GENIVI Alliance

Reference Architecture

29 October 2015

Sponsored by:
GENIVI Alliance

Abstract:
This document is a report on the GENIVI reference architecture and its components.

Keywords: GENIVI

GENIVI Alliance, Inc.
2400 Camino Ramon, Suite 375
San Ramon, CA 94583, USA

Acknowledgement

The document has been created by Gunnar Andersson, Lead Architect, GENIVI Alliance Infotainment, Volvo Car Corporation.

Table of Contents

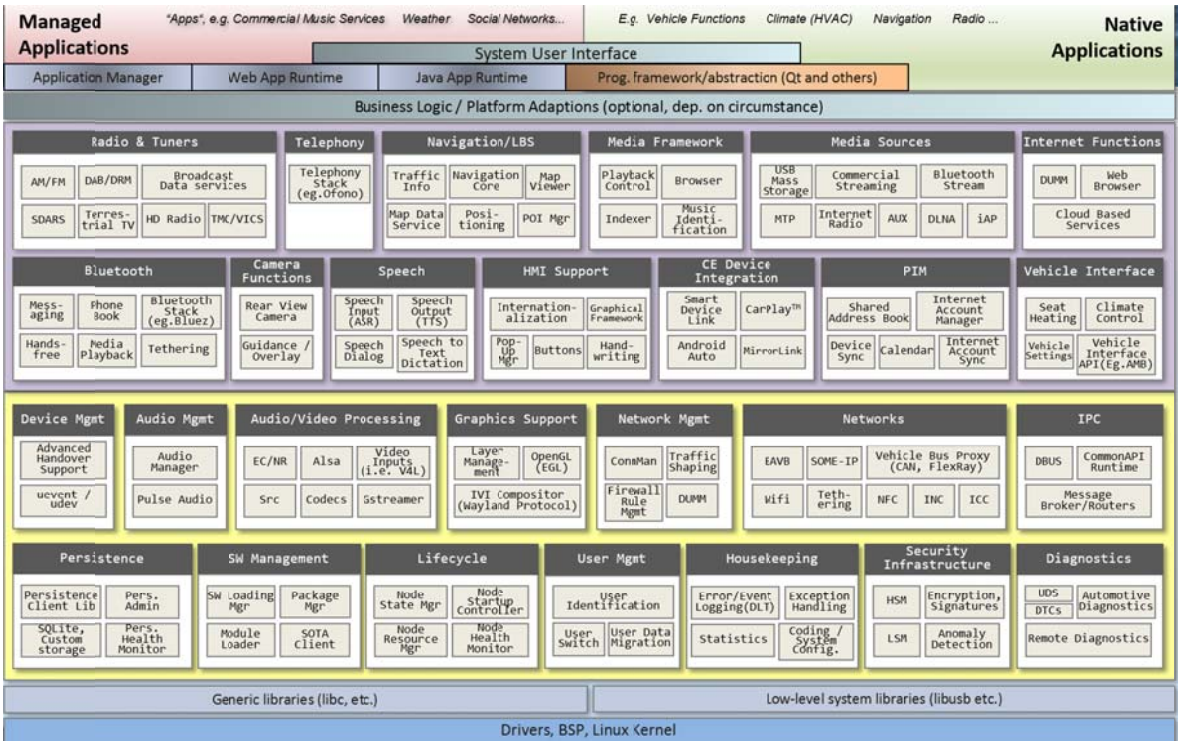
1 Reference Architecture

GENIVI builds its reusable, IVI platform based on a flexible but standard Reference Architecture. This Reference Architecture serves as a blueprint for building a full IVI solution and it also shows the primary target functional areas of GENIVI's work.

This common description allows members to use trusted open source components when building an Infotainment solution. The architecture also helps to avoid fragmentation on noncompetitive areas of the software design.

2 Architecture overview

GENIVI does not deliver a given design but a tool box that will let you make choices. In order to have a common understanding of the pieces in the box with a high level view of their role, a reference Block Diagram has been drawn.



GENIVI focus is on the “middleware” represented here by the yellow and the purple layers.

The yellow layer deals with key infrastructure domains. The smaller blocks map to individual Software components, that are available in the Open Source Community, some of them being developed in GENIVI Open Source Projects.

In the purple layer above, the smaller blocks represent functional domains that may need several Software components addressed as part of the Software Components needed may be commercial Software components.

3 One step deeper in architecture

This section provides a brief introduction to each area (domain) included in the scope of the reference architecture block diagram

The domain areas are defined mostly per *function* area.

3.1 Persistence

The Persistence Management subsystem is responsible for providing a shared solution to persistent data storage. Persistent data includes all data that is dynamically changed but needs to be stored on a head unit between restarts. It includes for example the state of user parameters when the system shuts down (what media is on, sound level and balance, historical data on destinations and phone calls). All types of dynamically changing but persistently stored data are expected to use the persistence subsystem. It does however not deal with software code, navigation map data and similar databases, which would typically be handled by Software Management.

3.2 Software Management

Software Management manages the storage of all program code running on an ECU including how to download and update programs, via standardized automotive diagnostic protocols, from storage devices or over a network, e.g., Software-over-the-air (SOTA).

A software management implementation updates different parts of the system and handles issues like rolling back unsuccessful updates to a known stable state. All the required steps like verifying software integrity/authenticity, system compatibility and similar issues need to be taken into account.

An Infotainment ECU frequently contains many different types of hardware, such as individual radio tuners, Bluetooth modules and other peripherals, some of which in turn need to be loaded with their own software or firmware using special protocols.

3.3 Lifecycle

Lifecycle deals with the startup, shutdown and management of the running of system components and integrated applications. It also keeps track of the state, and the status (health) of the system, including:

- Power-, Node state-, Boot- and Resource management.
- Interfaces to/of Thermal- and Supply management.

3.4 User Management

User Management is the support for multiple independent users, each having their own profiles and settings, and the mechanisms for keeping track of who is active (logged in) on different parts of the system. In combination with other subsystems it also cares for protection of personal data from unwanted disclosure in its interface to Personal Information Management subsystem.

The reference architecture describes some technical approaches for creating a multiuser system and solutions for storage and separation of private data and how to use GENIVI provided components to achieve this.

3.5 Housekeeping

This domain groups typical software modules that monitor software behavior and handle errors.

It includes Automotive Diagnostics Log and Trace (DLT) which is one of the earliest GENIVI software. It is an advanced log and trace framework that provides interfaces for applications to log messages at different levels from debug, info, to critical errors. In comparison to Linux mechanisms used in server and desktop systems like syslog / journal, DLT differs primarily by being highly optimized for embedded systems, primarily designed (not as secondary addition) for moving log messages out over a network to another node. Most of all however, DLT follows an automotive logging standard from AUTOSAR which makes it appropriate to integrate in a larger context of automotive systems and AUTOSAR tooling. Combinations with other Linux technologies are of course still possible. DLT is often used in GENIVI components for logging / tracing and is a much appreciated development tool.

3.6 Security Infrastructure

This subsystems groups together all cryptography libraries and other support functions for security solutions, abstractions and libraries for interacting with Hardware Security Modules (HSM), the management of signatures and encryption, intrusion/anomaly detection, and a Mandatory Access Control solution implemented by a Linux Security Module (LSM).

3.7 Diagnostics

This domain implements automotive diagnostics features. Unified Diagnostic Services (UDS) are codified in ISO 14229-1:2013 and allow diagnostics to control functions on an in-vehicle Electronic Control Unit (ECU). Standard onboard Diagnostic Trouble Codes (DTCs) are used.

3.8 Inter Process Communication (IPC)

The domain includes standards for internal communication such as d-bus, any additional infrastructure to support communication (such as a message broker, if used). It also includes the Common API Runtime. The domain includes Common API Runtime. CommonAPI is an *Inter Process Communication* language binding API that enables applications to use different IPC middleware as backend without any changes to the application code. The current primary implementation is C++ based although a C-based variant is being developed. CommonAPI is tightly connected to Franca IDL - tools require Franca IDL described input and the communication semantics must match. To read more about Franca, see the project website.

The currently well-tested communication backend for CommonAPI C++ is D-Bus while some companies make their own back ends. Future development may support kdbus as an alternative.

3.9 Networks

The Infotainment system often plays the role of a hub in a vehicle as it is connected to vehicle network on one side and to external resources on the other side (smart devices, cloud, etc.). A large set of network technologies and protocols are envisioned (CAN, Flexray, USB, Wi-Fi, NFC, AVB, etc.).

A SOME/IP communication stack and connection to CommonAPI is available. Whatever the type of IVI system we expect there is always some connection to a vehicle network. At the maximum the IVI system implements a user interface for lots of vehicle related functions including climate, seats, turning on and off driving features, powertrain settings and many other domains. At the very minimum, the power states (sleep, wakeup, etc.) of the ECU are typically controlled by another node in the electrical system.

The Networks subsystem is intended to group the implementations of different network technologies. Note however that Bluetooth is treated in its own subsystem. The basic implementation of Ethernet AVB support also resides here, while it is used by Audio Management and occasionally video subsystems.

3.10 Network Management

Network Management includes the infrastructure required to set up, take down, monitor and control network connections. Connection management (using connman), and helper libraries like Download Upload Messaging Manager (DUMM) are in this domain. Depending on the system, things like traffic shaping, firewalling and similar features would also be considered part of this subsystem.

3.11 Graphics Support

This includes graphical support libraries for rendering and compositing. Examples are the IVI Layer Manager, Compositor implementations based on Wayland protocol, IVI extension to Wayland, OpenGL and similar low-level graphics standards. Prominent image processing libraries could be put explicitly into this category, but just as often they may be just considered part of the "generic libraries" section of the platform.

3.12 Audio/Video Processing

This subsystem includes the codecs and infrastructure for putting together a processing pipeline for audio or video. Typically this means audio and video codec implementations, or interface libraries to such implementations if they exist in specialized hardware, optimized DSP code or similar. GStreamer provides pipeline/processing support.

3.13 Audio Management

The Audio domain deals with management, connection and routing of audio streams taking into account unique automotive (IVI) audio requirements (priorities, mixing, foreground/background, etc.). The primary implementation is the GENIVI AudioManager framework which consists of a shared implementation of the AudioManagerDaemon plus typically a set of product specific implementations of the plug-in abstract components. In this group we typically also include some available Linux technologies for the management of audio (pulseaudio, alsa).

3.14 Device Management

Low level support function reacting to and distributing events for connected devices ranging from simple USB memories up to connected Bluetooth devices. An interesting challenge to implement in this subsystem is preparation of the higher level logic that may be required to react to device connections, such as handing over from one connection (wireless) to another (physical) originating from the same device. A typical implementation may require product unique logic but the platform solution should consider how general patterns can be provided to allow reuse between projects.

3.15 Bluetooth

Typically a full-featured Bluetooth Stack is expected in a modern IVI system to support use cases such as data exchange, networking, media streaming and hands-free telephony. Because of its importance and its many parts and features Bluetooth software components are in their own separate subsystem.

3.16 Camera Functions

This subsystem contains code that implements, or supports implementing, camera based functions such as rear-view (reverse gear) camera, camera assisted parking, surround-views, etc.

3.17 Speech

Different components, from the basic technology libraries to databases, and dialog-control, and the necessary connections to other HMI functions make up a working speech subsystem.

3.18 HMI Support

HMI (Human Machine Interface) includes all technologies that the user interacts with. In addition to graphical displays, this includes buttons and knobs, speech control, gestures and handwriting recognition. A subsystem is created to include implementation of button routing, recognizers for gestures/handwriting, prioritization of HMI events, etc.

There are also typically more elaborate supporting frameworks used for graphical user interfaces above the low level features provided by Graphics Support subsystem. These may include frameworks helping the implementation of GUIs (Qt, GTK, etc.) It may differ from one GENIVI system to the next which, if any, of such libraries should logically be considered part of the platform or be seen as added on top of it. This is why the reference block diagram also suggests that technologies like Qt can if desired be considered to exist outside of platform specification.

3.19 CE Device Integration

This deals with the interaction between smart devices and the IVI system. Some different standards include SmartDeviceLink, MirrorLink™, CarPlay™ and Android Auto.

3.20 Personal Information Management (PIM)

In the context of an IVI system, PIM keeps a database of personal information about other people, such as an address book, including also phone numbers, email address, social application identities, etc. It also includes the user's own information

of similar kind (Web service accounts, passwords, etc.) and therefore interfaces with the user management domain.

Although some address books may be fetched or synced with Bluetooth, others may have an Internet account management elsewhere. It's assumed that many features may need access to PIM information, for example supporting a use case like navigating to the address of a contact. Designing platform components responsible for PIM enables such information to be shared and available to multiple applications instead of locked into individual ones.

In some systems, PIM extends to calendar information, appointments, todo-tasks. This tends to be heavily integrated with Email, which slides over to mobile office and even social networks. Typically we group those user functions into the "Internet Functions" group, but significant interaction with PIM is likely and different products may do it slightly differently depending on required integration.

3.21 Vehicle Interface

This subsystem deals with any abstraction or connection to typical automotive vehicle networks and to the signals and information handled there. In some systems this may include connection to a peripheral CPU, which in turn is actually connected to vehicle buses. In practice designing all the software for a function together of course makes sense even if different parts are deployed on different CPUs - nonetheless it is worthwhile to remind that the GENIVI reference architecture deals mainly with the main "multimedia" SoC, if the IVI system is a multi-CPU solution.

3.22 Internet Functions

This domain includes a traditional interactive Web browser function and noninteractive renderers for HTML documents, as well interfaces to cloud-based services. Very often frequently changing user features will be implemented as "apps" on top of a platform, but in any system where these are considered integrated into the platform, then features like email, social networks etc. may also be included in Internet Functions subsystem.

3.23 Media Sources

This platform subsystem includes the actual implementations that access media data from different sources, typically presenting their result in a uniform way to the Media Framework.

Network streams and connected devices (DLNA®, Bluetooth, Internet streaming), as well as other removable media and embedded storage are typically included, but broadcast technology such as AM/FM, SiriusXM, DAB, Television broadcasts, are grouped into another subsystem.

3.24 Media Framework

This is a platform support for media applications which includes the generic logic that can be shared between Media players and is extended by the sources provided by the Media Sources subsystem.

A media framework supports media applications with the ability to connect and integrate different sources of media as a combined resource.

In addition to playback logic, primary features include maintenance of music metadata, music identification services and coordination of media playback.

3.25 Navigation & Location Based Services (LBS)

This domain deals with traditional IVI navigation systems and the supporting technology for these such as positioning

It also includes all data services and innovative features that have location as their primary feature.

3.26 Telephony

This is the telephony software stack. It complements the Bluetooth communication (typically) with the actual logic for connecting and managing phone calls, call-waiting, conferences and so on. There may be multiple technologies behind such as an in-car embedded phone, or VOIP functionality and the telephony stack could (if desired) manage them similarly.

3.27 Radio & Tuners

This domain includes traditional broadcast technologies including receivers for radio technologies (AM/FM, DAB, High-Definition Radio, SiriusXM, etc.) as well as television broadcasts. Note that streaming and the "Internet radio" type of sources are addressed in the Media Sources domain. The functionality of the Tuner API is verified in the IVI Radio project.

3.28 Applications Layer

GENIVI makes the distinction between Native Applications like Vehicle Functions, Climate, Radio, Navigation, etc. and Managed Applications that may be added by the user like Music Services, Weather, Social Networks, etc.

The exact features implemented in the Native and Managed domain respectively are decided by the system builder - GENIVI has only provided some typical examples. In fact, it's optional to use the Managed or Native or (recommended) combining both approaches.