

**GENIVI Alliance**

GENIVI Document

EnhancedPositionService

Draft Version .0

**19-Jan-2015**

**Sponsored by:**

**Abstract:**

**Keywords:**

**License:**

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

Copyright © , 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.

All rights reserved.

The information within this document is the property of the copyright holders and its use and disclosure are restricted. Elements of GENIVI Alliance specifications may be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (and such third parties may or may not be members of GENIVI Alliance). GENIVI Alliance and the copyright holders are not responsible and shall not be held responsible in any manner for identifying, failing to identify, or for securing proper access to or use of, any or all such third party intellectual property rights.

GENIVI and the GENIVI Logo are trademarks of GENIVI Alliance in the U.S. and/or other countries. Other company, brand and product names referred to in this document may be trademarks that are claimed as the property of their respective owners.

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

The full license text is available at http://creativecommons.org/licenses/by-sa/4.0

The above notice and this paragraph must be included on all copies of this document that are made.

GENIVI Alliance

2400 Camino Ramon, Suite 375

San Ramon, CA 94583, USA

Revision History

The following table shows the revision history for this document.

Document revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Description |
| 10-Dec-2014 | 3.0.0 | Marco Residori, XS Embedded (now part of Mentor Graphics) | Updated API documentation and sequence diagrams. This is the first version of this document that uses the new GENIVI component specification template. Improvements after EG-LBS review |
| 19-Dec-2015 | 3.0.0 | Helmut Schmidt  Continental Automotive GmbH | Update text according remaining review comments |

Table of Contents

[1 Introduction 1](#_Toc409444916)

[1.1 System Overview 1](#_Toc409444917)

[1.2 Component Overview 1](#_Toc409444918)

[1.3 Document Overview 1](#_Toc409444919)

[2 References 2](#_Toc409444920)

[3 Glossary 3](#_Toc409444921)

[4 Requirements 4](#_Toc409444922)

[5 Constraints and Assumptions 5](#_Toc409444923)

[6 Architecture 6](#_Toc409444924)

[6.1 Architecture Overview 6](#_Toc409444925)

[6.1.1 Component Dependencies 7](#_Toc409444926)

[6.1.2 Component Traceability 8](#_Toc409444927)

[6.2 EnhancedPositionService 9](#_Toc409444928)

[6.2.1 Responsibility and Features 9](#_Toc409444929)

[6.2.2 Provided Interfaces 9](#_Toc409444930)

[6.2.3 Required Interfaces 9](#_Toc409444931)

[6.3 GNSSService 10](#_Toc409444932)

[6.3.1 Responsibility and Features 10](#_Toc409444933)

[6.3.2 Provided Interfaces 10](#_Toc409444934)

[6.3.3 Required Interfaces 10](#_Toc409444935)

[6.4 SensorsService 11](#_Toc409444936)

[6.4.1 Responsibility and Features 11](#_Toc409444937)

[6.4.2 Provided Interfaces 11](#_Toc409444938)

[6.4.3 Required Interfaces 11](#_Toc409444939)

[7 Collaboration 12](#_Toc409444940)

[7.1 Get Enhanced Position 12](#_Toc409444941)

[7.1.1 MapViewer retrieves enhanced position 12](#_Toc409444942)

[7.1.2 NavigationCore retrieves enhanced position 13](#_Toc409444943)

[7.2 Get Rotation Rate 14](#_Toc409444944)

[7.2.1 LBS Application retrieves rotation rate 14](#_Toc409444945)

[7.3 Get Satellite Details 15](#_Toc409444946)

[7.3.1 Navigation Application retrieves satellite information 15](#_Toc409444947)

[7.4 Set Navigation System 16](#_Toc409444948)

[7.4.1 Navigation Application sets navigation system 16](#_Toc409444949)

[8 Implementation 17](#_Toc409444950)

[8.1 Available Implementation details 17](#_Toc409444951)

[8.2 Usage examples 17](#_Toc409444952)

[8.3 Test Plan 17](#_Toc409444953)

[9 Interfaces 18](#_Toc409444954)

[9.1 D-Bus 18](#_Toc409444955)

[9.2 Git Repository 18](#_Toc409444956)

[9.3 Naming Conventions 18](#_Toc409444957)

[9.4 Data Types Convention 18](#_Toc409444958)

[9.5 Errors 19](#_Toc409444959)

# Introduction

## System Overview

The GENIVI Software Platform is a platform consisting of standardized middleware, application layer interfaces and frameworks defined or adopted by the GENIVI Alliance.

## Component Overview

The EnhancedPositionService is a software component of the above mentioned GENIVI Software Platform 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) or too inaccurate (e.g. in a city or in a canyon).

## Document Overview

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

# References

The following standards and specifications contain provisions, which through reference in this document constitute provisions of this specification. All the standards and specifications listed are normative references. At the time of publication, the editions indicated were valid. All standards and specifications are subject to revision, and parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the standards and specifications indicated below.

1. “GENIVI GNSSService – Component Specification” - <http://git.projects.genivi.org/?p=lbs/positioning.git;a=tree;f=gnss-service/doc>
2. “GENIVI SensorsService – Component Specification” –<http://git.projects.genivi.org/?p=lbs/positioning.git;a=tree;f=sensors-service/doc>
3. GENIVI UML Model - <https://svn.genivi.org/uml-model/genivi/trunk>

# Glossary

|  |  |  |
| --- | --- | --- |
| Acronym | Term | Definition |
| GNSS | Global Navigation Satellite System | GNSS is a space-based satellite navigation system that provides location and time information. |
| GPS | Global Positioning System | GPS is a space-based GNSS maintained by the United States government. |
| GLONASS | Globalnaya navigatsionnaya sputnikovaya sistema | GLONASS is a space-based GNSS operated by the Russian Aerospace Defence Forces. |
| BDS | BeiDou Navigation Satellite System | BDS is a [Chinese](http://en.wikipedia.org/wiki/China) GNSS, also known as COMPASS. |
|  | Galileo | Galileo is a GNSS currently being built by the European Union (EU) and European Space Agency (ESA). |
|  | Vehicle Sensors | Vehicle sensors are sensors used for positioning calculation which are located either in the vehicle itself or directly in the unit where the EnhancedPositionService is deployed. Examples are Gyroscopes, Accelerometers, wheel tick or vehicle speed sensors. |
| DR | Dead Reckoning | In strict sense:  A technique that calculates the current position of a vehicle by integrating the relative changes in heading and distance over time since leaving a known starting point. The starting point can be determined e.g. from a GNSS system and the heading and distance changes can be determined from the vehicle sensors.  In a more common sense:  The fusion of GNSS and vehicle sensor data to calculate improved position and velocity. I.e. even when a GNSS fix is available. |
|  |  |  |

Table – Acronym and Term Definitions

# Requirements

The requirements related to the EnhancedPositionService are located in the GENIVI UML model (see [[3]](#_References)) in the package *GENIVI Model/LogicalView/SW Platform requirements/Location Based Services/Positioning.*

# Constraints and Assumptions

This is a handwritten chapter that summarizes the constraints and assumptions done in the project for the component.

# Architecture

The information in this chapter is provided only for information purpose; this is not a normative part.

## Architecture Overview

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

• GNSSService (C library)

• SensorsService (C library)

• NavigationCore (example of client application)  
• MapViewer (example of client application)



### Component Dependencies

The EnhancedPositionService depends on the following GENIVI components:

* GNSSService (library)
* SensorsService (library)



### Component Traceability

The following diagrams shows to which requirements and use cases realizations the EnhancedPositionService is traced to:



## EnhancedPositionService



### Responsibility and Features

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.

### Provided 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.

### Required Interfaces

* **GNSS**: This interface abstracts the access to a GNSS device. Please see [[1].](#_References)
* **Sensors**:This interface abstracts the access to vehicle sensors. Please see [[2]](#_References).

## GNSSService



### Responsibility and Features

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.

### Provided Interfaces

The interfaces provided by this component are described at [1].

### Required Interfaces

None.

## SensorsService



### Responsibility and Features

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.

### Provided Interfaces

The interfaces provided by this component are described at [2].

### Required Interfaces

None.

# Collaboration

## Get Enhanced Position

### MapViewer retrieves enhanced position

The following sequence diagram describes how a client application can retrieve the vehicle position.



### NavigationCore retrieves enhanced position

The following sequence diagram describes how a client application can retrieve the vehicle position.



## Get Rotation Rate

### LBS Application retrieves rotation rate

The following sequence diagram describes how a client application can retrieve the vehicle rotation rate.



## Get Satellite Details

### Navigation Application retrieves satellite information

The following sequence diagram describes how a client application can retrieve satellite information.



## Set Navigation System

### Navigation Application sets navigation system

The following sequence diagram describes how a client application can set the satellite system.



# Implementation

## Available Implementation details

A Proof of concept (PoC) of the EnhancedPositionServiceis is available at: <http://git.projects.genivi.org/?p=lbs/positioning.git;a=tree>

## Usage examples

Please see: <http://git.projects.genivi.org/?p=lbs/positioning.git;a=tree;f=enhanced-position-service/test>.

## Test Plan

Please see: <http://git.projects.genivi.org/?p=lbs/positioning.git;a=blob;f=enhanced-position-service/doc/testplan.txt>

# Interfaces

The following pages describe the interfaces of the EnhancedPositionService.

## D-Bus

The EnhancedPositionService interfaces are D-Bus interfaces. They are defined using the D-Bus introspection data format, which is nothing but an IDL expressed in XML format.

For more information about the D-Bus data types please refer to the following website:

<http://dbus.freedesktop.org/doc/dbus-specification.html#message-protocol-signatures>

For more information about the D-Bus introspection data format, please refer to the following website:

<http://dbus.freedesktop.org/doc/dbus-specification.html#introspection-format>

## Git Repository

The EnhancedPositionService interfaces can be found in the GENIVI Git repository at:

<http://git.projects.genivi.org/?p=lbs/positioning.git;a=tree;f=enhanced-position-service/api>

## Naming Conventions

Please see <http://dbus.freedesktop.org/doc/dbus-specification.html>.

|  |  |  |
| --- | --- | --- |
| **Element** | **Description** | **Example** |
| Interface File | genivi.<component name or domain in lowercase character>.<interface name in lowercase characters> | org.genivi.positioning.Configuration |
| Methods/Signal/Properties | Camel case naming convention  First letter uppercase | GetPositionInfo |
| Arguments | Camel case naming convention  First letter lowercase | valuesToReturn |
|  |  |  |

## Data Types Convention

D-bus types code are used. Please refer to the following webpage for more information:

<http://dbus.freedesktop.org/doc/dbus-specification.html>

|  |  |  |
| --- | --- | --- |
| **Element** | **D-Bus Data Type Code** | **Example** |
| Enumerators | q (uint16) |  |
| Handles | y (uint8) |  |
| Maps | a{qv} | Dictionary of tuples (key, value)  The key is expressed as an enumerator |
|  |  |  |

## Errors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Error Type** | **Description** | **Example** | **Error Documentation** | **Note** |
| User Error | Error caused by user actions | The user tries to start route guidance, although guidance is already running | Application specific error string documented in the XML file | Can occur in final product |
| Hardware Error | Error related to hardware/database related problems | No map data | Application specific error string documented in the XML file | Can occur in final product |
| Protocol Error | Error caused by wrong sequence of commands | Wrong sequence of commands to enter destination | Standard D-Bus error string | Should not occur in final product |
| Bus Error | D-Bus communication error | Bus busy | Standard D-Bus error string | Can occur in final product |
| Programming Error | Programming Error | Invalid parameters | Standard D-Bus error string and debug messages | Should not occur in production code |

Only application-specific errors are documented directly in the interfaces (XML files). For all other errors, standard D-Bus strings are used. These kinds of strings are not documented in the interfaces. It is implicitly assumed that every method may return a standard D-Bus error string.

Please see <http://dbus.freedesktop.org/doc/api/html/group__DBusProtocol.html>.