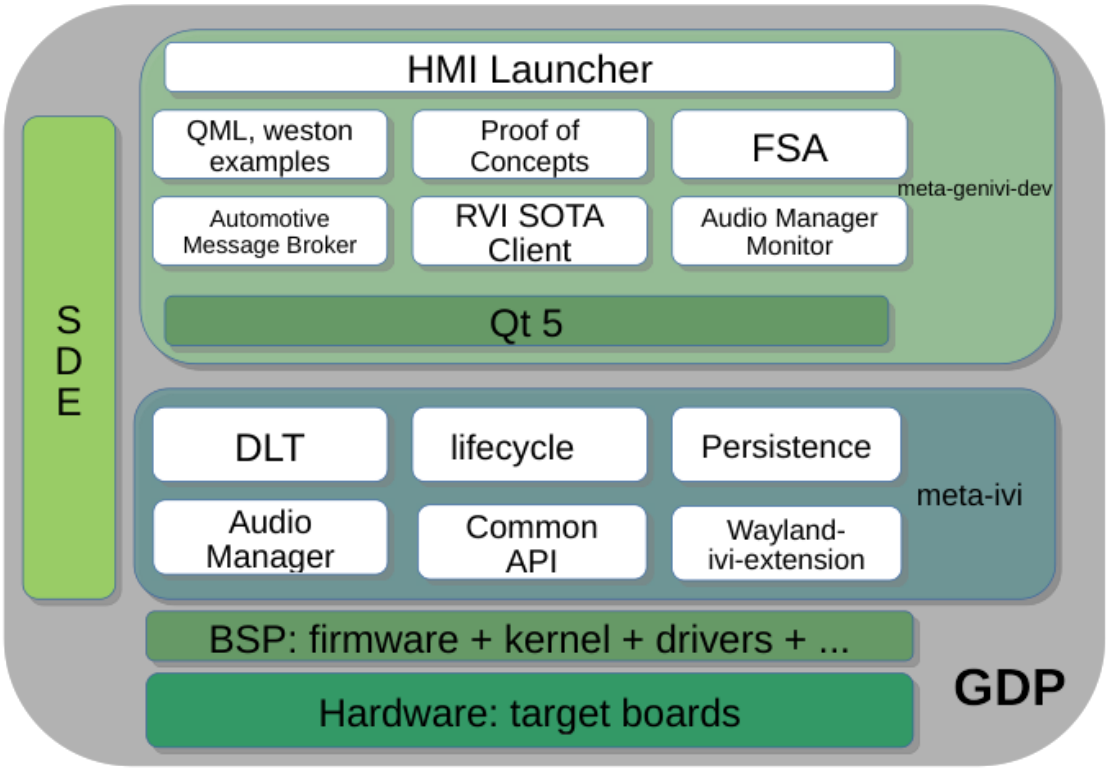
GDP in detail.

This page provides a more detailed information about what is GDP and how to use it. It also includes links to tutorials and Howtos together with further information about other initiatives within GENIVI related with GDP.

Block Diagram and description

This is the high level block diagram of GDP Master



The GENIVI Development Platform (GDP) includes, but is not limited to, the following list of components:

- Hardware: target boards:
  - GDP ship ports to a variety of target boards for each release. The number of boards expand based on contributions. You can find further information in the GDP releases wiki page
  - BSP layer: each target board requires a BSP layer with specific kernel, drivers and configurations to make the board work properly.

- Meta-ivi is a GENIVI project, the base layer for GDP. Find more info about meta-ivi in this wiki. Meta-ivi has the following main modules /components:
  - Diagnostic Log and Trace: Project Webpage / git
  - Persistence: Project Webpage
    - Administrator: git
    - Client Library: git
    - DB config tool: git
    - Health monitor: git
    - Common object: git
  - Lifecycle (not integrated in GDP yet).
    - Node State Manager: git
    - Node Startup Controller: git
    - Node Health Monitor: git

        - Wayland-ivi-extension: documentation / git
        - Audio Manager: documentation / git
        - Common API C++: documentation / git

- Meta Qt5: project WebPage / git

- meta-genivi-dev layer is formed by those components present in GDP on top of meta-ivi. The main ones are:
    - Automotive Message Broker: Documentation / git
    - RVI SOTA Client: documentation and git repository
    - Audio Manager Monitor: GDP Audio Manager Monitor / git
    - Fuel Stop Adviser (FSA): Project webpage / git
    - QML and weston examples
    - Proof of concepts:
        - Browser: Project webpage / git
        - FM Radio git
        - Climate Control git
        - Connected Home git
    - HMI launcher: user Interface documentation / git
    - Location-based Services
        - Positioning: git
        - Navigation: git

- meta-rvi is a layer that add the Remote Vehicle Interaction (RVI) to GDP. RVI provides a unified framework for remote interaction from devices and the cloud with a vehicle through various communication channels that are handled transparently. It's infrastructure is the foundation for applications falling into the three macro use cases for remote access to vehicles over networks:
    - Vehicle Control - retrieve vehicle status information and carry out control actions
    - SOTA - updating vehicle software, configuration data, maps, etc. over the air
    - Big Data - collecting, filtering, preprocessing and transmitting vehicle data

  RVI can be used by application in two different ways:
    - RVI Core - a high-reliability Erlang-based server - project details can be found at https://github.com/GENIVI/rvi_core
    - RVI Lib - a library developed in C that applications can statically (.a) and dynamically (.so) link against - project details can be found at https://github.com/GENIVI/rvi_lib

  The OE layer meta-rvi can be found at https://github.com/GENIVI/meta-rvi. It has recipes to build both RVI Core and RVI Lib.

GDP deliverables

GDP has two deliverables:

1. GDP Master targets contributors and developers familiar with Yocto.
2. GDP releases target users and application developers that want to use GDP as demo platform for their solutions.

Further Information

Tutorials and Howtos

- GENIVI Development platform (GDP) and RVI SOTA client Howto
- GENIVI Development platform (GDP) and OCF
    - Slides about OCF showed at 14th AMM
    - Further information
    - **GDP-160** - Getting issue details... STATUS
- GDP FAQ
- Some initial work has been done on providing acceptance tests for the QEMU build of GDP. The tests are contained in a GITHUB repository, see the python subdirectory for the code and python/Tests for some exemplar tests. These tests can be run using runOneTest.py - the tests assume that you are in the gdp-src-build directory in order to run successfully, alternatively you should set the QEMU_IMAGE_DIR to point to the correct location (or to "." for the currect directory. See the repos README for addtional information.

These links apply to GDP Intrepid version (GDP-ivi7). In any case, there are very interesting:

- SDK preparation, build and setup for GDP-ivi7
- Application development tutorial
- Platform development tutorial
- OCF (meta-genivi-ocf-demo) (page unavailable)

Links

- GENIVI baselines (Yocto and Baserock Baseline)
- GDP roadmap
- GDP FAQ: GDP Frequently Asked Questions
- GDP landing page (home)
- GDP download page.
- GDP feature pages:p
    - GDP 12 version