

**GENIVI Alliance**

GENIVI Document CS00050

Free Text Search

Component Specification

Draft Version 0.2

**2017-01-30**

**Sponsored by:**

GENIVI Alliance

**Abstract:**

This document provides the Component Specification for the Free Text Search Component. This component provides functionality for searching navigable locations (like addresses and POI's) based on free text.

**Keywords:**

GENIVI, Navigation, Free Text Search, Location Based Services.

**License:**

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

Copyright © , PCA Peugeot Citroën, XS Embedded GmbH, TomTom International B.V., Continental Automotive GmbH, Alpine Electronics R&D Europe GmbH, Harman-Becker Automotive GmbH.  
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

Document revision history

|  |  |  |  |
| --- | --- | --- | --- |
| *Date* | *Version* | *Author* | *Description* |
| 2015-12-22 | 0.1 | Peter Goedegebure (TomTom) | Initial revision. |
| 2017-01-30 | 0.2 | Morteza Damavandpeyma (TomTom) | Draft version |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 1](#_Toc473533332)

[1.1 System Overview 1](#_Toc473533333)

[1.2 Subsystem Overview 1](#_Toc473533334)

[1.3 Component Overview 1](#_Toc473533335)

[2 References 3](#_Toc473533336)

[3 Glossary 4](#_Toc473533337)

[4 Requirements 5](#_Toc473533338)

[4.1 Use Cases 5](#_Toc473533339)

[4.2 Requirements 7](#_Toc473533340)

[4.2.1 Functional Requirements 7](#_Toc473533341)

[4.2.2 Non-Functional Requirements 8](#_Toc473533342)

[5 Design Constraints 9](#_Toc473533343)

[6 Architecture 10](#_Toc473533344)

[6.1 Architecture Overview 10](#_Toc473533345)

[6.1.1 Component Interfaces 10](#_Toc473533346)

[6.1.2 Component Dependencies 10](#_Toc473533347)

[6.1.3 Component Traceability 10](#_Toc473533348)

[6.1.4 Component Composition 11](#_Toc473533349)

[6.1.5 Component Deployment 11](#_Toc473533350)

[7 Collaboration 12](#_Toc473533351)

[7.1 Use case realization “Finding a location by only typing a search text around my current location in area of approximately of 15 KM away” 12](#_Toc473533352)

[7.2 Use case realization “Finding an address by only typing a street name and scrolling down the result list” 13](#_Toc473533353)

[7.3 Use case realization “Finding a hotel by typing its name and selecting the 'hotel' POI category” 14](#_Toc473533354)

[7.4 Use case realization “Finding a petrol station along my current route and not further than 5 KM from the route” 14](#_Toc473533355)

[7.5 Use case realization “User doesn't know the exact spelling of a hotel (e.g., Cesar hotel) and still wants to find the location by entering a text close to the exact name (e.g., by entering Cedar hotel)” 15](#_Toc473533356)

[8 Interfaces 16](#_Toc473533357)

[8.1 Interface FreeTextSearch 16](#_Toc473533358)

[8.1.1 Franca IDL definition 16](#_Toc473533359)

[8.1.2 D-Bus XML 39](#_Toc473533360)

[9 Programmer’s Manual 43](#_Toc473533361)

[10 Implementation 44](#_Toc473533362)

[11 Test Plan 45](#_Toc473533363)

# 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.

## Subsystem Overview

The Free Text Search Component is part of the Navigation Package as shown in Figure 1.

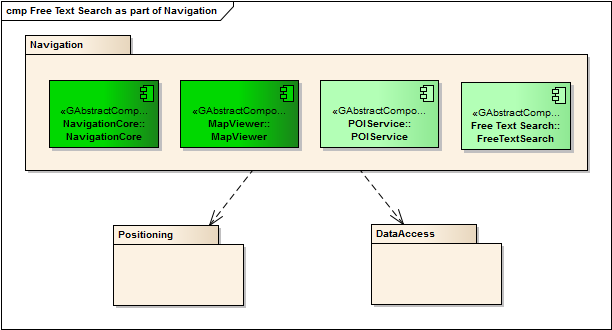


Figure Free Text Search as part of Navigation

Like the POI Service and the LocationInput interface of NavigationCore, the Free Text Search Component provides locations which can e.g. be used as a destination using the Routing interface of NavigationCore.

## Component Overview

Figure 2 shows the context diagram of the component.

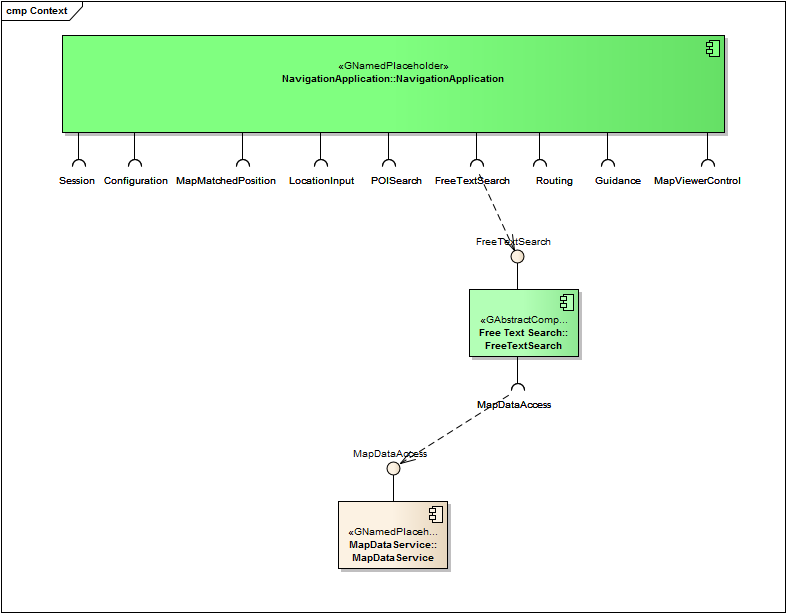


Figure Context diagram of the Free Text Search Component

The Free Text Search component is typically used by a Navigation Application. This Navigation Application uses the FreeTextSearch Interface to let the user search for a location. This location can e.g. be used as destination, waypoint or be shown on the map.

The Free Text Search component depends on the MapDataAccess Interface to obtain the addresses and POIs.

# References

At the time of publication, the referred documents were valid. All standards and specifications are subject to revision, and you are encouraged to investigate the possibility of applying the most recent editions of the standards and specifications indicated below.

1. GENIVI – “GENIVI NavigationCore API” – “3.0.2/30 June 2014” –[http://git.projects.genivi.org/lbs/navigation/doc/navigation-core/NavigationCoreAPI.pdf](http://git.projects.genivi.org/?p=lbs/navigation.git;a=blob_plain;f=doc/navigation-core/NavigationCoreAPI.pdf;hb=HEAD)

# Glossary

|  |  |  |
| --- | --- | --- |
| *Acronym* | *Term* | Definition |
| EA | Enterprise Architect | A proprietary UML tool. EA and UML is widely used in GENIVI interface definitions. |
| FTS | Free Text Search |  |
| UML | Unified Modeling Language |  |
| POI | Point of Interest |  |
|  |  |  |

Table – Acronym and Term Definitions

# Requirements

This chapter first lists the Free Text Search related Use Cases, followed by the requirements which are mainly derived from those Use Cases.

## Use Cases

The Use Case diagram for the Free Text Search Use Cases is shown in Figure 4.



Figure Free Text Search Use Case diagram

The Use Cases are reported in the following format:

|  |  |
| --- | --- |
| **<Use Case ID>** | **<Title>** |
| <Description> | |

The Use Cases show the basic idea of a Free Text Search:

* Independent of the kind of location the user is looking for, he/she just starts typing a search text
* The systems shows the most relevant locations for this text, and options to narrow down the search
* If the location is shown, the user can directly select it.  
  This is Use Case UC-FTS-001 for addresses and Use Case UC-FTS-003 for POIs.
* If the location is not shown, the user can:
  + Start scrolling until the location appears.  
    This is Use Case UC-FTS-002.
  + Add search criteria to narrow down the search by POI category.  
    Use Case UC-FTS-004, where the ‘hotel’ POI category is added as a search criterion.
  + Add search criteria to narrow down the search by its location.  
    Use Case UC-FTS-005, where the specific location is added as a search criterion.
  + Add search criteria to narrow down search to a geographical area.

Use Case UC-FTS-006, where the ‘geometry’ restricts the search area.

* + Add search criteria to narrow down the search along a route.  
    Use Case UC-FTS-007.
  + Add search criteria to narrow down the search along a strip of a route.

Use Case UC-FTS-008, where the corridor width define the strip area around the route.

* + Add search criteria to perform fuzzy match   
    Use Case UC-FTS-009, where the ‘fuzzy search’ can be done.

|  |  |
| --- | --- |
| **UC-FTS-001** | **User finds an address by only typing a street name** |
| The user types the name of a street in a search text box.  The system lists locations from the complete map database (addresses and POIs), which match the entered text, ordered by relevance. The system also lists relevant POI categories for narrowing down the search.  The street that the user is looking for is in the list, so the user selects it. | |

|  |  |
| --- | --- |
| **UC-FTS-002** | **User finds an address by only typing a street name and scrolling down the result list** |
| The user types the name of a street in a search text box.  The system lists locations from the complete map database (addresses and POIs), which match the entered text, ordered by relevance. The system also lists relevant POI categories for narrowing down the search.  The street that the user is looking for is not in the list, so the user scrolls down in the list. After scrolling down a few pages, the street that the user is looking for is in the list, so the user selects it. | |

|  |  |
| --- | --- |
| **UC-FTS-003** | **User finds a hotel by only typing its name** |
| The user types the name of a hotel in a search text box.  The system lists locations from the complete map database (addresses and POIs), which match the entered text, ordered by relevance. The system also lists relevant POI categories for narrowing down the search.  The hotel that the user is looking for is in the list, so the user selects it. | |

|  |  |
| --- | --- |
| **UC-FTS-004** | **User finds a hotel by typing its name and selecting the ‘hotel’ POI category** |
| The user types the name of a hotel in a search text box.  The system lists locations from the complete map database (addresses and POIs), which match the entered text, ordered by relevance. The system also lists relevant POI categories for narrowing down the search.  The hotel that the user is looking for is not in the list, and also many other locations (not being hotels) are shown. So the user selects the ‘hotel’ category to narrow down the search. | |

|  |  |
| --- | --- |
| **UC-FTS-005** | **User can search at a specific location** |
| The user shall be able to search for a free text at a specified location. For example "pizza rosso" around Odeonsplatz. The search engine returns results that can be displayed immediately in a map. The results shall be sorted by relevance | |

|  |  |
| --- | --- |
| **UC-FTS-006** | **User can restrict the search area by specifying a geometry** |
| The user shall be able to restrict the search area by specifying a geometry. Not mandatory, default geometry decided by implementer of the library specified. For search along the route the geometry shall be ignored if set. | |

|  |  |
| --- | --- |
| **UC-FTS-007** | **User can search along a route by typing a free text search** |
| User should be able to search along a route. For example, user wants to find a Shell petrol station along a route by enter keyword "petrol station shell" and specifying a route. | |

|  |  |
| --- | --- |
| **UC-FTS-008** | **User defines the width of the corridor for search along a route** |
| The user shall be able to define the width of the corridor. Not mandatory, default value decided by implementation authors. | |

|  |  |
| --- | --- |
| **UC-FTS-009** | **User can perform fuzzy search** |
| User can choose a fuzzy level for a search. For example, when user searches for "bas station" with a fuzzy level of 1, a result indicating "bus station" can be returned. Search fuzzy level are limited to range 0 to 5. 0 is exact match and 5 is the highest fuzzy level. | |

## Requirements

The requirements are reported in the following format:

|  |  |
| --- | --- |
| **<Requirement ID>** | **<Title>** |
| **<Priority>** | *<Components tracing to requirement>* |
| <Description> | |
| *<Rationale>* | |

The fields in the table are linked to the Requirement model element in EA.

### Functional Requirements

The information in this chapter is provided only for information purpose; this is not a normative part. The interface and behavior description (chapter 8 Interfaces) takes precedence in order to fulfill functional requirements.

|  |  |
| --- | --- |
| **SW-NAV-FTS-001** | **Free Text Search** |
| **P1** | <Components tracing to requirement> |
| It shall be possible to retrieve a list of locations by only providing a search text. The returned locations will be the most relevant locations which match the search text. Text matching can be done on any combination of the properties of the location like e.g. city name, street name, house number, POI name, POI category. The definition of most relevant is up to the implementation. | |
| This functionality is needed to support an HMI with a one-box-search, i.e. where the user can enter a text in a single text field, instead of having to choose for an address or POI first and then either enter country, city, street and house number, or a POI category. | |

|  |  |
| --- | --- |
| **SW-NAV-FTS-002** | **Free Text Search around location** |
| **P1** | <Components tracing to requirement> |
| It shall be possible to specify a search location in combination with a search text. In this case locations closer to the search location are more relevant then locations further away from the search location. | |
| The user is often searching for locations relatively near his location. By using both a search text and a location, a shorter text is needed to find the wanted location. Example: The user is looking for a street in the current city. By setting the search location to his current location, usually just a few characters are needed to find the street. | |

|  |  |
| --- | --- |
| **SW-NAV-FTS-003** | **Free Text Search restricted to specific areas** |
| **P1** | <Components tracing to requirement> |
| It shall be possible to restrict a Free Text Search to specific areas. | |
| If the locations are to be shown on a map, it makes no sense to retrieve location outside of the shown map area. | |

|  |  |
| --- | --- |
| **SW-NAV-FTS-004** | **Free Text Search restricted to one or more POI categories** |
| **P1** | <Components tracing to requirement> |
| It shall be possible to restrict a Free Text Search to one or more POI categories. For this, an initial Free Text Search will provide a list of relevant POI categories. | |
| By using both a search text and one or more POI categories, a shorter text is sufficient to find a POI location. | |

|  |  |
| --- | --- |
| **SW-NAV-FTS-005** | **Free Text Search restricted to one or more countries or states** |
| **P1** | <Components tracing to requirement> |
| It shall be possible to restrict a Free Text Search to one or more countries or states. | |
| By using both a search text and one or more countries,or states, a shorter text is sufficient to find a location. Also, there are often many reasons to look for a (POI) location at a specific side of a border (e.g. passport, currency, insurance restrictions). | |

### Non-Functional Requirements

This is a normative part. These requirements shall be fulfilled by compliant implementations.

There are no non-functional requirements.

# Design Constraints

This is a normative part that includes additional requirements on a compliant implementation outside of the interface and behavior descriptions.

There are no Design Constraints.

# Architecture

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

As the Free Text Search Component is an Abstract Component, this chapter only contains information related to the outside of the component.

## Architecture Overview

The architecture overview is shown in Figure 4.

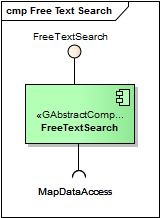


Figure Free Text Search Component

### Component Interfaces

The component provides the following interface:

* FreeTextSearch  
  This interface provides functionality for searching for locations, based on text and optional other criteria to narrow down the search.

### Component Dependencies

The component requires the following interface:

* MapDataAccess  
  MapDataAccess provides access to the actual locations in the map database. The map database shall contain indices to support an efficient search for these locations.

### Component Traceability

Figure 5 shows the requirements traceability, i.e. which requirements (as listed in chapter 4) are implemented by the component.

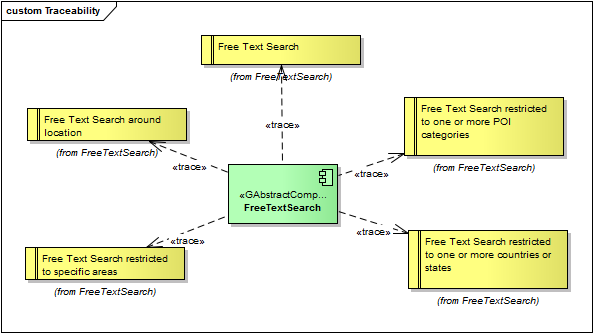


Figure Requirements traceability

### Component Composition

Not applicable as this is an Abstract Component.

### Component Deployment

Not applicable as this is an Abstract Component.

# Collaboration

This is a normative part of the document.

The Use Case realizations are shown in Figure 6.



Figure Free Text Search Use Case realizations

## Use case realization “Finding a location by only typing a search text around my current location in area of approximately of 15 KM away”

The sequence diagram for this Use Case realization is shown in Figure 7.



Figure Finding a location by only typing a search text around my current location in area of approximately of 15 KM away

It’s up to the NavigationApplication to decide when to start the search by invoking ftsRequest. This can e.g. be as soon as the user entered a specific number of characters, or when the user presses a ‘search’ button. A search session identified with ‘seassionHandle’ will be created for each ftsRequest. The ‘pageId’ is the identifier of the returned results.

In this example, there are results for addresses, POIs and POI category suggestions. Note that the order with which these results are provided is not specified. The search engine can even skip sending each of them in case no search result is available. ftsDone is sent to indicate the end of result for the current pageId in the specified search session. ftsCancel should be called to end the current search session.

Each of the returned results has a locationHandle. When such a handle is no longer needed, the application has to free it by calling deleteLocationHandles().

## Use case realization “Finding an address by only typing a street name and scrolling down the result list”

The sequence diagram for this Use Case realization is shown in Figure 8.



Figure Sequence Diagram Scrolling down the result list

How a user can scroll through the list of results is application specific, but typically is per page, or by one or more lines. If it is by page, the pageSize could be equal to the number of locations that can be shown on the screen. If it is by line, the pageSize could be somewhat larger than the number of locations that can be shown on the screen.  
In any case, when the user starts scrolling, at some point more results are needed. So the application call ftsNextPage(). In this example no more POI results are available, so only a subsequent set of addresses are received.

## Use case realization “Finding a hotel by typing its name and selecting the 'hotel' POI category”

The sequence diagram for this Use Case realization is similar to the sequence diagram shown in Figure 7; however with different values for ftsRequest call at the beginning.

The user starts a search with a search text indicating the hotel name and specifying hotel category identifier in ‘searchConditions’ field of ftsRequest. Hotel category ID can be retrieved using POI service. Another way is to search without specifying category ID in ‘searchConditions’ and using the category ID for hotels returned from ftsResultPoiSuggestion and redo the search by specifying the retrieved hotel category ID in search condition.

## Use case realization “Finding a petrol station along my current route and not further than 5 KM from the route”

The sequence diagram for this Use Case realization is similar to the sequence diagram shown in Figure 7; however with different values for ftsRequest call at the beginning.

The user starts a search with a search text indicating the petrol station name (e.g., Shell) and specifying Petrol Station category identifier in ‘searchConditions’ field of ftsRequest. Petrol Station category ID can be retrieved using POI service. The search area can be restricted to an area specified by ‘searchShapes’ field in ftsRequest. Search area can be restricted to an area around route using RouteCorridor in searchShapes. The route handle retrieved from the routing service shall be used to specify the route in the route corridor object with corridorWidth equal to 5000 meter.

## Use case realization “User doesn't know the exact spelling of a hotel (e.g., Cesar hotel) and still wants to find the location by entering a text close to the exact name (e.g., by entering Cedar hotel)”

The sequence diagram for this Use Case realization is similar to the sequence diagram shown in Figure 7; however with different values for ftsRequest call at the beginning.

The user starts a search with a search text indicating the hotel name (e.g., Cedar hotel) and enabling fuzzy search by setting ‘fuzzyLevel’ field in ftsSearch to a non-zero value. The search engine shall perform a fuzzy search and return the closet match to the entered text.

# Interfaces

This is a normative part of the document.

## Interface FreeTextSearch

### Franca IDL definition

The Franca IDL definition is based on Franca 0.9.1.

// SPDX-License-Identifier: MPL-2.0

// Copyright (C) 2015-2016, PCA Peugeot Citroën, XS Embedded GmbH, TomTom International B.V., Continental Automotive GmbH,

// Alpine Electronics R&D Europe GmbH, Harman-Becker Automotive GmbH.

// This Source Code Form is subject to the terms of the

// Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with

// this file, you can obtain one at http://mozilla.org/MPL/2.0/.

package org.genivi.navigation.freetextsearchservice

import org.genivi.CommonTypes.\* from "../../CommonTypes.fidl"

import org.genivi.navigation.NavigationTypes.\* from "../NavigationTypes.fidl"

<\*\*

@description : This interface allows a user to find locations by entering a single text string.

A user of a navigation system needs to find a location on a map, e.g. to use as a destination of a route.

Locations can be identified by different means, for example:

- An address, e.g. '2400 Camino Ramon, San Ramon, California, USA.'

- A point of interest, e.g. 'Eiffel Tower'.

- A named place, .e.g. 'Amsterdam'

The system will respond with locations that match the given text string.

For each location, a location handle will be returned that can be used as input to other interfaces,

for example to plan a route, or to get more attributes of a point of interest.

A free text search is initiated by sending a free text search request (Method FtsRequest) containing the search text and search options input by the user.

Depending on search options both addresses and POIs are searched for matches. A single request may get multiple result responses (via Signals):

- ftsResultAddresses to report address matches

- ftsResultPois to report POI matches

- ftsResultPoiSuggestions to give suggestions for doing specific POI queries

Each response comes as soon as the results are available, so address results may be received while the search process continues looking for POI matches.

The order in which the results will be received is not defined. The search engine determines which results to return first based on the user input.

If there are no matches found in a certain type of result, then no response for that type will be sent.

E.g. if the user input only matches an address but not POIs, then no ftsResultPois response will be sent.

To indicate that the search has finished an ftsDone signal is sent.

\*\*>

interface FreeTextSearch {

version {

major 0

minor 1

}

<\*\*

@description: Contains constant values used in this interface

\*\*>

enumeration FtsRangeConstants {

MAX\_FUZZY\_LEVEL = 5

MAX\_SHAPES = 31

MAX\_STRING\_LIST\_SIZE = 100

MAX\_RESULTS = 500

MAX\_STRING\_LENGTH = 4095

MAX\_LOCATION\_HANDLES\_LIST\_SIZE = 4096

}

<\*\*

@description: Search results are returned in some sequential pages. Each

page is identified with an ID called pageId. The pageId of the first

page is zero and pageIds are incremented by one for each subsequent page.

Value -1 indicates that pageId is not specified.

\*\*>

typedef PageId is Int32

<\*\*

@description: Type to represent most texts in this interface.

Character set = UNICODE

Maximum length = MAX\_STRING\_LENGTH

\*\*>

typedef FtsString is String

<\*\*

@description: A list of strings

maximum length = MAX\_STRING\_LIST\_SIZE

\*\*>

array FtsStringList of FtsString

<\*\*

@description: A route corridor

\*\*>

struct RouteCorridor {

<\*\*

@description: Route handle used to select a route.

Can be 0 to use the route for which guidance is started with startGuidance

from Guidance interface.

\*\*>

Handle routeHandle

<\*\*

@description: Offset on route from the start of the route, in meters.

\*\*>

DistanceInMeters startOffset

<\*\*

@description: Width of the corridor, in meters.

\*\*>

DistanceInMeters corridorWidth

<\*\*

@description: Length of the corridor, in meters.

Can be 0 to search until the end of the route.

\*\*>

DistanceInMeters corridorLength

}

<\*\*

@description: A union to contain any of the possible search shapes.

The search can be limited to the area of a circle or rectangle or to a route corridor

around a planned route.

\*\*>

union Shape {

Circle circle

Rectangle rectangle

RouteCorridor routeCorridor

}

<\*\*

@description: A list of shapes.

maximum length = MAX\_SHAPES

\*\*>

array ShapeList of Shape

<\*\*

@description: A maximum number of results to be returned.

Range[0:MAX\_RESULTS]

\*\*>

typedef PageSize is UInt16

<\*\*

@description: Search options for a search request.

The options are flags which can be OR-ed together to create SearchOptions.

\*\*>

enumeration SearchOption {

<\*\*

@description: Search addresses.

\*\*>

ADDRESS = 1

<\*\*

@description: Search POIs.

\*\*>

POI = 2

<\*\*

@description: Get POI suggestions.

\*\*>

POI\_SUGGESTION = 4

<\*\*

@description: Do not search places when doing an address search.

\*\*>

NO\_PLACES = 65536

<\*\*

@description: Do not search streets when doing an address search.

\*\*>

NO\_STREETS = 131072

<\*\*

@description: Do not make assumptions on POI categories.

\*\*>

NO\_POI\_CATEGORY\_ASSUMPTIONS = 262144

<\*\*

@description: Do not search political divisions when doing an address search.

\*\*>

NO\_POLITICAL\_DIVISIONS = 524288

<\*\*

@description: Indicate POI links.

If this option is set in the query, POI results will be marked appropriately

if they are parents or children.

\*\*>

POI\_INDICATE\_LINKS = 1048576

}

<\*\*

@description: An OR-ed combination of SearchOption flags.

Range[0:0xffffffff]

\*\*>

typedef SearchOptions is UInt32

<\*\*

@description: Search request status.

\*\*>

enumeration FtsStatus {

<\*\*

@description: No problem.

\*\*>

OK = 0

<\*\*

@description: User aborted search.

\*\*>

ABORTED = 1

<\*\*

@description: Search string is too short to evaluate.

\*\*>

PREFIX\_TOO\_SHORT = 2

<\*\*

@description: An internal error occurred in the FTS engine.

\*\*>

INTERNAL\_ERROR = 3

<\*\*

@description: No FTS index for the current map.

\*\*>

INDEX\_MISSING = 4

<\*\*

@description: The FTS index format is not supported.

\*\*>

BAD\_VERSION = 5

<\*\*

@description: A search query not supported by the FTS engine.

\*\*>

INVALID\_PARAMETER\_QUERY = 6

<\*\*

@description: A search location not on this earth.

\*\*>

INVALID\_PARAMETER\_SEARCH\_LOCATION = 7

<\*\*

@description: An invalid page size.

\*\*>

INVALID\_PARAMETER\_PAGE\_SIZE = 8

<\*\*

@description: Invalid search options passed to the FTS engine.

\*\*>

INVALID\_PARAMETER\_SEARCH\_OPTIONS = 9

<\*\*

@description: Invalid search conditions passed to the FTS engine.

\*\*>

INVALID\_PARAMETER\_SEARCH\_CONDITIONS = 10

<\*\*

@description: One or more of the search shapes are invalid.

\*\*>

INVALID\_PARAMETER\_SEARCH\_SHAPES = 11

<\*\*

@description: The fuzzy level is not in the allowed range: [0,MAX\_FUZZY\_LEVEL].

\*\*>

INVALID\_PARAMETER\_FUZZY\_LEVEL = 12

<\*\*

@description: The input consisted of invalid geographic coordinates.

\*\*>

INVALID\_COORDINATES\_INPUT = 13

}

<\*\*

@description: Handle to a location.

Range[0:0xffffffff]

\*\*>

typedef LocationHandle is UInt32

<\*\*

@description: A list of LocationHandles.

maximum length = MAX\_LOCATION\_HANDLES\_LIST\_SIZE

\*\*>

array LocationHandleList of LocationHandle

<\*\*

@description: If a user entered a non-existing house number then the closest existing

number is returned. The houseNumberFromInput will contain the house

number from the user input, the houseNumber will contain closest house number.

If the map does not contain house numbers for the street, but the user

input seems to have a house number, then houseNumber will be empty and

houseNumberFromInput will contain the number from the input.

\*\*>

struct StreetDetails {

FtsString streetName

<\*\*

@description: If a user entered a non-existing house number then the closest existing

number is returned. The houseNumberFromInput will contain the house

number from the user input, the houseNumber will contain closest house number.

If the map does not contain house numbers for the street, but the user

input seems to have a house number, then houseNumber will be empty and

houseNumberFromInput will contain the number from the input.

\*\*>

FtsString houseNumber

<\*\*

@description: The house number that the search engine recognized in the user input.

Can be empty.

\*\*>

FtsString houseNumberFromInput

}

enumeration PlaceType {

<\*\*

@description: a place of unknown type.

\*\*>

OTHER = 0

<\*\*

@description: a center of a settlement.

\*\*>

SETTLEMENT = 1

}

<\*\*

@description: Place details.

\*\*>

struct PlaceDetails {

PlaceType placeType

FtsString placeName

<\*\*

@description: The bounding box of this place (an estimate).

\*\*>

Rectangle placeBoundingBox

}

<\*\*

@description: Mapcode details.

\*\*>

struct MapCodeDetails {

<\*\*

@description: If the location of the mapcode is close to a street, this field

contains the name of that street, otherwise this field is left empty.

\*\*>

FtsString closestStreetName

}

<\*\*

@description: Political division details (country or state).

\*\*>

struct PoliticalDivisionDetails {

<\*\*

@description: The bounding box of the political division.

\*\*>

Rectangle boundingBox

}

<\*\*

@description: Geographic coordinate pair details.

\*\*>

struct CoordinatesDetails

{

<\*\*

@description: This field contains the name of the street that is closest

to the coordinates, as long as the straight line distance to

the closest street location is less than 25 meters.

\*\*>

FtsString closestStreetName

}

<\*\*

@description: A union to contain any of the specific types of address details.

\*\*>

union AddressDetails {

StreetDetails streetDetails

PlaceDetails placeDetails

MapCodeDetails mapCodeDetails

PoliticalDivisionDetails politicalDivisionDetails

CoordinatesDetails coordinatesDetails

}

<\*\*

@description: A higher score means a better match.

Range[0:0xffff]

\*\*>

typedef Score is UInt16

<\*\*

@description: Address information.

\*\*>

struct Address {

LocationHandle locationHandle

FtsString countryCode

<\*\*

@description: Optional state code.

\*\*>

FtsString stateCode

<\*\*

@description: There can be multiple place names.

\*\*>

FtsStringList places

<\*\*

@description: There can be multiple postal codes assigned to a street.

\*\*>

FtsStringList postalCodes

<\*\*

@description: Mapcode of the coordinate.

A mapcode represents a location. Every location on Earth can be represented by a mapcode.

Mapcodes were designed to be short, easy to recognize, remember and communicate.

They are precise to a few meters. For details see http://www.mapcode.com.

\*\*>

FtsString mapCode

AddressDetails addressDetails

Coordinate2D coordinate

<\*\*

@description: Distance in meters from the search location.

-1 if no search location is given.

\*\*>

DistanceInMeters distance

<\*\*

@description: A higher score means a better match.

\*\*>

Score score

<\*\*

@description: Indicates whether this match is fuzzy, i.e. it was matched with misspellings.

\*\*>

Boolean fuzzyMatch

}

<\*\*

@description: A list of addresses.

maximum length = MAX\_RESULTS

\*\*>

array Addresses of Address

<\*\*

@description: Numerical POI category code.

Range[0:0xffffffff]

\*\*>

typedef POICategoryCode is UInt32

<\*\*

@description: POI information.

\*\*>

struct POI {

LocationHandle locationHandle

POICategoryCode categoryCode

FtsStringList brandNames

FtsString poiName

FtsString address

FtsStringList place

FtsStringList postalCode

FtsString mapCode

FtsString countryCode

FtsString stateCode

FtsString telephone

Coordinate2D coordinate

<\*\*

@description: Distance in meters from the search location.

-1 if no search location is given.

\*\*>

DistanceInMeters distance

<\*\*

@description: A higher score means a better match.

\*\*>

Score score

<\*\*

@description: Indicates whether this match is fuzzy, i.e. it was matched with misspellings.

\*\*>

Boolean fuzzyMatch

<\*\*

@description: Indicates that POI has parents.

\*\*>

Boolean hasParents

<\*\*

@description: Indicates that POI has children.

\*\*>

Boolean hasChildren

<\*\*

@description: For searches along route, this value is the the detour distance,

in meters, expresses how much longer the total route will be if the POI is visited.

The detour distance will be -1 for all searches that don’t use a route corridor

(searches in all map, in circle or rectangle).

\*\*>

DistanceInMeters detourDistance

<\*\*

@description: For searches along route, this value is the detour time, in seconds,

expresses what’s the delay at the destination if the POI is visited.

The detour time will be -1 for all searches that don’t use a route corridor

(searches in all map, in circle or rectangle).

\*\*>

Int32 detourTime

}

<\*\*

@description: A list of POIs.

maximum length = MAX\_RESULTS

\*\*>

array POIs of POI

<\*\*

@description: Data type for POI category suggestions.

\*\*>

struct POICategory {

POICategoryCode categoryCode

FtsString categoryName

<\*\*

@description: The search engine recognizes synonyms and names associated with a

POI category. If a match on such a name is found, then this attribute is set to that name.

\*\*>

FtsString matchedName

<\*\*

@description: Text that can be used for a suggestion like: "<categoryName> <searchTextSuggestion>"

Typically this contains the original query text, where the recognized

category name or synonym has been removed.

\*\*>

FtsString searchTextSuggestion

<\*\*

@description: A list of addresses that can be used for suggestions like: "<categoryName> near <address>

Note the addresses in this list do not have a valid location handle

as the address is only needed to be displayed as a suggestion.

The coordinates of an address can be used for a POI window query if the

user picks a suggestion.

\*\*>

Addresses nearAddressSuggestions

<\*\*

@description: A higher score means a better match.

\*\*>

Score score

}

<\*\*

@description: A list of POI categories.

maximum length = MAX\_RESULTS

\*\*>

array POICategories of POICategory

<\*\*

@description: Indicates on a scale from 0 to MAX\_FUZZY\_LEVEL how fuzzy the search results can be.

0 means only an exact search is performed, 1 is the lowest level of fuzzy search

and MAX\_FUZZY\_LEVEL means a very fuzzy search.

Range[0:MAX\_FUZZY\_LEVEL]

\*\*>

typedef FuzzyLevel is Int8

<\*\*

@description : This method returns the API version implemented by the content access module.

\*\*>

method getVersion {

out {

<\*\*

@description: .

\*\*>

Version ^version

}

}

<\*\*

@description: Perform a free text search.

A single search request will trigger zero or one of each ftsResultAddresses,

ftsResultPois or ftsResultPoiSuggestions responses followed by an ftsDone response.

Each response will contain at most one page of results. A next page of results can be

retrieved with the FtsNextPage request.

A ftsRequest call creates a new FTS session identified with a unique sessionHandle. The

FTS session will explicitly be terminated by a ftsCancel call or will implicitly be terminated

by a subsequent ftsRequest call. When a new ftsRequest is sent before receiving a ftsDone response

on the previous request, the previous request will be canceled. A ftsDone response on the previous

request will still be sent. This way each request gets a response.

\*\*>

method ftsRequest {

in {

<\*\*

@description: Text to match.

\*\*>

FtsString inputString

<\*\*

@description: Location for scoring the search results.

A result closer to this location gets a higher score than a result further away.

\*\*>

Coordinate2D searchLocation

<\*\*

@description: Limit the search to the union of several shapes (each can be a circle or a rectangle

or a route corridor around a planned route). Pass an empty list to search the whole map.

\*\*>

ShapeList searchShapes

<\*\*

@description: The maximum number of addresses and POIs that will be returned.

\*\*>

PageSize pageSize

<\*\*

@description: Options to indicate what to search for.

One or more values of SearchOption can be added together.

\*\*>

SearchOptions searchOptions

<\*\*

@description: Additional search conditions, space separated.

The string can be empty when the POI search is not limited by categories

and not limited by countries/states.

Supported conditions:

"category:nnnn" Limit the POI search to a one or more POI categories,

where 'nnnn' is a comma-separated list of numerical POI category ids.

"country:aaaaa" Limit the search to one or more countries (or states),

where 'aaaaa' is a comma-separated list of ISO-3166-1 alpha 3

country code concatenated with an optional ISO-3166-2 state code.

Examples:

"category:9373,9374": limits the POI search to the union of categories 9373 and 9374.

"country:USACA": limits the search to the state of California.

"category:6315 country:USACA,MEX": limits the POI search to category 6315

and limits the search to the union of California state and Mexico.

\*\*>

FtsString searchConditions

<\*\*

@description: Indication of how fuzzy the search shall be performed.

\*\*>

FuzzyLevel fuzzyLevel

}

out {

<\*\*

@description: A new FTS session is created with session handle equal to sessionHandle.

\*\*>

Handle sessionHandle

<\*\*

@description: The pageId of the search result to be returned (zero for the first page).

\*\*>

PageId pageId

}

error {

<\*\*

@description: Indicates no error.

\*\*>

OK

<\*\*

@description: Indicates that no new FTS session can be created at the moment.

\*\*>

NoMoreSessionHandles

}

}

<\*\*

@description: Get next result page for the specified FTS session.

This request will trigger zero or one of each ftsResultAddresses, ftsResultPois

or ftsResultPoiSuggestions responses followed by an ftsDone response.

If no error is reported, ftsDone will be sent.

\*\*>

method ftsNextPage {

in {

<\*\*

@description: Specifies the session.

\*\*>

Handle sessionHandle

<\*\*

@description: Options to indicate what to search for.

One or more values of SearchOption can be added together.

You can only get results for options that have been passed to the initial FtsRequest.

\*\*>

SearchOptions searchOptions

}

out {

<\*\*

@description: The pageId of the search result to be returned.

\*\*>

PageId pageId

}

error {

<\*\*

@description: Indicates no error.

\*\*>

OK

<\*\*

@description: Indicates that a search request is already in progress. ftsDone has not yet been sent for the previous request.

\*\*>

AlreadyInProgress

<\*\*

@description: Indicates that no more search results are available for the given search session handle.

\*\*>

NoMoreSearchResultsAvailable

<\*\*

@description: Indicates that sessionHandle is unknown.

\*\*>

UnknownSessionHandle

}

}

<\*\*

@description: Cancel an FTS session.

\*\*>

method ftsCancel {

in {

<\*\*

@description: Handle to identify the Free Text Search session to be cancelled.

\*\*>

Handle sessionHandle

}

error {

<\*\*

@description: Indicates no error.

\*\*>

OK

<\*\*

@description: Indicates that sessionHandle is unknown.

\*\*>

UnknownSessionHandle

}

}

<\*\*

@description : Response indicating the search status.

\*\*>

broadcast ftsDone selective {

out {

<\*\*

@description: Specifies the session.

\*\*>

Handle sessionHandle

<\*\*

@description: Specifies the page for which this response is sent.

\*\*>

PageId pageId

<\*\*

@description: Status of the search request.

\*\*>

FtsStatus ftsStatus

}

}

<\*\*

@description : Response with address results.

\*\*>

broadcast ftsResultAddresses selective {

out {

<\*\*

@description: Specifies the session.

\*\*>

Handle sessionHandle

<\*\*

@description: Specifies the page for which this response is sent.

\*\*>

PageId pageId

<\*\*

@description: Matching addresses.

The order of the addresses is the order of matching relevance.

The first address is a better match than the last result.

\*\*>

Addresses addresses

<\*\*

@description: Flag to indicate if more addresses might be available in a next page.

\*\*>

Boolean moreAvailable

}

}

<\*\*

@description : Response with POI results.

\*\*>

broadcast ftsResultPois selective {

out {

<\*\*

@description: Specifies the session.

\*\*>

Handle sessionHandle

<\*\*

@description: Specifies the page for which this response is sent.

\*\*>

PageId pageId

<\*\*

@description: Matching POIs.

The order of the POIs is the order of matching relevance.

The first POI is a better match than the last result.

\*\*>

POIs pois

<\*\*

@description: Flag to indicate if more POIs might be available in a next page.

\*\*>

Boolean moreAvailable

}

}

<\*\*

@description : Response with POI category suggestions.

\*\*>

broadcast ftsResultPoiSuggestions selective {

out {

<\*\*

@description: Specifies the session.

\*\*>

Handle sessionHandle

<\*\*

@description: Specifies the page for which this response is sent.

\*\*>

PageId pageId

<\*\*

@description: POI category suggestions.

\*\*>

POICategories poiCategories

}

}

<\*\*

@description : Release handle(s) if they are not used anymore.

\*\*>

method deleteLocationHandles {

in {

<\*\*

@description: list of handles to release.

\*\*>

LocationHandleList locationHandleList

}

}

}

### D-Bus XML

This D-Bus XML interface definition is generated from the Franca IDL definition with the use of CommonAPI version 3.1.3.

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="introspect.xsl"?>

<!-- SPDX-License-Identifier: MPL-2.0

Copyright (C) 2016, 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., Neusoft Technology Solutions GmbH, Jaguar Land Rover Limited, Visteon Corporation

This Source Code Form is subject to the terms of the

Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with

this file, you can obtain one at http://mozilla.org/MPL/2.0/.

-->

<node xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="/org/genivi/navigation/freetextsearchservice/FreeTextSearch" xsi:noNamespaceSchemaLocation="introspect.xsd" >

<interface name="org.genivi.navigation.freetextsearchservice.FreeTextSearch">

<method name="getInterfaceVersion">

<arg name="value" type="uu" direction="out" />

</method>

<signal name="ftsDone">

<arg name="sessionHandle" type="u" direction="out" />

<arg name="pageId" type="i" direction="out" />

<arg name="ftsStatus" type="i" direction="out" />

</signal>

<method name="subscribeForftsDoneSelective">

<arg direction="out" name="status" type="b">

</arg>

</method>

<method name="unsubscribeFromftsDoneSelective">

</method>

<signal name="ftsResultAddresses">

<arg name="sessionHandle" type="u" direction="out" />

<arg name="pageId" type="i" direction="out" />

<arg name="addresses" type="a(ussasass(yv)(dd)iqb)" direction="out" />

<arg name="moreAvailable" type="b" direction="out" />

</signal>

<method name="subscribeForftsResultAddressesSelective">

<arg direction="out" name="status" type="b">

</arg>

</method>

<method name="unsubscribeFromftsResultAddressesSelective">

</method>

<signal name="ftsResultPois">

<arg name="sessionHandle" type="u" direction="out" />

<arg name="pageId" type="i" direction="out" />

<arg name="pois" type="a(uuasssasasssss(dd)iqbbbii)" direction="out" />

<arg name="moreAvailable" type="b" direction="out" />

</signal>

<method name="subscribeForftsResultPoisSelective">

<arg direction="out" name="status" type="b">

</arg>

</method>

<method name="unsubscribeFromftsResultPoisSelective">

</method>

<signal name="ftsResultPoiSuggestions">

<arg name="sessionHandle" type="u" direction="out" />

<arg name="pageId" type="i" direction="out" />

<arg name="poiCategories" type="a(usssa(ussasass(yv)(dd)iqb)q)" direction="out" />

</signal>

<method name="subscribeForftsResultPoiSuggestionsSelective">

<arg direction="out" name="status" type="b">

</arg>

</method>

<method name="unsubscribeFromftsResultPoiSuggestionsSelective">

</method>

<method name="getVersion">

<arg name="version" type="(qqqs)" direction="out" />

</method>

<method name="ftsRequest">

<arg name="inputString" type="s" direction="in" />

<arg name="searchLocation" type="(dd)" direction="in" />

<arg name="searchShapes" type="a(yv)" direction="in" />

<arg name="pageSize" type="q" direction="in" />

<arg name="searchOptions" type="u" direction="in" />

<arg name="searchConditions" type="s" direction="in" />

<arg name="fuzzyLevel" type="y" direction="in" />

<arg name="error" type="i" direction="out" />

<arg name="sessionHandle" type="u" direction="out" />

<arg name="pageId" type="i" direction="out" />

</method>

<method name="ftsNextPage">

<arg name="sessionHandle" type="u" direction="in" />

<arg name="searchOptions" type="u" direction="in" />

<arg name="error" type="i" direction="out" />

<arg name="pageId" type="i" direction="out" />

</method>

<method name="ftsCancel">

<arg name="sessionHandle" type="u" direction="in" />

<arg name="error" type="i" direction="out" />

</method>

<method name="deleteLocationHandles">

<arg name="locationHandleList" type="au" direction="in" />

</method>

</interface>

</node>

# Programmer’s Manual

This is an optional non-normative part providing examples of how to program against the interfaces

# Implementation

This section describes the implementation of the **proof-of-concept** for the FreeTextSearch component.

The following implementation was used for proving the specification. Note that updated versions of this software may exist at <https://github.com/GENIVI/navigation/tree/master/src/freetextsearch-service>

# Test Plan

A test application is provided in <https://github.com/GENIVI/navigation/tree/master/test/freetextsearch-service>

The test application implements a FreeTextSearch client. It exercises all the methods and registers all the callback functions to handle broadcasts coming from FreeTextSearch service.