuth, 7
 - elevation, 7
 - posResidual, 7
 - SNR, 7
 - satelliteId, 7
 - statusBits, 7
 - system, 7
 - timestamp, 7
 - validityBits, 7
- TGNSSStatus, 7
 - antStatus, 8
 - status, 8
 - timestamp, 8
 - validityBits, 8
- TGNSSTime, 8
 - day, 9
 - hour, 9
 - leapSeconds, 9
 - minute, 9
 - month, 9
 - ms, 9
 - scale, 9
 - second, 9
 - timestamp, 9
 - validityBits, 9
 - year, 9
- TGnssMetaData, 3
 - category, 3
 - cycleTime, 3
 - numChannels, 3
 - typeBits, 4
 - version, 4
- timestamp
 - TGNSSPosition, 6
 - TGNSSSatelliteDetail, 7
 - TGNSSStatus, 8
 - TGNSSTime, 9
- trackedSatellites
 - TGNSSPosition, 6
- typeBits
 - TGnssMetaData, 4
- usedSatellites
 - TGNSSPosition, 6
- usedSystems
 - TGNSSPosition, 6
- vSpeed
 - TGNSSPosition, 6
- validityBits
 - TGNSSConfiguration, 2
 - TGNSSPosition, 6
 - TGNSSSatelliteDetail, 7
 - TGNSSStatus, 8
 - TGNSSTime, 9

vdop
 TGNSSPosition, [6](#)
version
 TGnssMetaData, [4](#)
visibleSatellites
 TGNSSPosition, [6](#)

x
 TGNSSDistance3D, [3](#)

y
 TGNSSDistance3D, [3](#)
year
 TGNSSTime, [9](#)

z
 TGNSSDistance3D, [3](#)