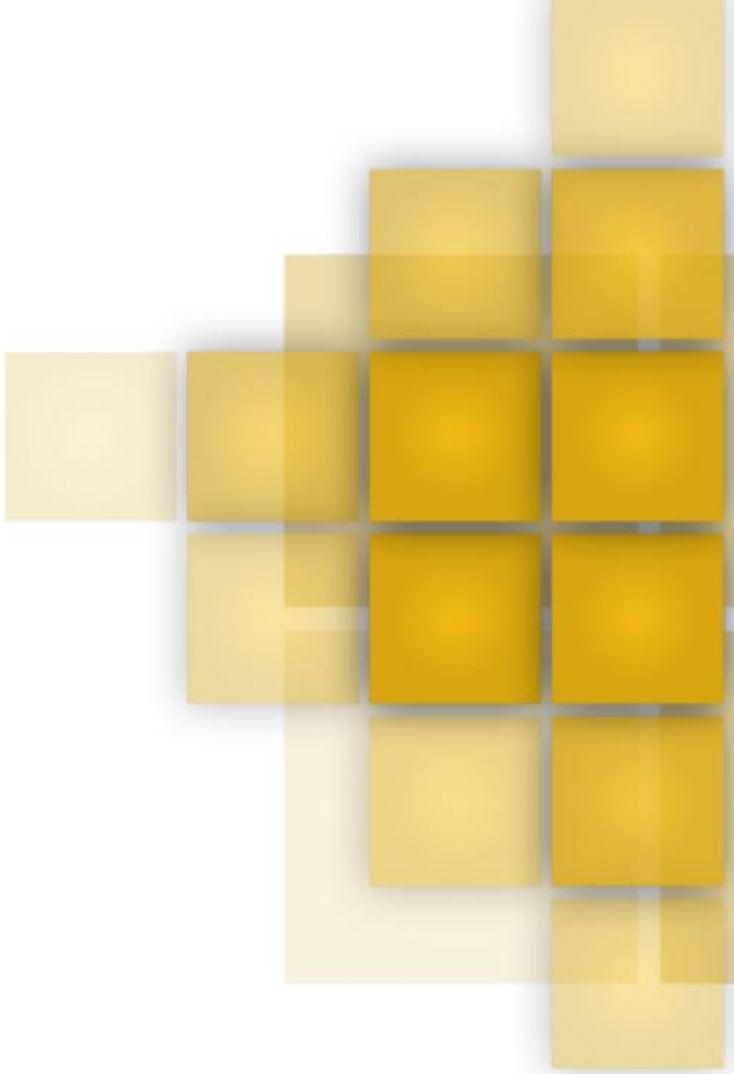


meta-ivi

The Yocto Project layer for In-Vehicle Infotainment

Contents

This presentation introduces **meta-ivi** the Yocto Project‘s layer for **In-Vehicle Infotainment**.

It should give an overview of the **building blocks** used by the GENIVI Alliance to create a software baseline that complies to its latest specification, which role The Yocto Project plays, what that project provides and what it gains.

The targeted audience for this presentations are software developers, with little to no knowledge of The Yocto Project.

Agenda

- Who I am?
- History & Roadmap
- What‘s the problem?
- Examples of non-differentiating?
- Unix like systems* to start from?
- Selection Criteria?
- What does the Yocto Project provide?
- How does the Yocto Project match up?
- The Yocto Project based IVI workflow
- How to use meta-ivi in my build?
- How to contribute? & What to help with?

Who I am?

Holger Behrens
Principal Technologist, Automotive Solutions

Currently active in the GENIVI Alliance
System Infrastructure Expert Group and as the Release Manager for the Yocto based GENIVI baseline within the Baseline Integration Team at the GENIVI Alliance.

I am one of the maintainers of the **meta-ivi** layer.

History of meta-ivi

- **First** „Yocto-IVI“ **prototype** available: February 2012
- **GIT repo**sitory meta-ivi established: 19. April 2012
  - Announced during GENIVI AMM 24-27 April, Paris, France
- **1st release** (denzil) of meta-ivi available: 16. May 2012
  - GENIVI 2.0 (Discovery)
  - based on Yocto 1.2 (denzil), meta-systemd (Yocto Project)
- **3.0.0** release (E-1.0) made available: 15. October 2012
  - GENIVI 3.0 (Excalibur)
  - based on Yocto (master), meta-systemd (Yocto Project)
- **3.0.1** release (E-1.1) available since: 31. October 2012
  - GENIVI 3.0 (Excalibur)
  - based on Yocto 1.3 (danny), meta-systemd (meta-openembedded)
  - GENIVI Open Source Projects (git.projects.genivi.org)

meta-ivi Roadmap will change without notice

- **3.0.2** release (E-1.2) target date: 14. December 2012
  - GENIVI 3.0 (Excalibur)
  - based on Yocto 1.3 (danny), meta-systemd (meta-openembedded)
  - GENIVI Open Source Projects (git.projects.genivi.org)
- **3.0.3** release (E-1.3) target date: 25. January 2013
  - GENIVI 3.0 (Excalibur)
  - based on Yocto 1.3 (danny), meta-systemd (meta-openembedded)
  - GENIVI Open Source Projects (git.projects.genivi.org)
- ...
- **4.0.0** release (F-1.0) target date: 19. April 2012
  - GENIVI next (Foton)
  - based on Yocto master (1.4 ?), meta-systemd (meta-openembedded)
  - GENIVI Open Source Projects (git.projects.genivi.org)

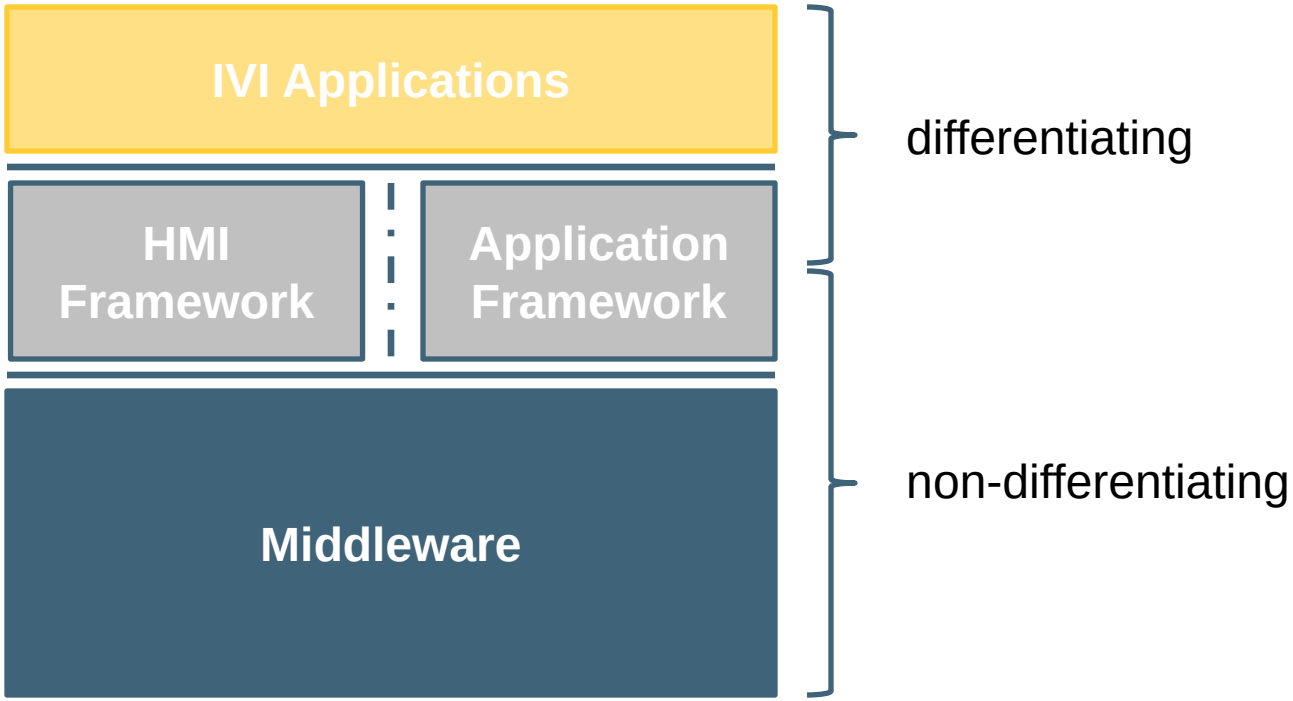
Where to get and how to use it?

- Pre-requisites
  - GENIVI Alliance credentials *(this may go away in the future)*
  - A Yocto Project Compliant Distribution / Build Environment
    - % git clone git://git.yoctoproject.org/poky
  - meta-openembedded - collection of layers for the OE-core universe, including the meta-systemd layer
    - % git clone git://git.openembedded.org/meta-openembedded
- Download layer
  - % git clone git://git.yoctoproject.org/meta-ivi
- Reference build (qemu)
  - follow the meta-ivi README.md

Agenda

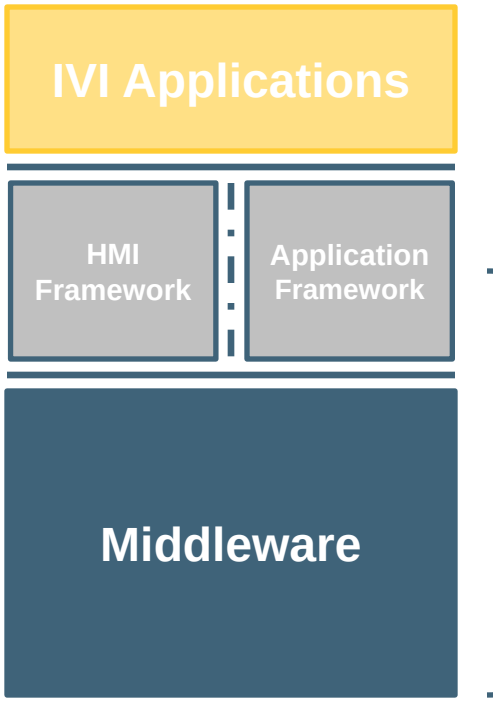
- Who I am?
- History & Roadmap
- What‘s the problem?
- Examples of non-differentiating?
- Unix like systems* to start from?
- Selection Criteria?
- What does the Yocto Project provide?
- How does the Yocto Project match up?
- The Yocto Project based IVI workflow
- How to use meta-ivi in my build?
- How to contribute? & What to help with?

What‘s the problem?



IVI Software Stack (simplified)

Examples for non-differentiating?



non-differentiating

GENIVI

- Linux kernel
- ALSA / WiFi / USB
- eglibc
- systemd & udev
- D-Bus
- FUSE
- ConnMan
- SQLite
- OpenGL-ES 2.0
- ...

IVI Software Stack (simplified)

Requirements / GENIVI® Compliance

- GENIVI: SC, AC, PC – P1 (mandatory), P2 (desirable)
- SC = Specific Component
  - Component that is ultimately defined by its implementation available in form of the source code.
- AC = Abstract Component
  - Component that is defined by its provided and required interfaces as well as its behavior, but does not refer to any specific implementation.
- PC = Placeholder Component
  - Component that is defined only by a set of requirements that its specific implementation has to fulfill.
- Examples:
  - SC – systemd; AC – OpenGL-ES; PC – BT Stack

Operating systems* to start from?*(a subset)*

- Ubuntu
- TIZEN
- Debian
- Fedora
- OpenSUSE
- ChromeOS
- Android
- Ångström
- Poky (The Yocto Project‘s Reference Distro)

***assuming Linux Kernel based software platforms, a subset**

Selection Criteria

- Embedded vs. Desktop vs. Server OS
- Processor Architecture Support
  ARM, Intel, PowerPC, MIPS, 32bit/64bit, ...
- Board Support
- Licensing (i.e. GPLv3 free)
- Commerzial Support
  long-term (LTS), longlong-term („Automotive-grade“ 2..3*LTS)
- Professional Service
- Rich Software Ecosystem
- Sizable and stable developer community
- „Track Records“

Agenda

- Who I am?
- History & Roadmap
- What‘s the problem?
- Examples of non-differentiating?
- Unix like systems* to start from?
- Selection Criteria?
- What does the Yocto Project provide?
- How does the Yocto Project match up?
- The Yocto Project based IVI workflow
- How to use meta-ivi in my build?
- How to contribute? & What to help with?

The Yocto Project

- an open source collaboration project
- hosted by the Linux Foundation
- Wind River is a Yocto Project founding member and Gold member of the advisory board, and actively holds key maintainer and technical lead positions

What does the Yocto Project provide?

- Provides templates, tools and std. methods
- Helps developers create their own custom Linux distributions
- can create customized device specific SDKs
- Supports any hardware architecture
  - validated and tested BSPs in a common format
- Supported by embedded industry leaders
  - Wind River, Mentor Graphics, MontaVista, Enea, ...
  - Intel, Freescale, TI, ...

It's not an embedded Linux distribution – It creates a custom one for you.

How does the Yocto Project match up?

- **Embedded** (vs. Desktop vs. Server OS)
- Processor Architecture Support
  ARM, Intel, PowerPC, MIPS, 32bit/64bit, ...
- Board Support (meta-ti, meta-fsl-arm, meta-intel, meta-fsl-ppc, ...)
- Licensing (supports a GPLv3 free build)
- Commerzial Support
  long-term (LTS), longlong-term („Automotive-grade“ 2..3*LTS)
- Professional Service
- Rich Software Ecosystem
- Sizable and stable developer community
- „Track Records“

Yocto Project Compliance and LTS

- **Reduce fragmentation** in the embedded market by encouraging collaborative development of a **common** set of tools, **standards**, and practices.
- Ensure that these tools, standards, and practices are architecturally **independent** as much as possible.

- Yocto Project and LTSI of Consumer Electronics Workgroup Announce Joint Roadmap (29. Aug. 2012)

Yocto Project Compatible – Criteria

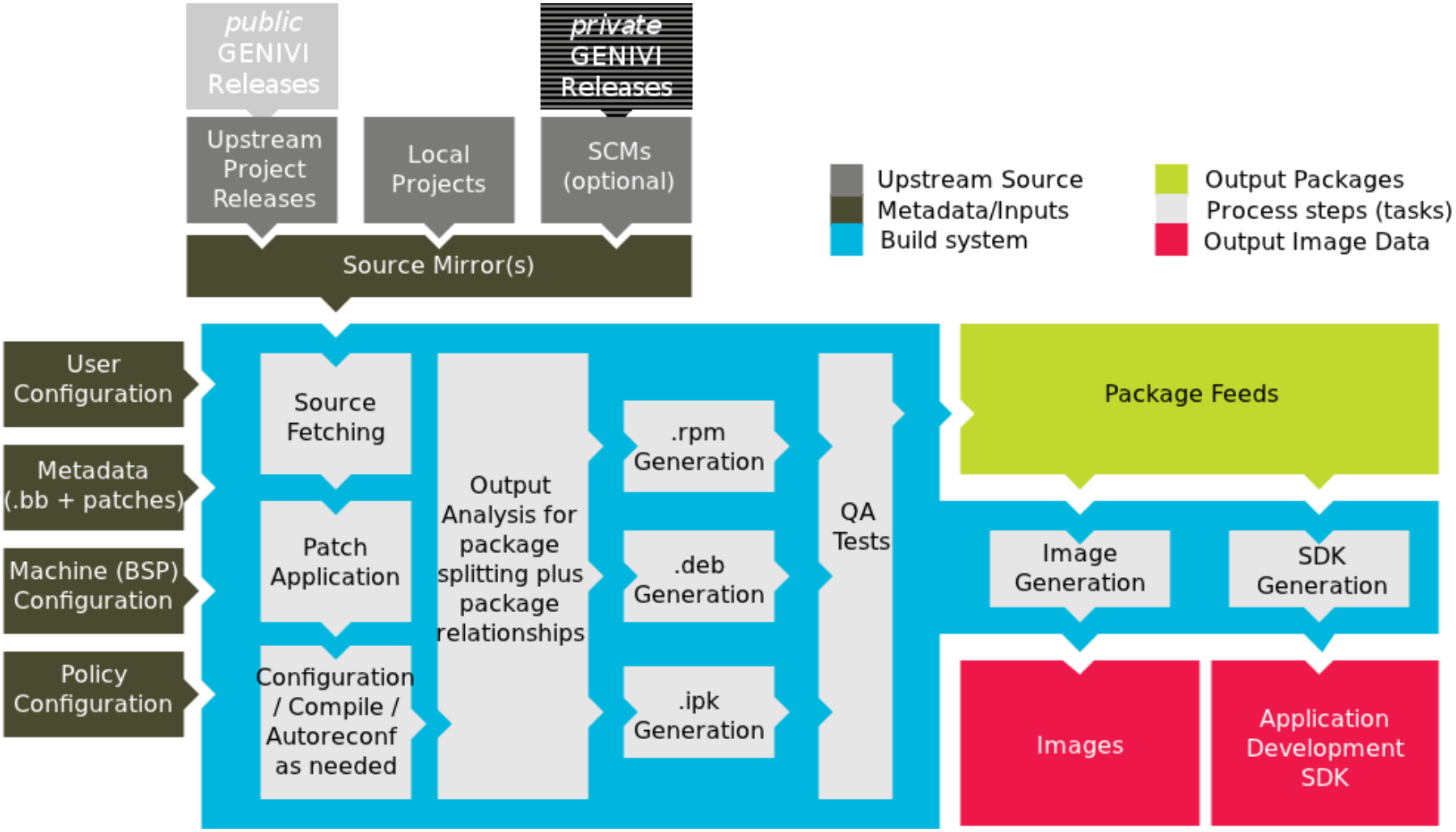
Some of the acceptance criteria include demonstrating the following:

- Hardware support, configuration policy and recipe meta data are separated into different layers that do not depend on each other
- Visible contributions to the OpenEmbedded and components projects of the Yocto Project
- Are an open source project, nonprofit or member of the Yocto Project
- All patches applied to BitBake and OpenEmbedded-Core have been submitted to the open source community
- All layers contain a README file

Agenda

- Who I am?
- History & Roadmap
- What‘s the problem?
- Examples of non-differentiating?
- Unix like systems* to start from?
- Selection Criteria?
- What does the Yocto Project provide?
- How does the Yocto Project match up?
- The Yocto Project based IVI workflow
- How to use meta-ivi in my build?
- How to contribute? & What to help with?

The Yocto Project based IVI workflow



WIND RIVER

meta-ivi

- Provides reference build that is GENIVI compliant (just P1's)
  - QEMU (ARMv7) — emulated machine: vexpressa9
- Should work with Yocto Project compatible board support packages (BSP) and other software or middleware layers
  - known to work on
    - meta-fsl-arm — i.MX53 QSB
    - meta-ti — Pandaboard (OMAP4)
  - known to work with
    - meta-browser - Chromium

How to use meta-ivi in my build?

- Create project build directory
  - source ./poky/oe-init-build
- Tell BitBake to also process meta-ivi layer
  - edit /path/to/build/conf/bblayers.conf
  - add meta-ivi layer to BBLAYERS
- Build image
  - for reference image recipes look into
    - /path/to/meta-ivi/recipes-yocto-ivi/images
  - for example(3.0.1):
    - bitbake excalibur-image

How to contribute?

Adhere to the Yocto Project contribution guide, which describes how to submit patches.

https://wiki.yoctoproject.org/wiki/Contribution_Guidelines

Maintainers:

meta-ivi	holger.behrens@windriver.com
	florin.sarbu@windriver.com

What to help with?

- Board Support
  - meta-ti — Pandaboard (OMAP5)
- Board Support
  - meta-fsl-arm — i.MX6
- Board Support
  - meta-intel, meta-fri2 — Fish River Island - 2
- ...

Questions?

References:

- Embedded Linux Yocto Project™ Announces Compliance Program, Releases Joint Roadmap with Long-Term Support Initiative
  http://www.linuxfoundation.org/news-media/announcements/2012/08/embedded-linux-yocto-project%E2%84%A2-announces-compliance-program-releases
- What is LTSI?
  http://ltsi.linuxfoundation.org/what-is-ltsi
- The Yocto Project Branding Compliance Guidelines
  http://www.yoctoproject.org/branding-compliance-guidelines
- The Yocto Project Contribution Guidelines
  https://wiki.yoctoproject.org/wiki/Contribution_Guidelines
- GENIVI Alliance
  http://www.genivi.org
- GENIVI Open Source Projects
  http://www.genivi.org/projects

