



EnhancedPositionService API

Version 2.0.0

Thu Mar 27 2014

Accepted for release by:

Approved by the EG-LBS and the SAT for the GENIVI Gemini Release.

Abstract:

This document describes the architecture and the interface of the GENIVI EnhancedPositionService.

Keywords:

GENIVI, Position, GPS, GNSS, Positioning, Dead-Reckoning

SPDX-License-Identifier: CC-BY-SA-3.0

Copyright (C) 2014, BMW Car IT GmbH, Continental Automotive GmbH, PCA Peugeot Citroën, XS Embedded GmbH, TomTom International B.V., Alpine Electronics R&D Europe GmbH, AISIN AW CO., LTD.

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License.

To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

Participants

This document has been created by members of the Location Based Services expert group.

Change History

The following table shows the change history of the current document:

Date	Document Version	Changes
thomas.bader	2012-03-16	Created initial version
thomas.bader	2012-03-19	added API, Architecture, Discovery-Links
thomas.bader	2012-03-21	fixed typos, added sequence diagrams
thomas.bader	2012-03-26	created version 1.0, added images
marco.residori (XS Embedded)	2013-06-03	Updated context description, architecture description, UML diagrams
marco.residori (XS Embedded)	2013-06-04	API version in the XML files set to 2.0.0
marco.residori (XS Embedded)	2013-08-23	API documentation generated automatically from XML files
marco.residori (XS Embedded)	2014-03-27	Added copyright notes

Contents

1	Introduction	1
2	Architecture	1
2.1	EnhancedPositionService	2
2.2	GNSSService	3
2.3	SensorsService	3
2.4	NavigationCore	4
2.5	MapView	4
3	Requirements	5
4	API	5
4.1	Repositories and Links	5
4.2	org.genivi.positioning.EnhancedPosition	6
5	Sequence Diagrams	6
5.1	GetData (Simple)	6
5.2	GetSatelliteInfo	7
5.3	GetData (Complex)	7

1 Introduction

This document describes the architecture and the interface of the GENIVI EnhancedPositionService.

2 Architecture

The following component diagram shows how the EnhancedPositionService interacts with other GENIVI components:

- GNSSService (Library)
- SensorService (Library)
- EnhancedPositionService (Daemon with D-Bus interface)
- NavigationCore
- MapViewer

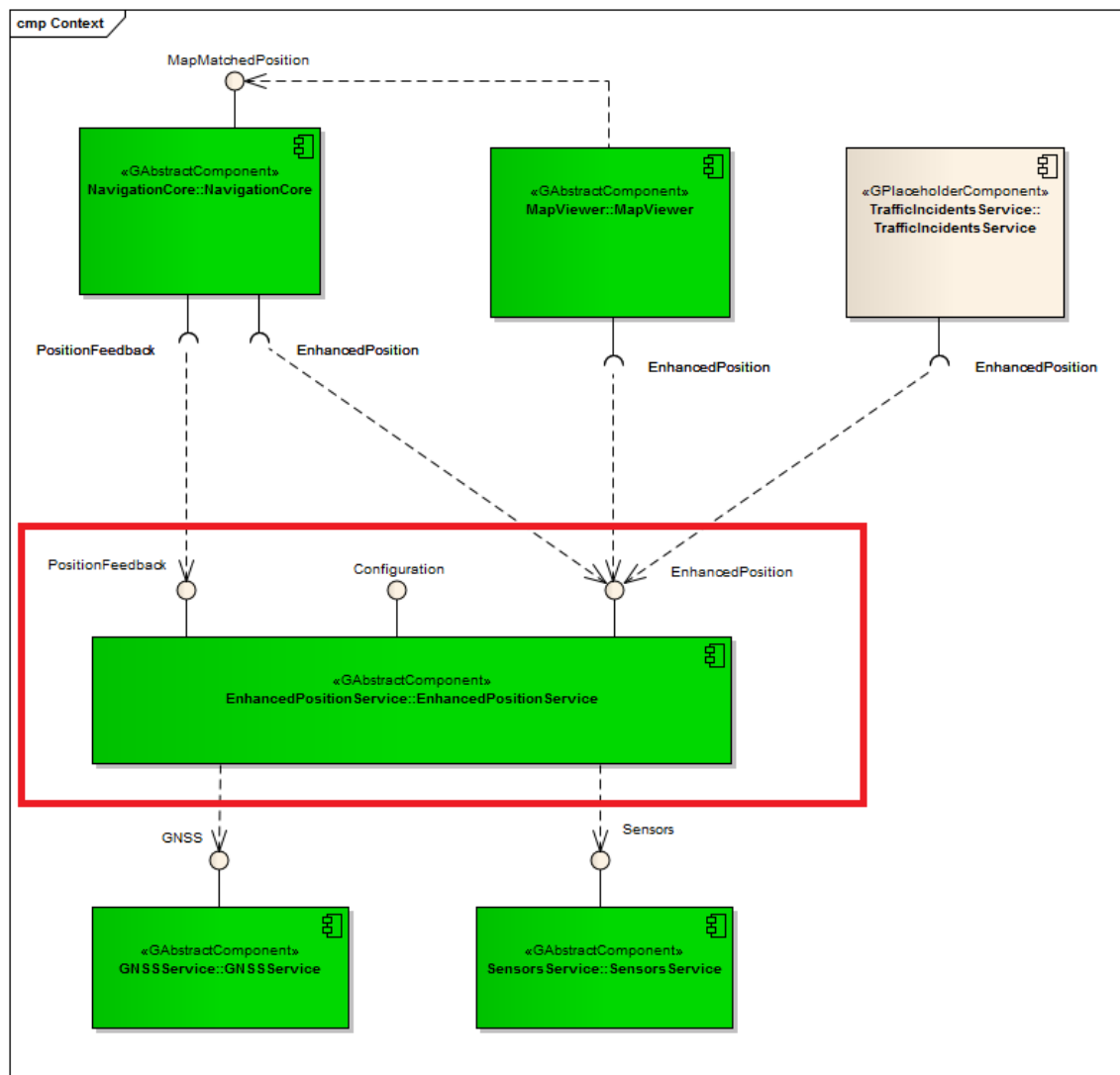


Figure 1: EnhancedPositionService

2.1 EnhancedPositionService

The EnhancedPositionService is a software component that offers positioning information to client applications. To calculate the current vehicle position, data from a GNSS receiver (e.g. GPS data) and available vehicle sensors (e.g. gyroscope and wheel ticks) are taken into account (dead-reckoning). In this way the EnhancedPositionService can calculate the current position even on roads, where the GNSS signal is too weak (e.g. in a tunnel, or in a parking garage).

The result of the map matching can be provided as feedback to this module by the NavigationCore component.

This component is the main client of the GNSSService and of the SensorsService.

The EnhancedPositionService will be typically implemented as a multi-client daemon with a D-Bus interface.

2.1.1 EnhancedPositionService Interfaces

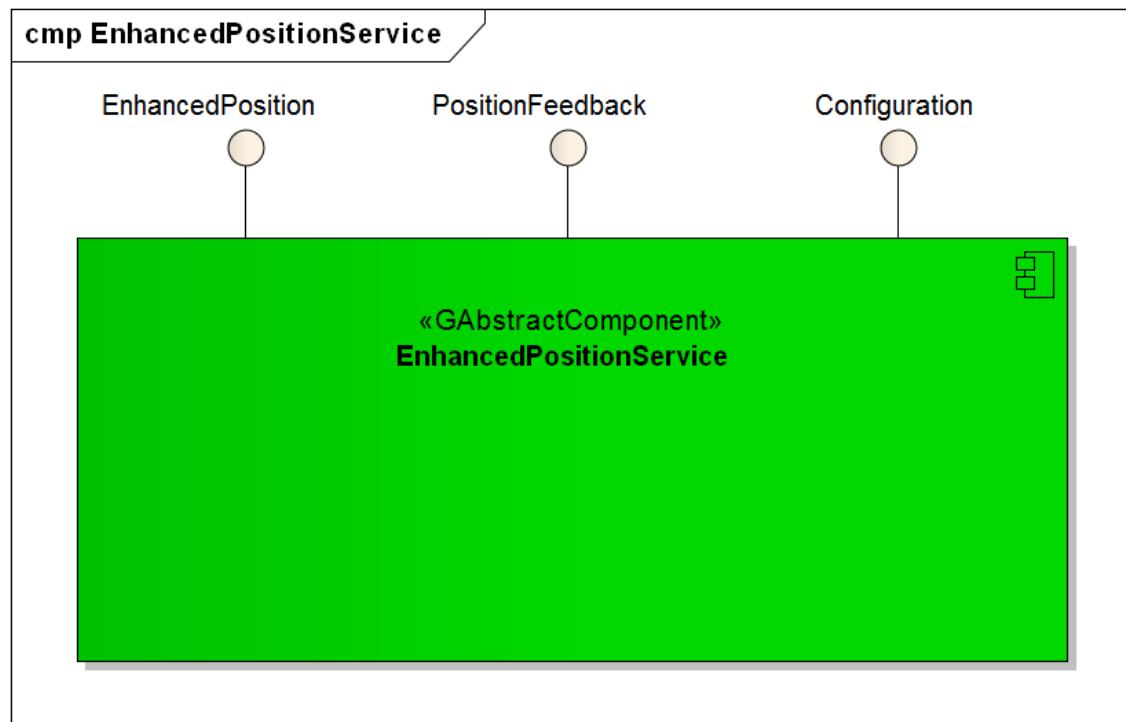


Figure 2: EnhancedPositionService Interfaces

- **EnhancedPosition** : This interface provides a 'filtered' position that takes into account the value coming from the vehicle sensors (dead-reckoning).
- **PositionFeedback** : This interface offers methods that allows the NavigationCore to provide a position feedback to the EnhancedPositionService. The component that implements the Position-Feedback interface requires the data provided by a 'map matcher' (typically the NavigationCore component). The PositionFeedback is an added improvement which does not negatively affect systems that don't support maps or have a map-matching feature.
- **Configuration**: This interface allows a client application to manage configuration parameters, like the GNSS type.

2.2 GNSSService

The GNSSService is a component that retrieves positioning data from a GNSS receiver (e.g. NMEA sentences from a GPS receiver) and presents them to its client applications.

The GNSSService will be typically implemented as a single-client library.

2.3 SensorsService

The SensorsService is a component that retrieves sensor data from several vehicle sensors (e.g. gyroscope, wheel ticks) and presents them to its client applications.

The SensorsService will be typically implemented as a single-client library.

2.4 NavigationCore

The NavigationCore is a component that is responsible for:

- providing all data for selection of locations
- calculating routes
- providing guiding information
- providing a map matched position

The NavigationCore uses the interface EnhancedPositionService to retrieve information about the current position.

The NavigationCore can match the current position to the local position and provide a feedback to the EnhancedPositionService via the interface MapFeedback.

2.5 MapViewer

This MapViewer is responsible for:

- rendering maps and other information related to the navigation on a display
- setting the graphic features that must be visualized
- setting user preferences

The Map Viewer is a separate component, logically separated from the NavigationCore. It uses the results of the map matching as input for drawing the car on the map.

3 Requirements

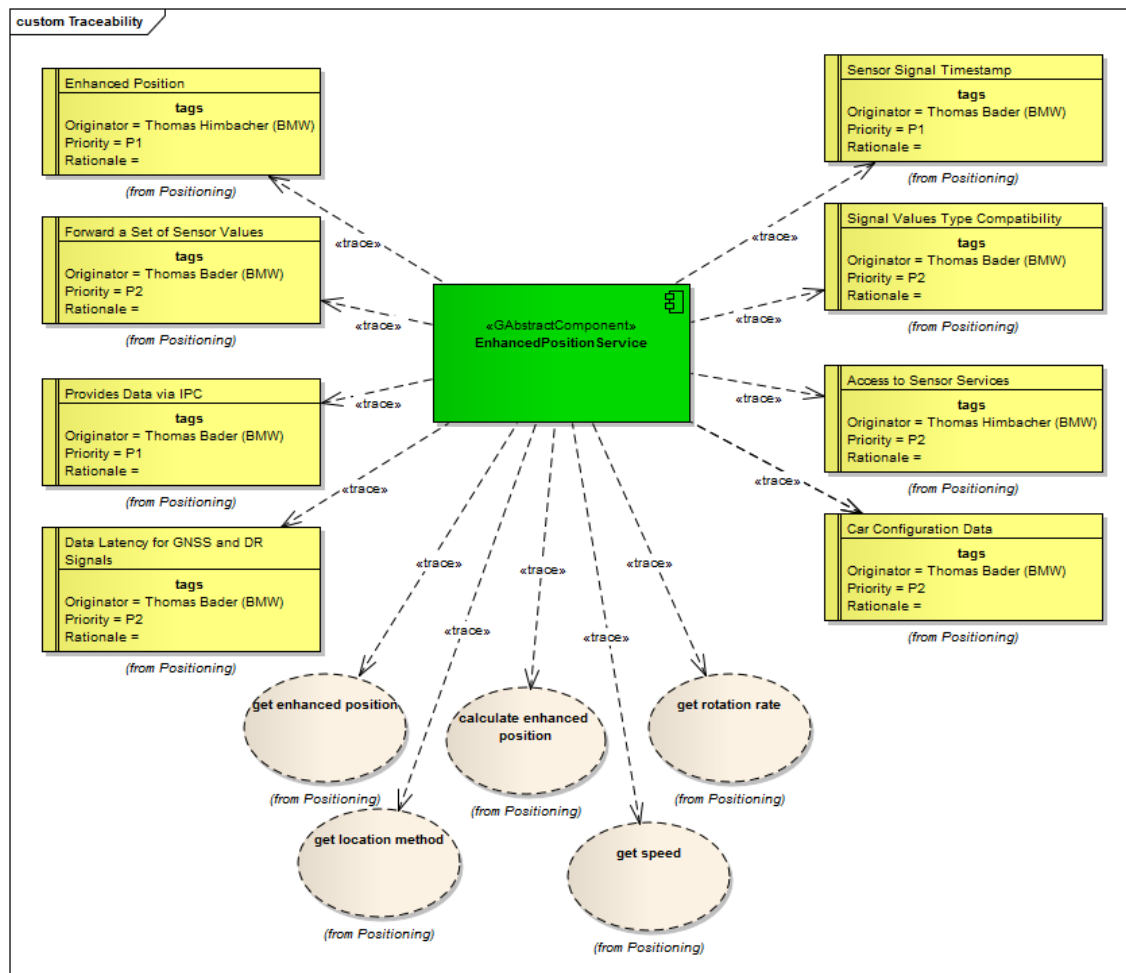


Figure 3: Enhanced Position Service - traces against requirements (generated from EA)

4 API

4.1 Repositories and Links

The official API which is release is located in the 'positioning' git repository. All header files of the EnhancedPositionService can be found at:

4.1.1 GIT Webview

<https://git.genivi.org/git/gitweb.cgi?p=positioning;a=tree;f=EnhancedPosition-Service>

4.1.2 GIT Repository

```
# clone git repository
$ git clone https://git.genivi.org/srv/git/positioning

# checkout tagged version 'gemini-final'
$ git checkout <release tag>

# navigate to the GNSSService directory containing the D-Bus APIs
$ cd EnhancedPositionService/api
```

4.2 org.genivi.positioning.EnhancedPosition

This interface offers functionalities to retrieve the (enhanced) position of the vehicle.

5 Sequence Diagrams

The following sequence diagrams are generated out of the GENIVI Enterprise Architect model.

5.1 GetData (Simple)

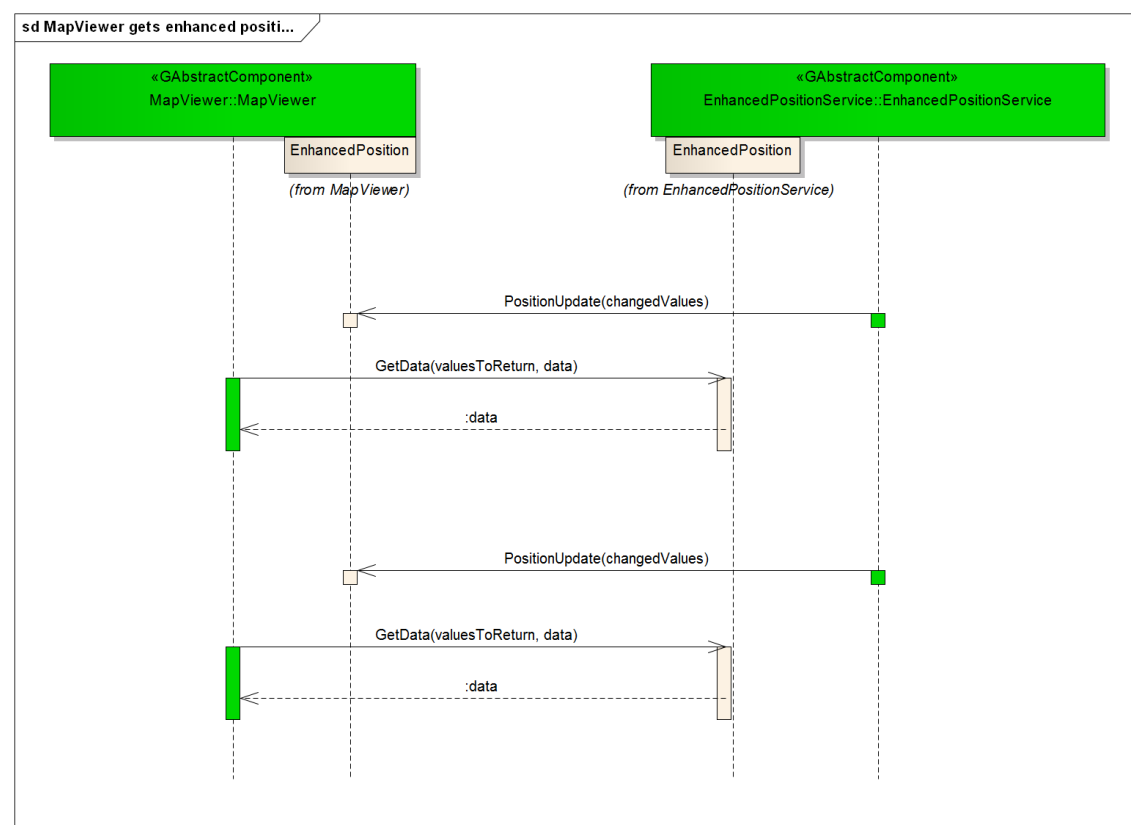


Figure 4: EnhancedPositionService - Sequence Diagram

5.2 GetSatelliteInfo

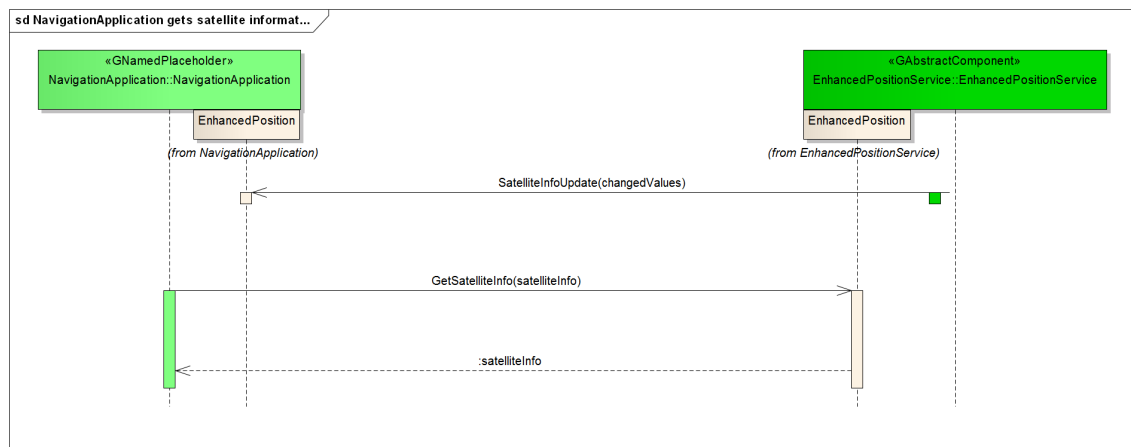


Figure 5: EnhancedPositionService - Sequence Diagram

5.3 GetData (Complex)

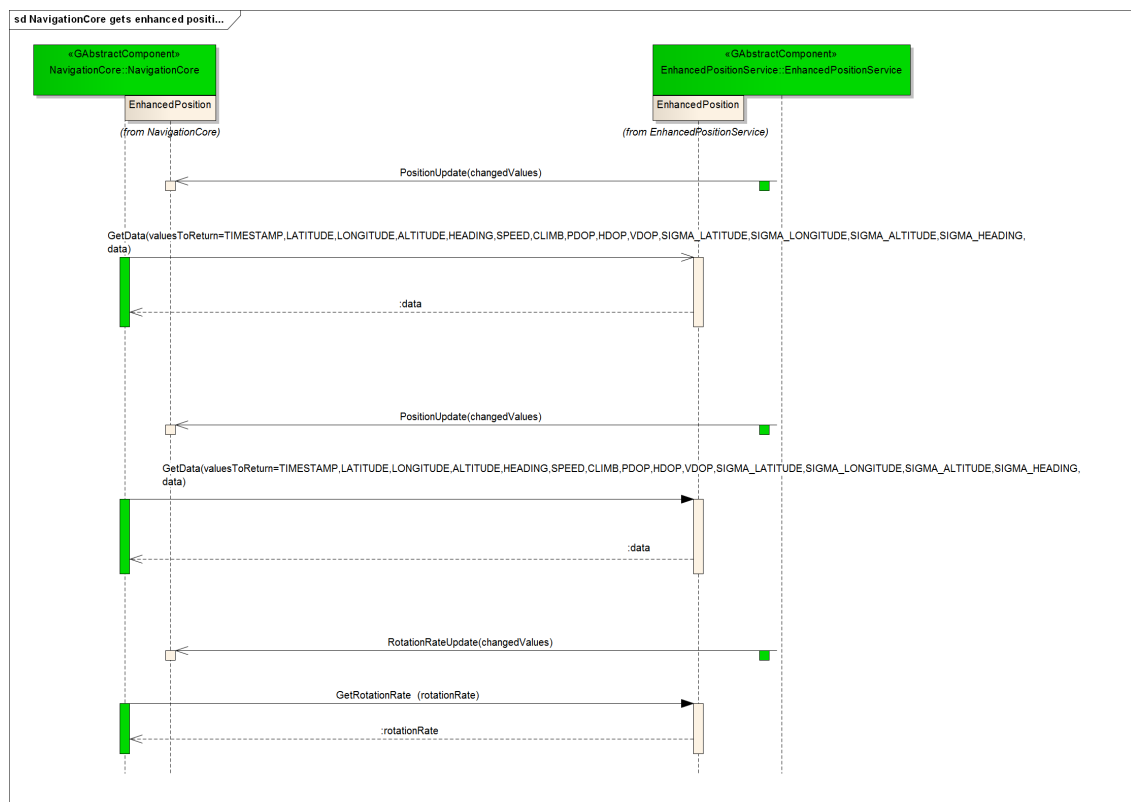


Figure 6: Enhanced Position Service - Sequence Diagram

interface

org.genivi.positioning.EnhancedPosition

version 2.0.0 (04-06-2013)

EnhancedPosition = This interface offers functionalities to retrieve the enhanced position of the vehicle

GetVersion = This method returns the API version implemented by the server application

```
method GetVersion

    version = struct(major,minor,micro,date)
    major = when the major changes, then backward compatibility with previous releases is not granted
    minor = when the minor changes, then backward compatibility with previous releases is granted, but something
    changed in the implementation of the API (e.g. new methods may have been added)
    micro = when the micro changes, then backward compatibility with previous releases is granted (bug fixes or
    documentation modifications)
    date = release date (e.g. 21-06-2011)
    out (qqqs) version
```

GetData = This method returns a given set of positioning data (e.g. Position, Course, Accuracy, Status, ...)

```
method GetData

    valuesToReturn= array[fieldType]
    key =
    enum(INVALID,TIMESTAMP,LATITUDE,LONGITUDE,ALTITUDE,HEADING,SPEED,CLIMB,ROLL_RATE,PITCH_RATE,YAW_RATE,...,ALL)
    in aq valuesToReturn

    position = dictionary[key,value]
    dictionary = array of tuples (key,value)
    Invalid data is not be returned to the client application
    key =
    enum(INVALID,TIMESTAMP,LATITUDE,LONGITUDE,ALTITUDE,HEADING,SPEED,CLIMB,ROLL_RATE,PITCH_RATE,YAW_RATE,PDOP,HDOP,VDOP,SIGMA_LATITUDE,SIGMA_LONGITUDE,SIGMA_ALTITUDE,SIGMA_HEADING,...,
    ,ALL
    key = TIMESTAMP, value = value of type 'i', that represents a timestamp in ms
    key = LATITUDE, value = value of type 'd', that expresses the latitude of the current position. Range [-90;+90].
    Example: 48.053250
    key = LONGITUDE, value = value of type 'd', that expresses the longitude of the current position. Range [-
    180;+180]. Example: 8.324500
    key = ALTITUDE, value = value of type 'i', that expresses the altitude above the sea level of the current position in
    meters
    key = HEADING, value = value of type 'i', that expresses the course angle in degree. Range [0;360]. 0 = north, 90
    = east, 180 = south, 270 = west
    key = SPEED, value = value of type 'd', that expresses speed measured in m/s. A negative value indicates that the
    vehicle is moving backwards
    key = CLIMB, value = value of type 'i', that expresses the road gradient in degrees
    key = ROLL_RATE, value = value of type 'i', rotation rate around the X-axis in degrees/s. Range [-100;+100]
    key = PITCH_RATE, value = value of type 'i', rotation rate around the Y-axis in degrees/s. Range [-100;+100]
    key = YAW_RATE, value = value of type 'i', rotation rate around the Z-axis in degrees/s. Range [-100;+100]
    key = PDOP, value = value of type 'd', that represents the positional (3D) dilution of precision
    key = HDOP, value = value of type 'd', that represents the horizontal (2D) dilution of precision
    key = VDOP, value = value of type 'd', that represents vertical (altitude) dilution of precision
    key = SIGMA_LATITUDE, value = value of type 'd', that represents the standard deviation for latitude in m
    key = SIGMA_LONGITUDE, value = value of type 'd', that represents the standard deviation for longitude in m
    key = SIGMA_ALTITUDE, value = value of type 'd', that represents the standard deviation for altitude in m
    key = SIGMA_HEADING, value = value of type 'd', that represents the standard deviation for altitude in degrees
    out a{qv} data
```

GetPosition = This method returns the current position

```
method GetPosition

    position = dictionary[key,value]
    dictionary = array of tuples (key,value)
    Invalid data is not be returned to the client application
    key = enum(INVALID,TIMESTAMP,LATITUDE,LONGITUDE,ALTITUDE,HEADING,SPEED,CLIMB,...)
    key = TIMESTAMP, value = value of type 'i', that represents a timestamp in ms
    key = LATITUDE, value = value of type 'd', that expresses the latitude of the current position. Range [-90;+90].
    Example: 48.053250
    key = LONGITUDE, value = value of type 'd', that expresses the longitude of the current position. Range [-
    180;+180]. Example: 8.324500
    key = ALTITUDE, value = value of type 'i', that expresses the altitude above the sea level of the current position in
    meters
    key = HEADING, value = value of type 'i', that expresses the course angle in degree. Range [0;360]. 0 = north, 90
    = east, 180 = south, 270 = west
    key = SPEED, value = value of type 'd', that expresses speed measured in m/s. A negative value indicates that the
    vehicle is moving backwards
    key = CLIMB, value = value of type 'i', that expresses the road gradient in degrees
    out a{qv} position
```

PositionUpdate = This signal is called to notify a client application of a position change. The update frequency is implementation specific. The maximal allowed frequency is 10Hz

```
signal PositionUpdate

    changedValues = array[value]
    value = enum(INVALID,TIMESTAMP,LATITUDE,LONGITUDE,ALTITUDE,HEADING,SPEED,CLIMB,...)
    in aq changedValues
```

GetRotationRate = This method returns the rotation rate

method GetRotationRate

```
rotationRates = dictionary[key,value]
dictionary = array of tuples (key,value)
If you request for a specific value which is invalid, it's not returned in the dictionary.
key = enum(INVALID,TIMESTAMP,ROLL_RATE,PITCH_RATE,YAW_RATE, ... )
key = TIMESTAMP, value = value of type 't', that represents a timestamp in ms
key = ROLL_RATE, value = rotation rate around the X-axis in degrees/s. Range [-100:+100]
key = PITCH_RATE, value = rotation rate around the Y-axis in degrees/s. Range [-100:+100]
key = YAW_RATE, value = rotation rate around the Z-axis in degrees/s. Range [-100:+100]
out a{qv} rotationRate
```

RotationRateUpdated = This signal is emitted when the rotation rate changes

signal RotationRateUpdate

```
changedValues = array[key]
key = enum(INVALID,TIMESTAMP,ROLL_RATE,PITCH_RATE,YAW_RATE, ... )
in aq changedValues
```

GetAccuracy = This method returns the accuracy

method GetAccuracy

```
accuracy = dictionary[key,value]
dictionary = array of tuples (key,value)
If you request for a specific value which is invalid, it's not returned in the dictionary.
key =
enum(INVALID,TIMESTAMP,PDOP,HDOP,VDOP,SIGMA_LATITUDE,SIGMA_LONGITUDE,SIGMA_ALTITUDE,SIGMA_HEADING,...
)
key = TIMESTAMP, value = value of type 't', that represents a timestamp in ms
key = PDOP, value = value of type 'd', that represents the positional (3D) dilution of precision
key = HDOP, value = value of type 'd', that represents the horizontal (2D) dilution of precision
key = VDOP, value = value of type 'd', that represents vertical (altitude) dilution of precision
key = SIGMA_LATITUDE, value = value of type 'd', that represents the standard deviation for latitude in m
key = SIGMA_LONGITUDE, value = value of type 'd', that represents the standard deviation for longitude in m
key = SIGMA_ALTITUDE, value = value of type 'd', that represents the standard deviation for altitude in m
key = SIGMA_HEADING, value = value of type 'd', that represents the standard deviation for altitude in degrees
out a{qv} accuracy
```

AccuracyUpdated = This signal is emitted when the accuracy changes

signal AccuracyUpdate

```
changedValues = array[value]
value =
enum(INVALID,TIMESTAMP,PDOP,HDOP,VDOP,SIGMA_LATITUDE,SIGMA_LONGITUDE,SIGMA_ALTITUDE,SIGMA_HEADING,
... )
in aq changedValues
```

GetSatelliteInfo = This method returns information about the current satellite constellation

method GetSatelliteInfo

```
satelliteInfo = dictionary[key,value]
dictionary = array of tuples (key,value)
If you request for a specific value which is invalid, it's not returned in the dictionary.
key =
enum(INVALID,TIMESTAMP,USED_SATELLITES,TRACKED_SATELLITES,VISIBLE_SATELLITES,SATELLITE_DETAILS,
...)
key = TIMESTAMP, value = value of type 't', that represents a timestamp in ms
key = USED_SATELLITES, value = value of type 'y', that represents the number of used satellites
key = TRACKED_SATELLITES, value = value of type 'y', that represents the number of tracked satellites
key = VISIBLE_SATELLITES, value = value of type 'y', that represents the number of visible satellites
key = SATELLITE_DETAILS, value = value of type 'a(ubuuu)', that represents an
array(struct{satId,inUse,elevation,azimuth,snr}). The satID numbering scheme shall be 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
out a{qv} satelliteInfo
```

SatelliteInfoUpdate = This signal is emitted when information about the current satellite information is updated

signal SatelliteInfoUpdate

```
changedValues = array[value]
key =
enum(INVALID,TIMESTAMP,USED_SATELLITES,TRACKED_SATELLITES,VISIBLE_SATELLITES,SATELLITE_DETAILS,
...)
in aq changedValues
```

GetStatus = This method returns the status of this service

method GetStatus

```
status = dictionary[key,value]
dictionary = array of tuples (key,value)
If you request for a specific value which is invalid, it's not returned in the dictionary.
key = enum(INVALID,TIMESTAMP,GNSS_FIX_STATUS,DR_STATUS, ... )
key = TIMESTAMP, value = value of type 't', that represents a timestamp in ms
key = GNSS_FIX_STATUS, value = value of type 'q', that represents an
enum(INVALID(0x00),NO_FIX(0x01),TIME_FIX(0x02),2D_FIX(0x03),3D_FIX(0x04), ... )
key = DR_STATUS, value = value of type 'b', where TRUE means that a dead-reckoning algorithm has been used
to calculate the current position
out a{qv} status
```

StatusUpdate = This signal is emitted when the status of this service changes

signal StatusUpdate

changedValues = array[value]

key = enum(INVALID,TIMESTAMP,GNSS_FIX_STATUS,DR_STATUS, ...)

in aq changedValues

GetTime = This method returns UTC time and date

method GetTime

time = dictionary[key,value]

dictionary = array of tuples (key,value)

If you request for a specific value which is invalid, it's not returned in the dictionary.

key = enum(INVALID,TIMESTAMP,YEAR,MONTH,DAY,HOUR,MINUTE,SECOND,MS, ...)

key = TIMESTAMP, value = value of type 't', that represents a timestamp in ms

key = YEAR, value = value of type 'q', 4 digits number that indicates the year. Example: 2012

key = MONTH, value = value of type 'y', 2 digits number that indicates the month. Example: 03 means March

key = DAY, value = value of type 'y', 2 digits number that indicates the day. Range [0:31]. Example: 07

key = HOUR, value = value of type 'y', 2 digits number that indicates the hour. Range [0:23]. Example: 01

key = MINUTE, value = value of type 'y', 2 digits number that represents the minutes. Range [0:59]. Example: 01

key = SECOND, value = value of type 'y', 2 digits number that represents the seconds. Range [0:59]. Example: 01

key = MS, value = value of type 'q', 3 digits number that represents the milliseconds. Range [0:999]. Example: 007

out a{qv} time

interface

org.genivi.positioning.Configuration

version 2.0.0 (04-06-2013)

Configuration = This interface allows a client application to set and retrieve configuration options

GetVersion = This method returns the API version implemented by the server application.

method GetVersion

version = struct(major,minor,micro,date)

major = when the major changes, then backward compatibility with previous releases is not granted

minor = when the minor changes, then backward compatibility with previous releases is granted, but something changed in the implementation of the API (e.g. new methods may have been added)

micro = when the micro changes, then backward compatibility with previous releases is granted (bug fixes or documentation modifications)

date = release date (e.g. 21-06-2011)

out (qqqs) version

GetProperties = This method returns all global system properties.

method GetProperties

out a{sv} properties

SetProperty = This method changes the value of the specified property

Only properties that are listed as read-write are changeable

On success a PropertyChanged signal will be emitted

method SetProperty

name = property name

in s name

value = property value

in v value

error org.genivi.positioning.Configuration.Error.InvalidProperty

PropertyChanged = This signal is emitted when a property changes

signal PropertyChanged

name = property name

in s name

value = property value

in **v** value

SatelliteSystem = enum(INVALID,GPS,GLONASS,GALILEO,COMPASS, ...)

property SatelliteSystem *readwrite* **q**

UpdateInterval = update interval in ms

property UpdateInterval *readwrite* **i**

GetSupportedProperties = This method returns all supported global system properties

method GetSupportedProperties

properties = array[property]

property = dictionary[key,value]

key = enum(SatelliteSystem,UpdateInterval, ...)

key = SatelliteSystem, value = value of type 'aq'; 'q' is an enum(INVALID,GPS,GLONASS,GALILEO,COMPASS, ...)

key = UpdateInterval, value = value of type 'ai'; 'i' is the update interval in ms

out **a{sv}** properties

interface

org.genivi.positioning.PositionFeedback

version 2.0.0 (04-06-2013)

PositionFeedback = This interface allows the application implementing the map-matching algorithm to provide a position feedback to the EnhancedPositionService

GetVersion = This method returns the API version implemented by the server application

method GetVersion

version = struct(major,minor,micro,date)

major = when the major changes, then backward compatibility with previous releases is not granted

minor = when the minor changes, then backward compatibility with previous releases is granted, but something changed in the implementation of the API (e.g. new methods may have been added)

micro = when the micro changes, then backward compatibility with previous releases is granted (bug fixes or documentation modifications)

date = release date (e.g. 21-06-2011)

out (qqqs) version

SetPositionFeedback = This method allows a client application to provide the EnhancedPositionService with a position feedback

Note: This interface is typically used by the application that implements the map-matching algorithm

Such application can hand over to the EnhancedPositionService an array of map-matched positions with different values of reliability

method SetPositionFeedback

feedback = array[position]

position = dictionary[key,value]

dictionary = array of tuples (key,value)

key = enum(INVALID,LATITUDE,LONGITUDE,ALTITUDE,HEADING,SPEED,CLIMB,RELIABILITY_INDEX, ...)

key = LATITUDE, value = value of type 'd', that expresses the latitude of the current position in format %3.6f. [-90,+90]. Example: 48.053250

key = LONGITUDE, value = value of type 'd', that expresses the longitude of the current position in format %3.6f. [-180,+180]. Example: 8.324500

key = ALTITUDE, value = value of type 'i', that expresses the altitude above the sea level of the current position in meters

key = HEADING, value = value of type 'i', that expresses the course angle in degree. [0,360]. Example: 0 => north, 90 => east, 180 => south, 270 => west

key = SPEED, value = value of type 'd', that expresses speed measured in m/s

key = CLIMB, value = value of type 'i', that expresses the inclination measured in degrees

key = RELIABILITY_INDEX, value = value of type 'y', that indicates the position feedback reliability. It can assume values from 0 to 100

in aa{qv} feedback

timestamp = timestamp in ms

in t timestamp

feedbackType = enum(INVALID,MAP_MATCHED_FEEDBACK,TEST_FEEDBACK, ...)

in q feedbackType

constants *EnhancedPositionService* version 2.0.0 (04-06-2013)

• *This document defines the constants that are used in the EnhancedPositionService APIs*

• INVALID = 0x0000

• TIMESTAMP = 0x0001

• ALL = 0xffff

• LATITUDE = 0x0020

• LONGITUDE = 0x0021

• ALTITUDE = 0x0022

• HEADING = 0x0030

• SPEED = 0x0031

• CLIMB = 0x0032

• COUNTRY = 0x0040

• CITY = 0x0041

• STREET = 0x0042

• NUMBER = 0x0043

• CROSSING = 0x0044

• DISTRICT = 0x0045

• TIMEZONE_OFFSET = 0x0046

• DAYLIGHT_OFFSET = 0x0047

• MATCH_TYPE = 0x0048

• ROLL_RATE = 0x0060

• PITCH_RATE = 0x0061

• YAW_RATE = 0x0062

• GNSS_FIX_STATUS = 0x0070

• DR_STATUS = 0x0071

• MM_STATUS = 0x0072

- RELIABILITY_INDEX = 0x0073
- MAP_MATCHED_FEEDBACK = 0x0074
- TEST_FEEDBACK = 0x0075
- PDOP = 0x0080
- HDOP = 0x0081
- VDOP = 0x0082
- SIGMA_LATITUDE = 0x0083
- SIGMA_LONGITUDE = 0x0084
- SIGMA_ALTITUDE = 0x0085
- SIGMA_HEADING = 0x0086
- YEAR = 0x00a0
- MONTH = 0x00a1
- DAY = 0x00a2
- HOUR = 0x00a3
- MINUTE = 0x00a4
- SECOND = 0x00a5
- MS = 0x00a6
- GPS = 0x00b0
- GLONASS = 0x00b1
- GALILEO = 0x00b2
- COMPASS = 0x00b3
- USED_SATELLITES = 0x00c0
- TRACKED_SATELLITES = 0x00c1
- VISIBLE_SATELLITES = 0x00c2
- SATELLITE_DETAILS = 0x00c3