Navigation APIs:
From native to Web with Franca and CommonAPI

October 20, 2016  |  AMM Burlingame - All Members

Philippe COLLIOT, LBS-EG Lead
PSA Group

GENIVI is a registered trademark of the GENIVI Alliance in the USA and other countries. Copyright © GENIVI Alliance 2016.
Purpose of the presentation

• To report on the Franca and Common API migration of the LBS-EG interfaces
• To give a status of the Navigation Web APIs
• To discuss about how to achieve the goal and to concretely complete the Web API
Reminder: goals

Official API definition in the GENIVI compliance

DBus .xml

Franca .fidl

Official API definition in the GENIVI compliance

generator

DBus xml
C
C++
Web idl
JS

Today

Tomorrow
Statement of work

- **05/16**: Migration started (with draft Franca files provided by EB)
- **08/16**: POC update started *
- **10/16**: Release Candidate
  - POC updated
  - FSA under debug
- **12/16**: ‘Hand written’
  - Navigation Routing
  - Web API
- **??/17**: Franca generator

* As much as possible, the former DBus definition is aligned and the POC updated accordingly
Overview of the LBS APIs
Portfolio, available documentation…
Portfolio of interfaces

- **navigation-core**
- **map-viewer**
- **poi-service**

**EnhancedPosition**
**Persistence**
**PositionFeedback**

**NavigationCore::NavigationCore**
**Configuration**
**EnhancedPosition**
**Guidance**
**LocationInput**
**MapMatchedPosition**

**MapViewer::MapViewer**
**Configuration**
**EnhancedPosition**
**MapViewerControl**
**Persistence**
**Session**
**SessionClient**

**POIService::POIService**
**POIContentAccess**
**POIContentAccessModule**
**POISearch**
**Session**
**SessionClient**
**Settings**

**EnhancedPositionService::EnhancedPositionService**
**Configuration**
**EnhancedPosition**
**PositionFeedback**

**GNSSService::GNSSService**
**GNSS**

**SensorsService::SensorsService**
**Sensors**

**TrafficFlowService::TrafficFlowService**
**TrafficFlow**
**TrafficFlowListener**
**TrafficFlowUpdates**

**SpeechServices::SpeechServices**
**SpeechInputService**
**IObserves**
**SpeechOutputService**
**IObserves**
**ISpeechOutputService**
**SpeechDialogService**

**DataAccess**
**MapDataService**

**Positioning**
**EnhancedPositionService**
**GNSSService**
**SensorsService**

**TrafficInformation**
**TrafficFlowService**
**TrafficFlowListener**
**TrafficFlowUpdates**

**FreetextSearch**
**SpeechOutput**

---

available in Miranda
planned for Nostromo
Available documentation

IVI Navigation Web portal
https://at.projects.genivi.org/wiki/display/NAV/IVI+Navigation+Home

APIs, documentation and code of proof of concepts in GitHub

Change request management by Jira issues

Use cases and requirements into an UML model

Copyright © GENIVI Alliance 2016 | October 18, 2016 |
Interface release and proof of concept

https://github.com/GENIVI/navigation-application
POC for navigation: client
Fuel Stop Advisor application

https://github.com/GENIVI/navigation
POC for navigation: server & test script

https://github.com/GENIVI/positioning
POC for positioning: server & test script

POC for POI search: server & client
POC for Traffic Incident: server & client
POC for Speech Output: server & client
Migration to Franca and CommonAPI, a feedback

Naming constraints, reserved keywords …
Versions

• Current versions used are:
  – Franca 0.9.1
  – Common API C++ and C++ DBus 3.1.5 v2
  – DBus patched 1.8.14

• Plan to move to the latest ones after the AMM
Overview of faced issues

- At API definition level:
  - Differences in naming conventions (camel case with capital letter first vs small letter first)
  - Propagation of comments
- At Common API implementation level (for DBus generation):
  - Refinement of the directory tree
  - One interface per file
  - Reserved names for Common API default methods
Directory tree

Example of LocationInput.fidl

```fidl
package org.genivi.navigation.navigationcore

import org.genivi.CommonTypes.* from "../../CommonTypes.fidl"
import org.genivi.navigation.NavigationTypes.* from "NavigationTypes.fidl"
import org.genivi.navigation.navigationcore.NavigationCoreTypes.* from "NavigationCoreTypes.fidl"

/**
 * @description : LocationInput - This interface offers functions that implement the location-input
 ***/

interface LocationInput {
    version {
        major 4
        minor 0
    }
}
```
Reserved names

- In Common API:
  - ‘client’ is a reserved word, so it's not possible to use it for parameters, you got an error because of a conflicting declaration
  - For method names, the following ones are not allowed:
    - getAddress, isAvailable, isAvailableBlocking, getProxyStatusEvent, getInterfaceVersionAttribute

- Franca has also reserved names (see documentation)
- 'attributes' is a keyword in c++ --> result of Xtext check
Variant management in Common API DBus

```c
enum WaypointElementType {
    LATITUDE = 168,
    LONGITUDE = 161,
    ALTITUDE = 162,
    LOCATION_INPUT = 17,
    WAYPOINT_TYPE = 289,
};

union WaypointItem {
    Double coordinateValue
    WaypointType waypointValue
    Uint8[] metaData
}

map WayPoint {
    WaypointElementType to WaypointItem
}
```

```c
typedef CommonAPI::Variant<double, WaypointType, std::vector<uint8_t>> WaypointItem;
```

```c
template<class Visitor, class Variant>
struct ApplyVoidVisitor<Visitor, Variant> {
    static const uint8_t index = 0;
    static void visit(Visitor& Variant) {
        assert(false);
    }
    static void visit(Visitor& const Variant&) {
        assert(false);
    }
};
```
Web APIs, expected deliveries
Data types limitations, method call, Web IDL …
First investigation: node.js based POC

- First feedback: key point is to manage complex data types of the GENIVI API
  - JSON encapsulation to serialize data could be a good solution

Example of complex data type: Dbus signature extract

```xml
<method name="GetCategoriesDetails">
  <doc>
  <line>GetCategoriesDetails = This method retrieves the details associated to one or more POI categories.</line>
  <line>It contains the name, the parent categories, the top level attribute, the list of attributes, the icons, ... .</line>
  </doc>
  <arg name="categories" type="au" direction="in">
    <doc>
    <line>list of categories = enum(INVALID,ALL_CATEGORIES,AIRPORT,RESTAURANT,HOTEL,GAZ_STATION,CAR_PARK, ...)</line>
    <line>Note: A POI category is a unique ID. It could be a predefined category or a custom one defined by a POI plug-in.</line>
    </doc>
  </arg>
  <arg name="results" type="a((uau(yv)sbs(yv))a(usia(is(yv)))a(us))" direction="out">
    <doc>
    <line>results = array[details, attributeList, sortOptions]</line>
    </doc>
  </arg>
</method>
```
Franca to generate Web stuff

- HTML
- JS API
- JS binding
- Web socket binding
- CommonAPI C++
- Server C++ code
- fidl
- Franca generator