GENIVI GNSSService

Component Specification

Release 3.0.0-alpha
Status: Draft
24.04.2014

Accepted for release by:
This document is a draft of the GNSSService API 3.0.0 defined by the GENIVI expert group Location Based Services (LBS).

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

Keywords:
GNSSService, GNSS, GPS, Positioning API.

Table of Contents

Change History

Version	Date	Author	Change
0.1	27.08.2013	MResidori	Document Created
0.2	18.11.2013	MResidori	Document generated from the GENIVI Enterprise Architect model
0.3	27.03.2014	MResidori	Added copyright notes
3.0.0-alpha	24.04.2014	MResidori	Changed license version from 3.0 to 4.0

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

Term	***Description***
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
tags
Originator = Thomas Himbacher (BMW)
Priority = P1
Rationale =
Sensor Signal Timestamp
tags
Originator = Thomas Bader (BMW)
Priority = P1
Rationale =
Signal Measurement Units
tags
Originator = Thomas Bader (BMW)
Priority = P1
Rationale =
Car Configuration Data
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Signal Values Type Compatibility
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Access to Sensor Services
tags
Originator = Thomas Himbacher (BMW)
Priority = P2
Rationale =
Sensor Directory
tags
Originator = Thomas Himbacher (BMW)
Priority = P2
Rationale =
Support of different Global Navigation Satellite Systems (GNSS) to calculate the current position.
tags
Originator = Philippe Colliot (PSA)
Priority = P2
Rationale = <memo>
SW-POS-GNSS
GNSS Service
tags
Originator = Thomas Himbacher (BMW)
Priority = P1
Rationale =
Extended GNSS Service
tags
Originator = Thomas Himbacher (BMW)
Priority = P2
Rationale =
PPS Signal
tags
Originator = Carsten Isert (BMW)
Priority = P2
Rationale =
AGPS Support
tags
Originator = Thomas Bader (BMW)
Priority = P3
Rationale =
SW-POS-SNS
VehicleSpeed Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Odometer Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Wheel Tick/Speed Sensor Service
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Simple Gyroscope Sensor Service
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Extended Gyroscope Sensor Service
tags
Originator = Thomas Bader (BMW)
Priority = P3
Rationale =
ReverseGear Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Accelerator Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Extended Acceleration Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P3
Rationale =
Inclination Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P3
Rationale =
Slip Angle Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P3
Rationale =
Vehicle State Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P3
Rationale =
SteeringAngle Sensor
tags
Originator = Thomas Bader (BMW)
Priority = P3
Rationale =
SW-POS-ENP
Enhanced Position
tags
Originator = Thomas Himbacher (BMW)
Priority = P1
Rationale =
Provides Data via IPC
tags
Originator = Thomas Bader (BMW)
Priority = P1
Rationale =
Forward a Set of Sensor Values
tags
Originator = Thomas Bader (BMW)
Priority = P2
Rationale =
Data Latency for GNSS and DR Signals
tags
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.
- 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 m^2.
- 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.
- 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:
- 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.

Accuracy:
- fixStatus
- hdop, pdop, vdop
- numberOfSatellites
- sigmaLatitude, sigmaLongitude, sigmaAltitude

Satellite Details:
- Information per satellite: azimuth, elevation, inUse, SatelliteId, signalNoiseRatio

Course Details:
- speed for 3-axis

Antenna:
- Antenna Position in 3D coordinates in relation to the reference point (see reference system).

Updated at least with 1Hz frequency additionally to the Signals provided by GNSS-Only Service.

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 isconfigurable (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 checks 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
- 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.

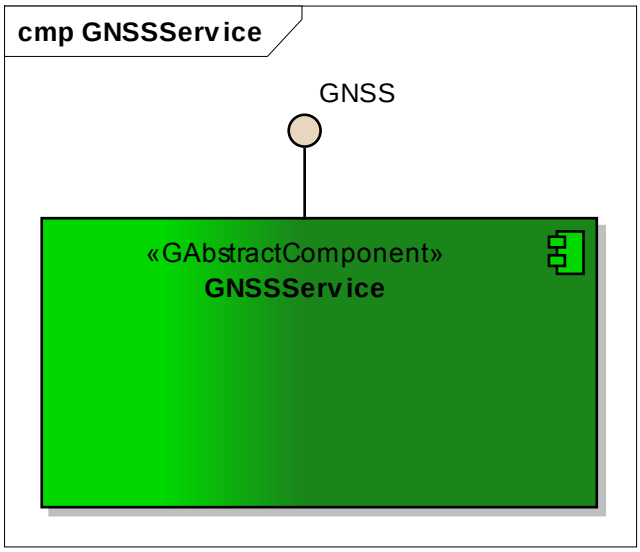

Figure: 2

3. Traceability Diagram

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

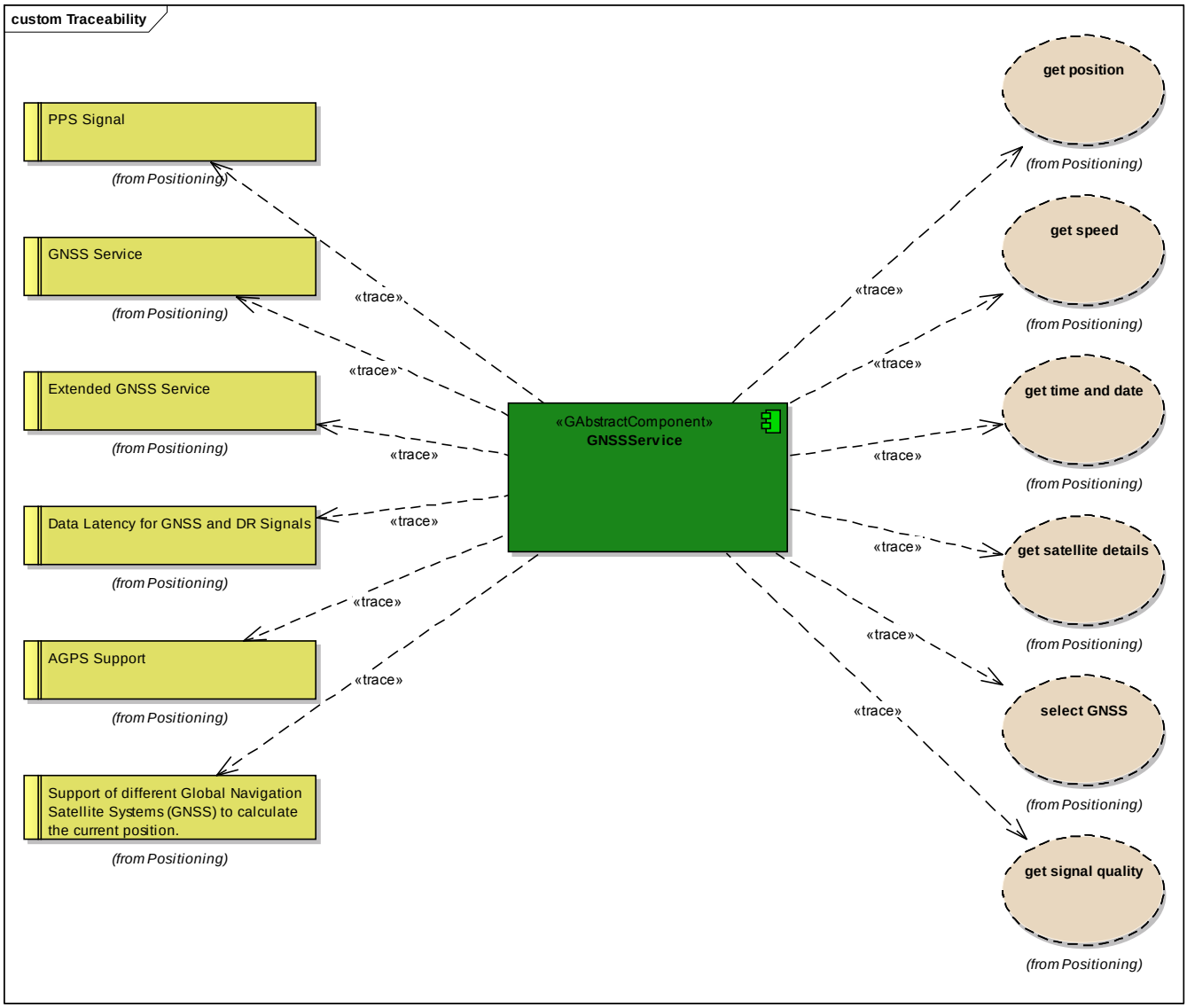

Figure: 3

4. Context Diagram

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

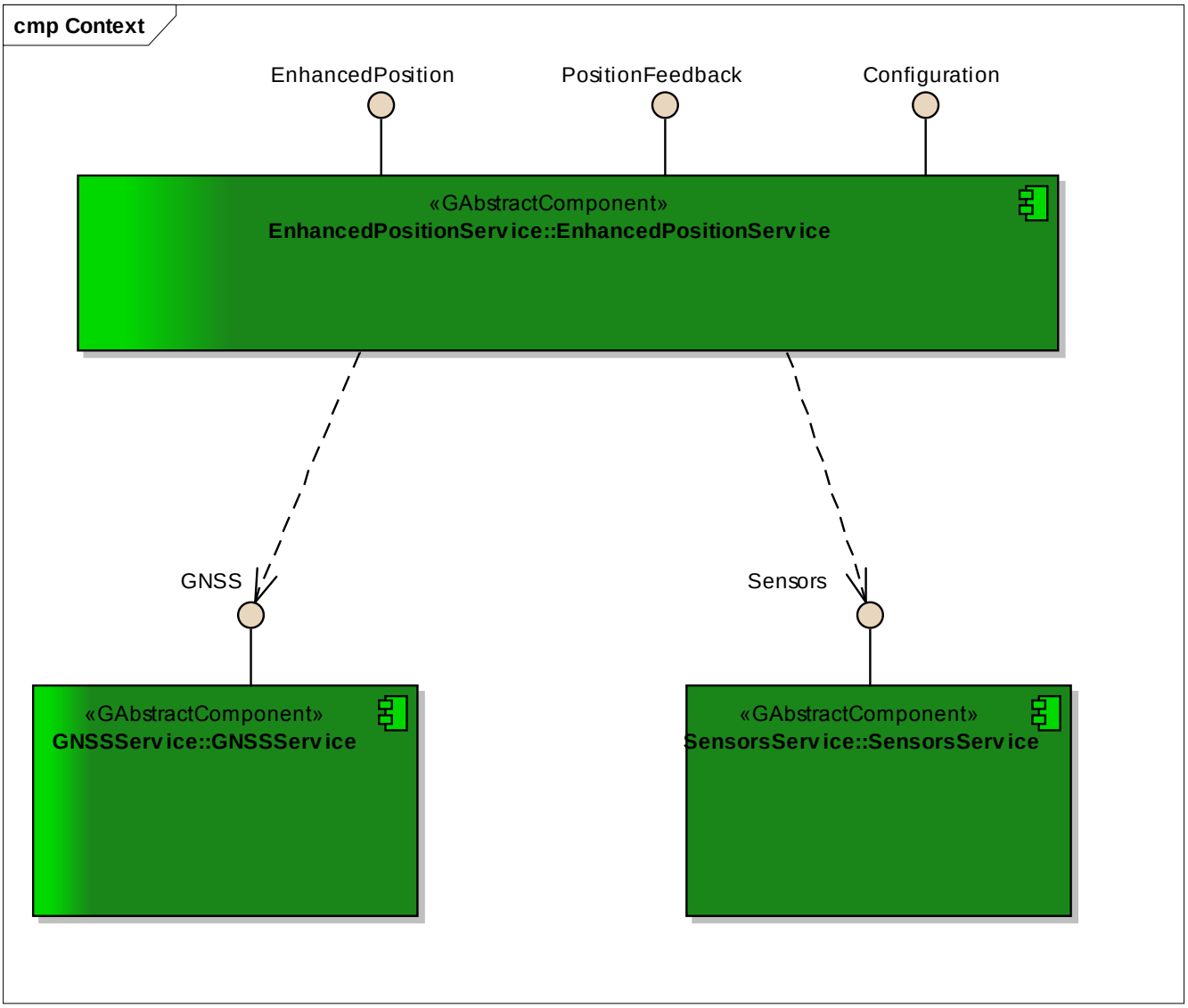

Figure: 4

GNSSService

Generated by Doxygen 1.7.6.1

Thu Apr 24 2014 17:22:00

Contents

1 Class Documentation

1.1 TGNSSAccuracy Struct Reference

`#include <gnss-ext.h>`

Public Attributes

- uint64_t timestamp
- float pdop
- float hdop
- float vdop
- uint16_t usedSatellites
- uint16_t trackedSatellites
- uint16_t visibleSatellites
- float sigmaLatitude
- float sigmaLongitude
- float sigmaAltitude
- EGNSSFixStatus fixStatus
- uint32_t fixTypeBits
- uint32_t validityBits

1.1.1 Detailed Description

Accuracy and status information about the GNSS position. This data structure provides accuracy information. Either the directly from most GNSS receivers available DOP information or if available the error expressed in sigmas for the different axis.

1.1.2 Member Data Documentation

1.1.2.1 EGNSSFixStatus TGNSSAccuracy::fixStatus

Value representing the GNSS mode.

1.1.2.2 uint32_t TGNSSAccuracy::fixTypeBits

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

1.1.2.3 float TGNSSAccuracy::hdop

The horizontal (2D) dilution of precision.

1.1.2.4 float TGNSSAccuracy::pdop

The positional (3D) dilution of precision.

1.1.2.5 float TGNSSAccuracy::sigmaAltitude

Standard error estimate of altitude in [m].

1.1.2.6 float TGNSSAccuracy::sigmaLatitude

Standard error estimate of latitude in [m].

1.1.2.7 float TGNSSAccuracy::sigmaLongitude

Standard error estimate of longitude in [m].

1.1.2.8 uint64_t TGNSSAccuracy::timestamp

Timestamp of the acquisition of the accuracy data.

1.1.2.9 uint16_t TGNSSAccuracy::trackedSatellites

Number of tracked satellites.

1.1.2.10 uint16_t TGNSSAccuracy::usedSatellites

Number of used satellites.

1.1.2.11 uint32_t TGNSSAccuracy::validityBits

Bit mask indicating the validity of each corresponding value. [bitwise or'ed EGNSS-AccuracyValidityBits values]. Must be checked before usage.

1.1.2.12 float TGNSSAccuracy::vdop

The vertical (altitude) dilution of precision.

1.1.2.13 uint16_t TGNSSAccuracy::visibleSatellites

Number of visible satellites.

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

- gnss-ext.h

1.2 TGNSSCourse Struct Reference

`#include <gnss-simple.h>`

Public Attributes

- uint64_t timestamp
- float speed
- float climb
- float heading
- uint32_t validityBits

1.2.1 Detailed Description

GNSS course provides information about the currently course of the receiver. There is an extended service providing the speed for each axis seperately in GNSS-Extended.

1.2.2 Member Data Documentation

1.2.2.1 float TGNSSCourse::climb

Incline / decline in degrees [degree].

1.2.2.2 float TGNSSCourse::heading

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

1.2.2.3 float TGNSSCourse::speed

Speed measured by the GPS receiver [m/s].

1.2.2.4 uint64_t TGNSSCourse::timestamp

Timestamp of the acquisition of the GNSS course signal in [ms]. To enable an accurate DR filtering a defined clock has to be used.

1.2.2.5 uint32_t TGNSSCourse::validityBits

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

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

- gnss-simple.h

1.3 TGNSSCourse3D Struct Reference

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

Public Attributes

- uint64_t timestamp
- float speedLatitude
- float speedLongitude
- float speedAltitude
- uint32_t validityBits

1.3.1 Detailed Description

Course 3D is an extension to the normal course information provided by the GNSS service. The course information is given for each axis seperately.

1.3.2 Member Data Documentation

1.3.2.1 float TGNSSCourse3D::speedAltitude

Speed in direction of altitude in [m/s].

1.3.2.2 float TGNSSCourse3D::speedLatitude

Speed in direction of latitude in [m/s].

1.3.2.3 float TGNSSCourse3D::speedLongitude

Speed in direction of longitude in [m/s].

1.3.2.4 uint64_t TGNSSCourse3D::timestamp

Timestamp of the acquisition of the accuracy data.

1.3.2.5 uint32_t TGNSSCourse3D::validityBits

Bit mask indicating the validity of each corresponding value. [bitwise or'ed EGNSSCourse3DValidityBits values]. Must be checked before usage.

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

- gnss-ext.h

1.4 TGNSSDistance3D Struct Reference

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

Public Attributes

- float x
- float y
- float z

1.4.1 Detailed Description

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

1.4.2 Member Data Documentation

1.4.2.1 float TGNSSDistance3D::x

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

1.4.2.2 float TGNSSDistance3D::y

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

1.4.2.3 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-ext.h

1.5 TGnssMetaData Struct Reference

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

Public Attributes

- uint32_t version
- EGnssCategory category
- uint32_t typeBits
- uint32_t cycleTime
- uint16_t numChannels

1.5.1 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.

1.5.2 Member Data Documentation

1.5.2.1 EGnssCategory TGnssMetaData::category

GNSS Category (Physical/Logical).

1.5.2.2 uint32_t TGnssMetaData::cycleTime

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

1.5.2.3 uint16_t TGnssMetaData::numChannels

Number of GNSS receiver channels for satellite signal reception.

1.5.2.4 uint32_t TGnssMetaData::typeBits

GNSS Type: combination of bits defined in EGnssTypeBits.

1.5.2.5 uint32_t TGnssMetaData::version

Version of the GNSS service.

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

- gnss-meta-data.h

1.6 TGNSSPosition Struct Reference

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

Public Attributes

- uint64_t timestamp
- double latitude

- double longitude
- float altitude
- uint32_t validityBits

1.6.1 Detailed Description

Main position information.

1.6.2 Member Data Documentation

1.6.2.1 float TGNSSPosition::altitude

Altitude in [m]. See TGNSSSimpleConfiguration.typeOfAltitude for reference level (ellipsoid or MSL).

1.6.2.2 double TGNSSPosition::latitude

Latitude in WGS84 in degrees.

1.6.2.3 double TGNSSPosition::longitude

Longitude in WGS84 in degrees.

1.6.2.4 uint64_t TGNSSPosition::timestamp

Timestamp of the acquisition of the position data. [ms]

1.6.2.5 uint32_t TGNSSPosition::validityBits

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

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

- gnss-simple.h

1.7 TGNSSSatelliteDetail Struct Reference

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

Public Attributes

- uint64_t timestamp
- EGNSSSystem system
- uint16_t satelliteId
- uint16_t azimuth
- uint16_t elevation
- uint16_t SNR

- uint32_t statusBits
- uint32_t validityBits

1.7.1 Detailed Description

Detailed data from one GNSS satellite.

1.7.2 Member Data Documentation

1.7.2.1 uint16_t TGNSSSatelliteDetail::azimuth

Satellite Azimuth in degrees. Value range 0..359

1.7.2.2 uint16_t TGNSSSatelliteDetail::elevation

Satellite Elevation in degrees. Value range 0..90

1.7.2.3 uint16_t TGNSSSatelliteDetail::satelliteId

Satellite ID. See also https://collab.genivi.org/issues/browse/G-T-2299 Numbering scheme as defined by NMEA-0183 (v3.01 or later) for the GSV sentence 1..32: GPS satellites (by PRN) 33..64: SBAS/WAAS satellites 65..96: GLONASS satellites Note: Later NMEA-0183 versions probably already have Galileo support

1.7.2.4 uint16_t TGNSSSatelliteDetail::SNR

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

1.7.2.5 uint32_t TGNSSSatelliteDetail::statusBits

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

1.7.2.6 EGNSSSystem TGNSSSatelliteDetail::system

Value representing the GNSS system.

1.7.2.7 uint64_t TGNSSSatelliteDetail::timestamp

Timestamp of the acquisition of the satellite detail data.

1.7.2.8 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-ext.h

1.8 TGNSSSimpleConfiguration Struct Reference

`#include <gnss-simple.h>`

Public Attributes

- EGNSSAltitudeType altitudeType
- uint32_t validityBits

1.8.1 Detailed Description

GNSS Simple Configuration Data.

1.8.2 Member Data Documentation

1.8.2.1 EGNSSAltitudeType TGNSSSimpleConfiguration::altitudeType

Reference level for the GNSS altitude.

1.8.2.2 uint32_t TGNSSSimpleConfiguration::validityBits

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

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

- gnss-simple.h

1.9 TUTCDate Struct Reference

`#include <gnss-ext.h>`

Public Attributes

- uint8_t day
- uint8_t month
- uint16_t year
- bool valid

1.9.1 Detailed Description

Provides the UTC (Coordinated Universal Time) date part.

1.9.2 Member Data Documentation

1.9.2.1 uint8_t TUTCDate::day

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

1.9.2.2 uint8_t TUTCDate::month

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

1.9.2.3 bool TUTCDate::valid

Defines the validity of the complete structure. Must be checked before usage.

1.9.2.4 uint16_t TUTCDate::year

Year fraction of the UTC time. Unit: [year] Number equivalent to the year

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

- gnss-ext.h

1.10 TUTCTime Struct Reference

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

Public Attributes

- uint8_t hour
- uint8_t minute
- uint8_t second
- uint16_t ms
- bool valid

1.10.1 Detailed Description

Provides the UTC (Coordinated Universal Time) time part.

1.10.2 Member Data Documentation

1.10.2.1 uint8_t TUTCTime::hour

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

1.10.2.2 uint8_t TUTCTime::minute

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

1.10.2.3 uint16_t TUTCTime::ms

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

1.10.2.4 uint8_t TUTCTime::second

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

1.10.2.5 bool TUTCTime::valid

Defines the validity of the complete structure. Must be checked before usage.

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

- gnss-ext.h

2 File Documentation

2.1 gnss-ext.h File Reference

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

Classes

- struct TGNSSDistance3D
- struct TGNSSAccuracy
- struct TGNSSCourse3D
- struct TUTCTime
- struct TUTCDate
- struct TGNSSSatelliteDetail

Typedefs

- typedef void(∗ GNSSAccuracyCallback)(const TGNSSAccuracy accuracy[], uint16_t numElements)
- typedef void(∗ GNSSCourse3DCallback)(const TGNSSCourse3D course[], uint16_t numElements)
- typedef void(∗ GNSSUTCTimeCallback)(const TUTCTime time[], uint16_t numElements)
- typedef void(∗ GNSSUTCDateCallback)(const TUTCDate date[], const TUTCTime time[], uint16_t numElements)
- typedef void(∗ GNSSSatelliteDetailCallback)(const TGNSSSatelliteDetail satelliteDetail[], uint16_t numElements)

Enumerations

- enum EGNSSAccuracyValidityBits { GNSS_ACCURACY_PDOP_VALID = 0x00000001, GNSS_ACCURACY_HDOP_VALID = 0x00000002, GNSS_ACCURACY_VDOP_VALID = 0x00000004, GNSS_ACCURACY_USAT_VALID = 0x00000008, GNSS_ACCURACY_TSAT_VALID = 0x00000010, GNSS_ACCURACY_VSAT_VALID = 0x00000020, GNSS_ACCURACY_SLAT_VALID = 0x00000040, GNSS_ACCURACY_SLON_VALID = 0x00000080, GNSS_ACCURACY_SALT_VALID = 0x00000100, GNSS_ACCURACY_STAT_VALID = 0x00000200, GNSS_ACCURACY_TYPE_VALID = 0x00000400 }
- enum EGNSSFixStatus { GNSS_FIX_STATUS_NO, GNSS_FIX_STATUS_2D, GNSS_FIX_STATUS_3D, GNSS_FIX_STATUS_TIME }
- 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_ESTIMATED = 0x00100000, GNSS_FIX_TYPE_DEAD_RECKONING = 0x00200000, GNSS_FIX_TYPE_MANUAL = 0x10000000, GNSS_FIX_TYPE_SIMULATOR_MODE = 0x20000000 }
- enum EGNSSCourse3DValidityBits { GNSS_COURSE3D_SPEEDLAT_VALID = 0x00000001, GNSS_COURSE3D_SPEEDLON_VALID = 0x00000002, GNSS_COURSE3D_SPEEDALT_VALID = 0x00000004 }
- enum EGNSSSystem { GNSS_SYSTEM_GPS = 1, GNSS_SYSTEM_GLONASS = 2, GNSS_SYSTEM_GALILEO = 3, GNSS_SYSTEM_COMPASS = 4, × GNSS_SYSTEM_SBAS_WAAS = 101, GNSS_SYSTEM_SBAS_EGNOS = 102, GNSS_SYSTEM_SBAS_MSAS = 103, GNSS_SYSTEM_SBAS_QZSS_SAIF = 104, GNSS_SYSTEM_SBAS_SDCM = 105, GNSS_SYSTEM_SBAS_GAGAN = 106 }
- enum EGNSSSatelliteFlag { GNSS_SATELLITE_USED = 0x00000001, GNSS_SATELLITE_EPHEMERIS_AVAILABLE = 0x00000002 }
- enum EGNSSSatelliteDetailValidityBits { GNSS_SATELLITE_SYSTEM_VALID = 0x00000001, GNSS_SATELLITE_ID_VALID = 0x00000002, GNSS_SATELLITE_AZIMUTH_VALID = 0x00000004, GNSS_SATELLITE_ELEVATION_VALIDLID = 0x00000008, GNSS_SATELLITE_SNR_VALID = 0x00000010, GNSS_SATELLITE_USED_VALID = 0x00000020, GNSS_SATELLITE_EPHEMERIS_AVAILABLE_VALID = 0x00000040 }

Functions

- bool gnssExtendedInit ()
- bool gnssExtendedDestroy ()
- bool gnssExtendedGetMetaData (TGnssMetaData ∗data)
- bool gnssExtendedGetAntennaPosition (TGNSSDistance3D ∗distance)
- bool gnssExtendedGetAccuracy (TGNSSAccuracy ∗accuracy)
- bool gnssExtendedRegisterAccuracyCallback (GNSSAccuracyCallback callback)

- bool gnssExtendedDeregisterAccuracyCallback (GNSSAccuracyCallback callback)
- bool gnssExtendedGet3DCourse (TGNSSCourse3D *course)
- bool gnssExtendedRegister3DCourseCallback (GNSSCourse3DCallback callback)
- bool gnssExtendedDeregister3DCourseCallback (GNSSCourse3DCallback callback)
- bool gnssExtendedGetUTCTime (TUTCTime *time)
- bool gnssExtendedRegisterUTCTimeCallback (GNSSUTCTimeCallback callback)
- bool gnssExtendedDeregisterUTCTimeCallback (GNSSUTCTimeCallback callback)
- bool gnssExtendedGetUTCDate (TUTCDate *date, TUTCTime *time)
- bool gnssExtendedRegisterUTCDateCallback (GNSSUTCDateCallback callback)
- bool gnssExtendedDeregisterUTCDateCallback (GNSSUTCDateCallback callback)
- bool gnssExtendedGetSatelliteDetails (TGNSSSatelliteDetail *satelliteDetails, uint16_t count, uint16_t *numSatelliteDetails)
- bool gnssExtendedRegisterSatelliteDetailCallback (GNSSSatelliteDetailCallback callback)
- bool gnssExtendedDeregisterSatelliteDetailCallback (GNSSSatelliteDetailCallback callback)
- bool gnssExtendedGetPrecisionTimingOffset (int32_t *delta)

2.1.1 Typedef Documentation

2.1.1.1 typedef void(* GNSSAccuracyCallback)(const TGNSSAccuracy accuracy[], uint16_t numElements)

Callback type for extended GNSS accuracy. Use this type of callback if you want to register for extended GNSS accuracy 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

accuracy	pointer to an array of TGNSSAccuracy with size numElements
numElements,:	allowed range: >=1. If numElements >1, buffered data are provided.

2.1.1.2 typedef void(* GNSSCourse3DCallback)(const TGNSSCourse3D course[], uint16_t numElements)

Callback type for extended GNSS 3D course. Use this type of callback if you want to register for extended GNSS 3D course 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

course	pointer to an array of TGNSSCourse3D with size numElements
numElements,:	allowed range: >=1. If numElements >1, buffered data are provided.

2.1.1.3 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

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

2.1.1.4 typedef void(* GNSSUTCDateCallback)(const TUTCDate date[], const TUTCTime time[], uint16_t numElements)

Callback type for extended GNSS UTC date. Use this type of callback if you want to register for extended GNSS UTC date and the 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

date	pointer to an array of TUTCDate with size numElements
time	pointer to an array of TUTCTime with size numElements
numElements,:	allowed range: >=1. If numElements >1, buffered data are provided.

2.1.1.5 typedef void(* GNSSUTCTimeCallback)(const TUTCTime time[], uint16_t numElements)

Callback type for extended GNSS UTC time. Use this type of callback if you want to register for extended 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

time	pointer to an array of TUTCTime with size numElements
numElements,:	allowed range: >=1. If numElements >1, buffered data are provided.

2.1.2 Enumeration Type Documentation

2.1.2.1 enum EGNSSAccuracyValidityBits

TGNSSAccuracy::validityBits provides information about the currently valid signals of the GNSS accuracy data. It is a or'ed bitmask of the EGNSSAccuracyValidityBits values.

Enumerator:

GNSS_ACCURACY_PDOP_VALID Validity bit for field TGNSSAccuracy::pdop.

GNSS_ACCURACY_HDOP_VALID Validity bit for field TGNSSAccuracy::hdop.

GNSS_ACCURACY_VDOP_VALID Validity bit for field TGNSSAccuracy::vdop.

GNSS_ACCURACY_USAT_VALID Validity bit for field TGNSSAccuracy::usedSatellites.

GNSS_ACCURACY_TSAT_VALID Validity bit for field TGNSSAccuracy::trackedSatellites.

GNSS_ACCURACY_VSAT_VALID Validity bit for field TGNSSAccuracy::visibleSatellites.

GNSS_ACCURACY_SLAT_VALID Validity bit for field TGNSSAccuracy::sigmaLatitude.

GNSS_ACCURACY_SLON_VALID Validity bit for field TGNSSAccuracy::sigmaLongitude.

GNSS_ACCURACY_SALT_VALID Validity bit for field TGNSSAccuracy::sigmaAltitude.

GNSS_ACCURACY_STAT_VALID Validity bit for field TGNSSAccuracy::fixStatus.

GNSS_ACCURACY_TYPE_VALID Validity bit for field TGNSSAccuracy::fixTypeBits.

2.1.2.2 enum EGNSSCourse3DValidityBits

TGNSSCourse3D::validityBits provides information about the currently valid signals of the GNSS course 3D data. It is a or'ed bitmask of the EGNSSCourse3DValidityBits values.

Enumerator:

GNSS_COURSE3D_SPEEDLAT_VALID Validity bit for field TGNSSCourse3D-::speedLatitude.

GNSS_COURSE3D_SPEEDLON_VALID Validity bit for field TGNSSCourse3D-::speedLongitude.

GNSS_COURSE3D_SPEEDALT_VALID Validity bit for field TGNSSCourse3D-::speedAltitude.

2.1.2.3 enum EGNSSFixStatus

Description of the fix status of the GNSS reveiver.

Enumerator:

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

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.

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

2.1.2.4 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 EGNSS-FixType 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_ESTIMATED The position is propagated without additional sensor input

GNSS_FIX_TYPE_DEAD_RECKONING The position is propagated supported with additional sensor input

GNSS_FIX_TYPE_MANUAL Position is set by manual input

GNSS_FIX_TYPE_SIMULATOR_MODE Position is simulated

2.1.2.5 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_VALIDLID 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.

2.1.2.6 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.

2.1.2.7 enum EGNSSSystem

Enumeration to describe the type of GNSS system to which a particular GNSS satellite belongs.

Enumerator:

GNSS_SYSTEM_GPS GPS

GNSS_SYSTEM_GLONASS GLONASS

GNSS_SYSTEM_GALILEO GALILEO

GNSS_SYSTEM_COMPASS COMPASS / Bei Du

GNSS_SYSTEM_SBAS_WAAS WASS (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)

2.1.3 Function Documentation

2.1.3.1 bool gnssExtendedDeregister3DCourseCallback (GNSSCourse3DCallback *callback*)

Deregister extended GNSS 3D course callback. After calling this method no new GNSS 3D course data will be delivered to the client.

Parameters

callback	The callback which should be deregistered.

Returns

True if callback has been deregistered successfully.

2.1.3.2 bool gnssExtendedDeregisterAccuracyCallback (GNSSAccuracyCallback *callback*)

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

Parameters

callback	The callback which should be deregistered.

Returns

True if callback has been deregistered successfully.

2.1.3.3 bool gnssExtendedDeregisterSatelliteDetailCallback (GNSSSatelliteDetailCallback *callback*)

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

Parameters

callback	The callback which should be deregistered.

Returns

True if callback has been deregistered successfully.

2.1.3.4 bool gnssExtendedDeregisterUTCDateCallback (GNSSUTCDateCallback *callback*)

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

Parameters

callback	The callback which should be deregistered.

Returns

True if callback has been deregistered successfully.

2.1.3.5 bool gnssExtendedDeregisterUTCTimeCallback (GNSSUTCTimeCallback *callback*)

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

Parameters

callback	The callback which should be deregistered.

Returns

True if callback has been deregistered successfully.

2.1.3.6 bool gnssExtendedDestroy ( )

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

Returns

True if shutdown has been successfull.

2.1.3.7 bool gnssExtendedGet3DCourse (TGNSSCourse3D * *course*)

Method to get the extended 3D course data at a specific point in time. All valid flags are updated. The data is only guaranteed to be updated when the valid flags are true.

Parameters

course	After calling the method the currently available GNSS 3D course data is written into this parameter.

Returns

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

2.1.3.8 bool gnssExtendedGetAccuracy (TGNSSAccuracy * *accuracy*)

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

Parameters

accuracy	After calling the method the currently available GNSS accuracy data is written into this parameter.

Returns

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

2.1.3.9 bool gnssExtendedGetAntennaPosition (TGNSSDistance3D * *distance*)

Accessing static configuration information about the antenna position.

Parameters

distance	After calling the method the currently available antenna configuration data is written into this parameter.

Returns

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

Static configuration data for the extended GNSS service. The reference point mentioned in the vehicle configuration lies underneath the center of the rear axle on the surface of the road. The reference coordinate system is the car reference system as provided in the documentation. See https://collab.genivi.org/wiki/display/genivi/LBSSensorServiceRequirements-Borg#LBSSensorServiceRequirementsBorg-ReferenceSystem

2.1.3.10 bool gnssExtendedGetMetaData (TGnssMetaData * *data*)

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

Parameters

data	Meta data content about the sensor service.

Returns

True if meta data is available.

2.1.3.11 bool gnssExtendedGetPrecisionTimingOffset (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

delta	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 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.

2.1.3.12 bool gnssExtendedGetSatelliteDetails (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

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

Returns

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

2.1.3.13 bool gnssExtendedGetUTCDate (TUTCDate * *date,* TUTCTime * *time*)

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

Parameters

date	After calling the method the current GNSS UTC date is written into this parameter.
time	After calling the method the current GNSS UTC time is written into this parameter.

Returns

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

2.1.3.14 bool gnssExtendedGetUTCTime (TUTCTime * *time*)

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

Parameters

time	After calling the method the current GNSS UTC time is written into this parameter.

Returns

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

2.1.3.15 bool gnssExtendedInit ( )

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

Returns

True if initialization has been successfull.

2.1.3.16 bool gnssExtendedRegister3DCourseCallback (GNSSCourse3DCallback *callback*)

Register extended GNSS 3D course callback. This is the recommended method for continuously accessing the 3D course data. The callback will be invoked when new course data is available from the GNSS receiver. It is intended to extend the GNSS course information by data split into each axis. All valid flags are updated. The data is only guaranteed to be updated when the valid flags are true.

Parameters

callback	The callback which should be registered.

Returns

True if callback has been registered successfully.

2.1.3.17 bool gnssExtendedRegisterAccuracyCallback (GNSSAccuracyCallback *callback*)

Register GNSS accuracy callback. This is the recommended method for continuously accessing the accuracy data. The callback will be invoked when new accuracy data is available from the GNSS receiver. All valid flags are updated. The data is only guaranteed to be updated when the valid flags are true.

Parameters

callback	The callback which should be registered.

Returns

True if callback has been registered successfully.

2.1.3.18 bool gnssExtendedRegisterSatelliteDetailCallback (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

callback	The callback which should be registered.

Returns

True if callback has been registered successfully.

2.1.3.19 bool gnssExtendedRegisterUTCDateCallback (GNSSUTCDateCallback *callback*)

Register extended GNSS UTC date 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

callback	The callback which should be registered.

Returns

True if callback has been registered successfully.

2.1.3.20 bool gnssExtendedRegisterUTCTimeCallback (GNSSUTCTimeCallback *callback*)

Register extended 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

callback	The callback which should be registered.

Returns

True if callback has been registered successfully.

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

- int32_t getGnssMetaDataList (const TGnssMetaData **metadata)

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 int32_t getGnssMetaDataList (const TGnssMetaData ** *metadata*)

Retrieve the metadata of all available GNSS sensors.

Parameters

returns	a a pointer an array of TGnssMetaData (maybe NULL if no metadata is available)

Returns

number of elements in the array of TGnssMetaData

2.3 gnss-simple.h File Reference

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

Classes

- struct TGNSSSimpleConfiguration
- struct TGNSSPosition
- struct TGNSSCourse

Typedefs

- typedef void(* GNSSPositionCallback)(const TGNSSPosition pos[], uint16_t numElements)
- typedef void(* GNSSCourseCallback)(const TGNSSCourse course[], uint16_t numElements)

Enumerations

- enum EGNSSAltitudeType { GNSS_ALTITUDE_UNKNOWN = 0, GNSS_ALTITUDE_ELLIPSOIDE = 1, GNSS_ALTITUDE_ABOVE_MEAN_SEA_LEVEL = 2 }
- enum EGNSSSimpleConfigurationValidityBits { GNSS_SIMPLE_CONFIG_ALTITUDE_TYPE_VALID = 0x00000001 }
- enum EGNSSPositionValidityBits { GNSS_POSITION_LATITUDE_VALID = 0x00000001, GNSS_POSITION_LONGITUDE_VALID = 0x00000002, GNSS_POSITION_ALTITUDE_VALID = 0x00000004 }

- enum EGNSSCourseValidityBits { GNSS_COURSE_SPEED_VALID = 0x00000001, GNSS_COURSE_CLIMB_VALID = 0x00000002, GNSS_COURSE_HEADING_VALID = 0x00000004 }

Functions

- bool gnssSimpleInit ()
- bool gnssSimpleDestroy ()
- bool gnssSimpleGetConfiguration (TGNSSSimpleConfiguration *gnssConfig)
- bool gnssSimpleGetMetaData (TGnssMetaData *data)
- bool gnssSimpleGetPosition (TGNSSPosition *pos)
- bool gnssSimpleRegisterPositionCallback (GNSSPositionCallback callback)
- bool gnssSimpleDeregisterPositionCallback (GNSSPositionCallback callback)
- bool gnssSimpleGetCourse (TGNSSCourse *course)
- bool gnssSimpleRegisterCourseCallback (GNSSCourseCallback callback)
- bool gnssSimpleDeregisterCourseCallback (GNSSCourseCallback callback)

2.3.1 Typedef Documentation

2.3.1.1 typedef void(* GNSSCourseCallback)(const TGNSSCourse course[], uint16_t numElements)

Callback type for GNSS course. Use this type of callback if you want to register for GNSS course 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

course	pointer to an array of TGNSSCourse with size numElements
numElements,:	allowed range: >=1. If numElements >1, buffered data are provided.

2.3.1.2 typedef void(* GNSSPositionCallback)(const TGNSSPosition pos[], uint16_t numElements)

Callback type for GNSS position. 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

pos	pointer to an array of TGNSSPosition with size numElements
num-Elements,:	allowed range: >=1. If numElements >1, buffered data are provided.

2.3.2 Enumeration Type Documentation

2.3.2.1 enum EGNSSAltitudeType

Reference level for the GNSS altitude.

Enumerator:

GNSS_ALTITUDE_UNKNOWN Reference level is unknown.

GNSS_ALTITUDE_ELLIPSOIDE Reference level is the WGS-84 ellopsoid.

GNSS_ALTITUDE_ABOVE_MEAN_SEA_LEVEL Reference level is the geoid (mean sea level).

2.3.2.2 enum EGNSSCourseValidityBits

TGNSSCourse::validityBits provides information about the currently valid signals of the GNSS course data. It is a or'ed bitmask of the EGNSSCourseValidityBits values.

Enumerator:

GNSS_COURSE_SPEED_VALID Validity bit for field TGNSSCourse::speed.

GNSS_COURSE_CLIMB_VALID Validity bit for field TGNSSCourse::climb.

GNSS_COURSE_HEADING_VALID Validity bit for field TGNSSCourse::heading.

2.3.2.3 enum EGNSSPositionValidityBits

TGNSSPosition::validityBits provides information about the currently valid signals of the GNSS position 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_ALTITUDE_VALID Validity bit for field TGNSSPosition-::altitude.

2.3.2.4 enum EGNSSSimpleConfigurationValidityBits

TGNSSSimpleConfiguration::validityBits provides information about the currently valid values of GNSS configuration data. It is a or'ed bitmask of the EGNSSSimple-ConfigurationValidityBits values.

Enumerator:

GNSS_SIMPLE_CONFIG_ALTITUDE_TYPE_VALID Validity bit for field TGNSSSimpleConfiguration::typeOfAltitude.

2.3.3 Function Documentation

2.3.3.1 bool gnssSimpleDeregisterCourseCallback (GNSSCourseCallback *callback*)

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

Parameters

callback	The callback which should be deregistered.

Returns

True if callback has been deregistered successfully.

2.3.3.2 bool gnssSimpleDeregisterPositionCallback (GNSSPositionCallback *callback*)

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

Parameters

callback	The callback which should be deregistered.

Returns

True if callback has been deregistered successfully.

2.3.3.3 bool gnssSimpleDestroy ( )

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

Returns

True if shutdown has been successfull.

2.3.3.4 bool gnssSimpleGetConfiguration (TGNSSSimpleConfiguration * *gnssConfig*)

Accessing static configuration information about the GNSS sensor.

Parameters

gnssConfig	After calling the method the currently available GNSS configuration data is written into gnssConfig.

Returns

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

2.3.3.5 bool gnssSimpleGetCourse (TGNSSCourse * *course*)

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

Parameters

course	After calling the method the currently available position data is written into this parameter.

Returns

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

2.3.3.6 bool gnssSimpleGetMetaData (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

data	Meta data content about the sensor service.

Returns

True if meta data is available.

2.3.3.7 bool gnssSimpleGetPosition (TGNSSPosition * *pos*)

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 flags are true.

Parameters

pos	After calling the method the currently available position data is written into this parameter.

Returns

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

2.3.3.8 bool gnssSimpleInit ( )

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

Returns

True if initialization has been successfull.

2.3.3.9 bool gnssSimpleRegisterCourseCallback (GNSSCourseCallback *callback*)

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

Parameters

callback	The callback which should be registered.

Returns

True if callback has been registered successfully.

2.3.3.10 bool gnssSimpleRegisterPositionCallback (GNSSPositionCallback *callback*)

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

Parameters

callback	The callback which should be registered.

Returns

True if callback has been registered successfully.

2.4 gnss.h File Reference

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

Defines

- #define GENIVI_GNSS_API_MAJOR 2
- #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)

2.4.1 Define Documentation

2.4.1.1 #define GENIVI_GNSS_API_MAJOR 2

2.4.1.2 #define GENIVI_GNSS_API_MICRO 0

2.4.1.3 #define GENIVI_GNSS_API_MINOR 0

2.4.2 Function Documentation

2.4.2.1 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 successfull.

2.4.2.2 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

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

2.4.2.3 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 successfull.