Enabling your i.MX design with the Yocto Project®

Rogerio Nunes

Field Applications Engineer

i.MX – Greater Boston

September 2018 | AMF-ENT-T3322

SECURE CONNECTIONS FOR A SMARTER WORLD

Agenda

- Preface
- Yocto Project Overview
- i.MX Reference Enablement
- Beyond the Reference
- Q&A

Preface

To put things in perspective.

SECURE CONNECTIONS FOR THE SMARTER WORLD

Everything **Smart**

40B+ devices with intelligence shipped in 2020

Processing

Everything **Connected**

1B+ additional consumers online, 30B+ connected devices

Connectivity

Everything **Secure**

Potential economy savings up to half trillion dollars

Security

Automotive | Industrial | Connected Devices | IoT

Embedded Linux for IoT Devices

- Smarter, connected, secure
  - More complexity in the SW stack
  - Potentially larger teams
  - Scalable solutions – multiple applications and multiple devices
  - Security patches
  - Firmware Over The Air updates (OTA)

Easier Is Relative…

- Developing Embedded Linux is an inherently complex undertaking
- We can improve tools and processes to make it more efficient, and somewhat easier
- In the end it is still a very complex problem, that is constantly growing in complexity

Hopefully, awareness will avoid some headaches

Yocto Project Overview

What the Yocto Project (YP) is, what it is not, how it can help you.

Open Source Collaboration Project

IT'S NOT AN EMBEDDED LINUX DISTRIBUTION,
IT CREATES A CUSTOM ONE FOR YOU.

- Provides templates, tools, and methods
- Space for embedded developers to share technologies, software stacks, configurations and best practices
- Helps developers create custom Linux-based systems for embedded products, regardless of the hardware architecture
- Very well documented
- Hierarchical governance structure

Overview and Concepts Manual: https://www.yoctoproject.org/docs/2.5.1/overview-manual/overview-manual.htm
Good introductory article: https://www.embedded.com/electronics-blogs/say-what-/4458600/Why-the-Yocto-Project-for-my-IoT-Project-
Yocto Project Mega-Manual: https://www.yoctoproject.org/docs/2.5.1/mega-manual/mega-manual.html

Members (Aug 2018)

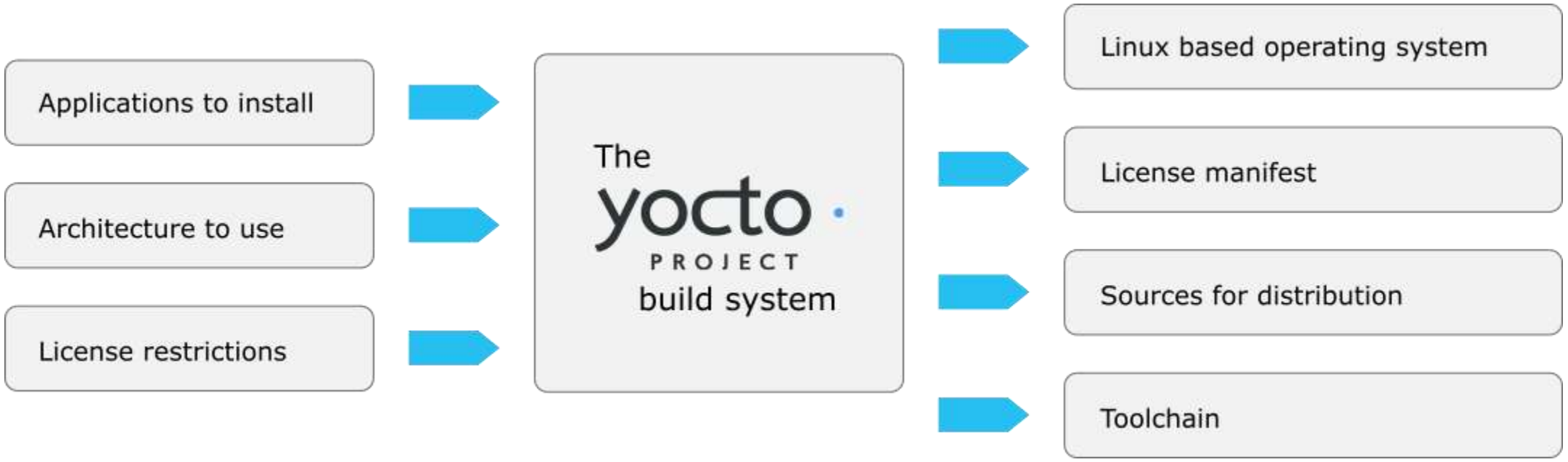
Platinum members

Gold members

Silver members

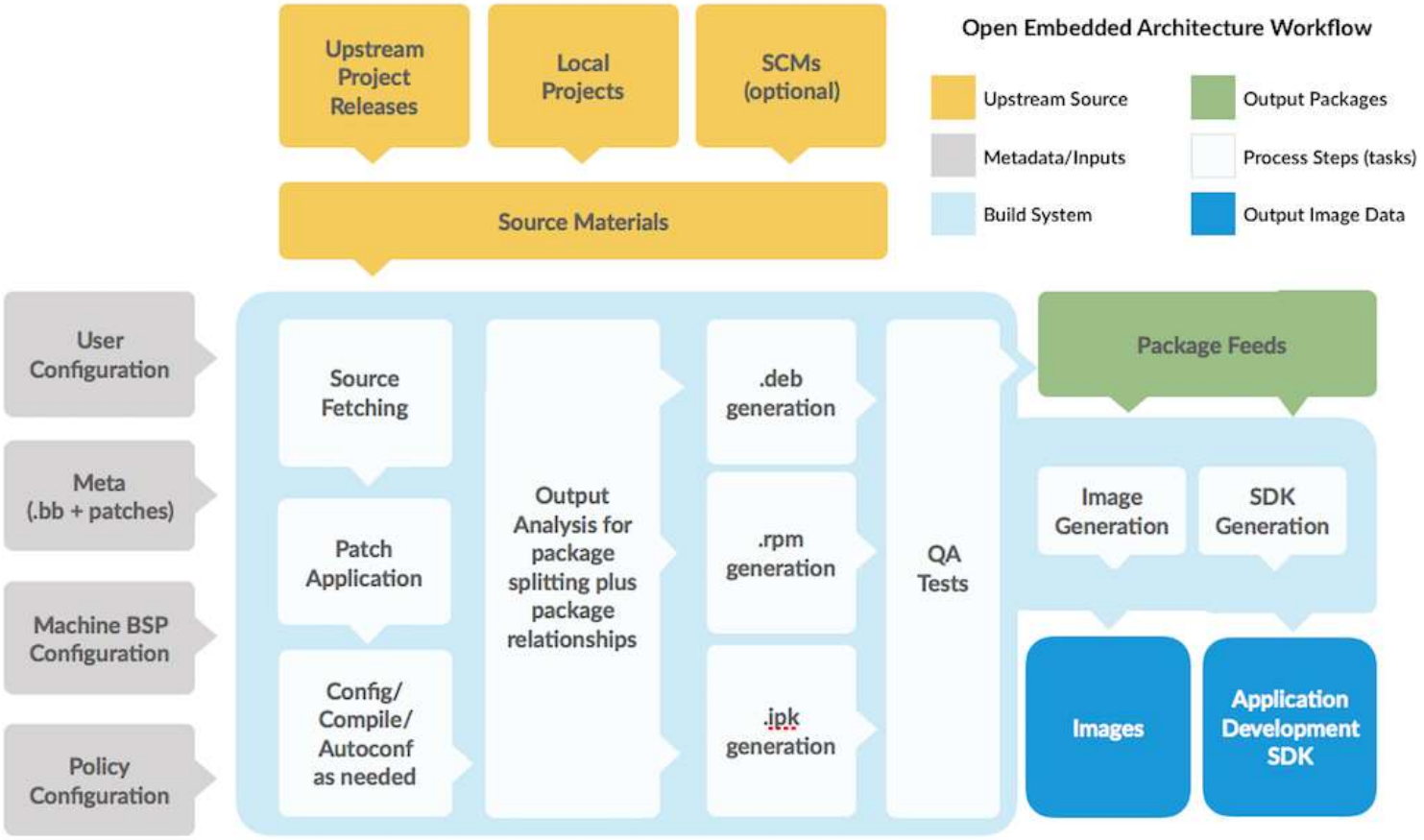
Source: https://www.yoctoproject.org/ecosystem/members/

High Level View of the Yocto Project Core System



Source: https://github.com/CollaborativeWritersHub/heading-for-the-yocto-project

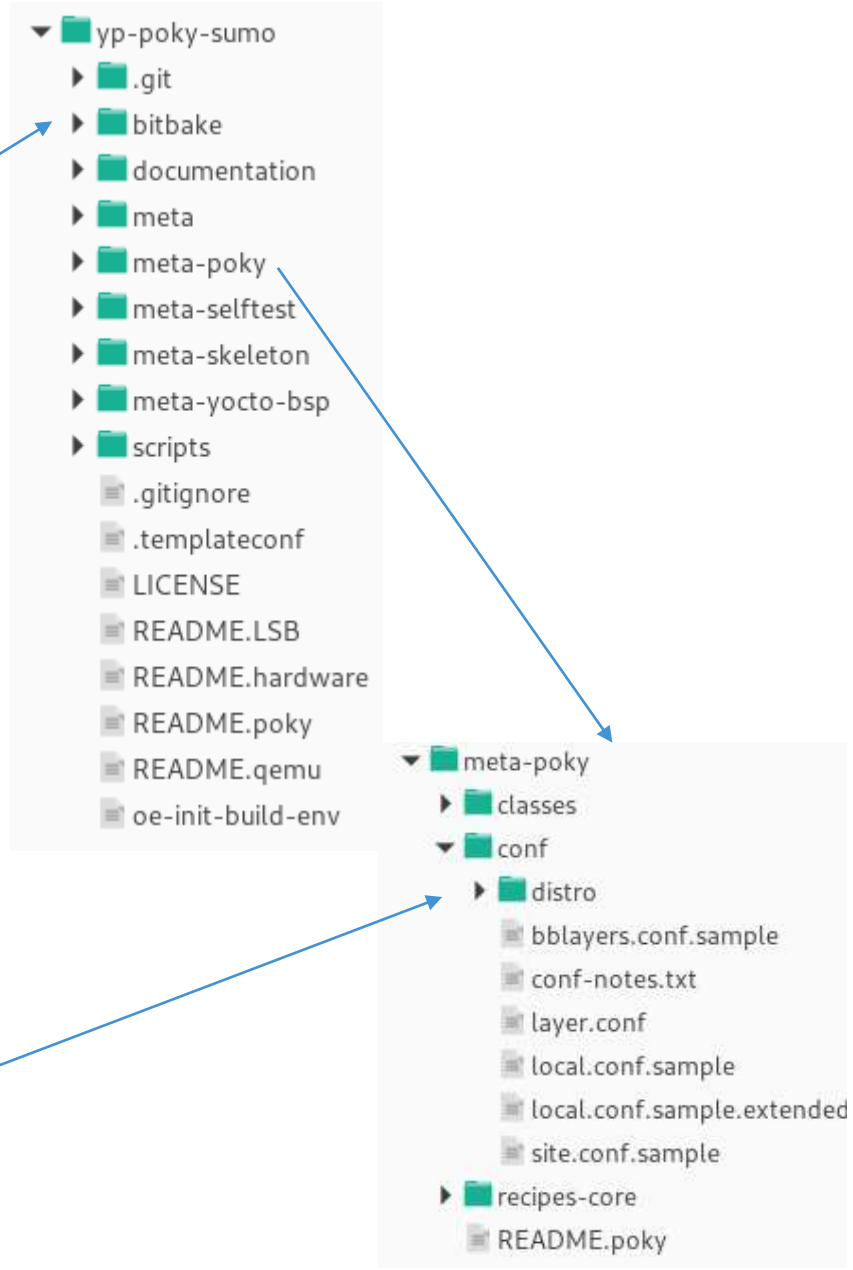
General Workflow



Source: https://www.yoctoproject.org/wp-content/uploads/2017/07/yp-how-it-works-new-diagram.png

Building Blocks

- Metadata
  - Recipes
    - .bb, .bbappend
  - Configuration
    - .conf
  - Classes
    - .bbclass
- Bitbake
  - Build tool used to bake recipes
- Poky
  - Contains the build system (bitbake)
  - Reference metadata and distro configuration



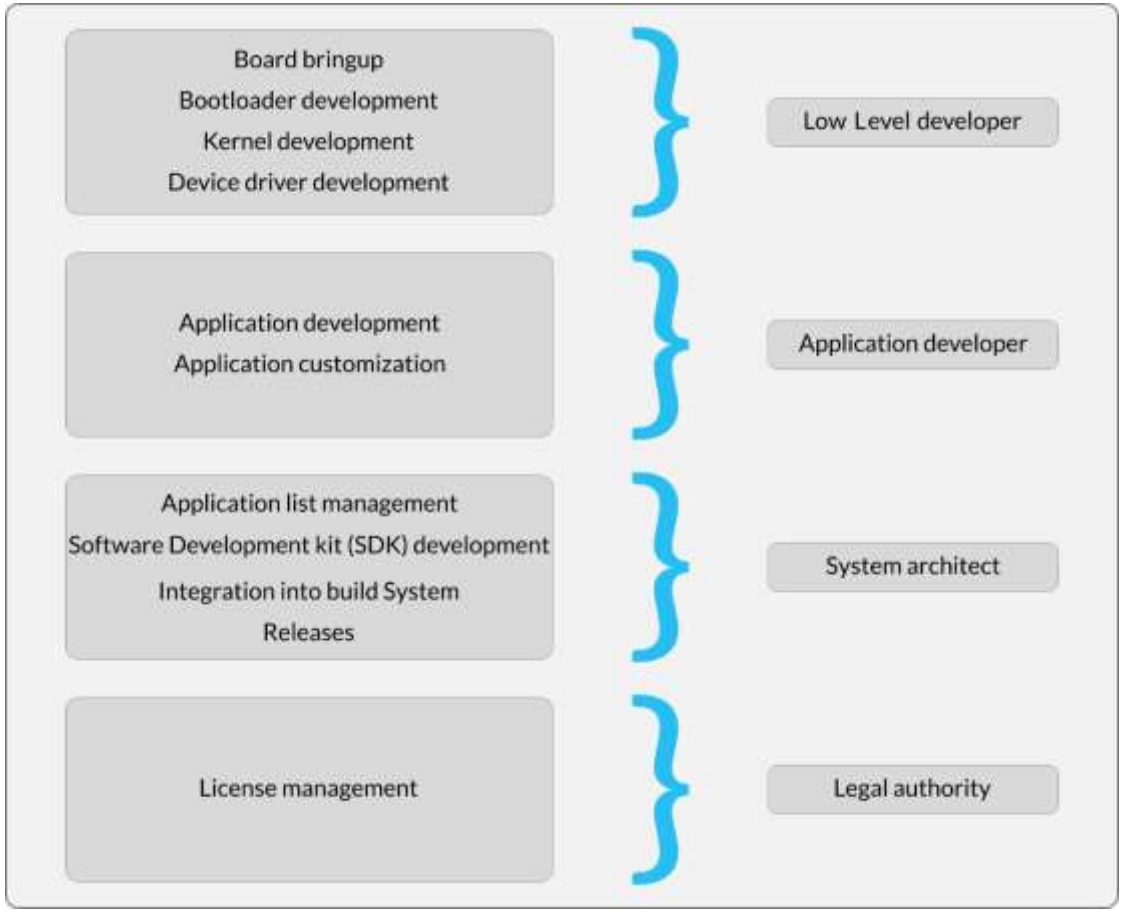
Terms for reference: https://www.yoctoproject.org/software-overview/
More details in the backup section at the end.

Layers

- Different types of metadata and customizations can be isolated into layers
- Re-usable across projects
- Individually maintained
  - Layers are prioritized
  - Not all layers are necessarily compatible
- Example of layer types: Base, BSP, Distro, Middleware, GUI…
- Content from one layer can override content from another layer
  - bbapend, and configuration files
- Layer names start with meta- by convention

More on layers: https://www.yoctoproject.org/docs/2.5.1/dev-manual/dev-manual.html#understanding-and-creating-layers
More details in the backup section at the end.

Example of Project Activities & Roles Supported by the YP



Source: https://github.com/CollaborativeWritersHub/heading-for-the-yocto-project

SDK Generation

- meta-toolchain recipe generates a basic toolchain tuned for the specific target HW
  - # bitbake meta-toolchain
- Recipes inherit the populate_sdk task, which creates a BSP with a sysroot compatible with the recipe passed as argument
  - # bitbake -c populate_sdk <image_recipe>
- SDKs are deployed under <build>/tmp/deploy/sdks
- SDKs are executable scripts that install toolchain and sysroot in any host machine for software development

i.MX Reference Enablement

What YP enablement there is for the i.MX product line, and how to get started.

Why a Yocto Project Based BSP*

- Broad and active community
- Diversity of layers and recipes
- Support for multiple architectures
- Tools for cross-development
- Usually more flexible and scalable than traditional Linux distributions
- NXP focus on the BSP layer to enable and support HW specifics

*We often refer to the i.MX BSP as a different entity than the i.MX BSP layer in the Yocto Project. The first contains the second, along with other SW pieces…

NXP YP Linux BSP Overview

- Composed by multiple layers ==>
  - Only the first released/maintained by NXP
- Uses Repo tool and manifest files to tie the layers together
- Each release is a snapshot
  - Points to specific commits for each layer
- Based on the community BSP
  - meta-freescale layer introduced in 2015
    - Based on meta-fsl-am with all its history

- **meta-fsl-bsp-release (NXP)**
  - **https://source.codeaurora.org/external/imx/meta-fsl-bsp-release**
- meta-browse
  - git://github.com/OSSystems/meta-browser
- meta-freescale
  - git://git.yoctoproject.org/meta-freescale
- meta-freescale-3rdparty
  - git://github.com/Freescale/meta-freescale-3rdparty
- meta-freescale-distro
  - git://github.com/Freescale/meta-freescale-distro
- meta-openembedded
  - git://github.com/openembedded/meta-openembedded
- meta-qt5
  - git://github.com/meta-qt5/meta-qt5
- poky
  - git://git.yoctoproject.org/poky

i.MX Yocto Project User's Guide: https://www.nxp.com/docs/en/user-guide/i.MX_Yocto_Project_User's_Guide_Linux.pdf

Repo

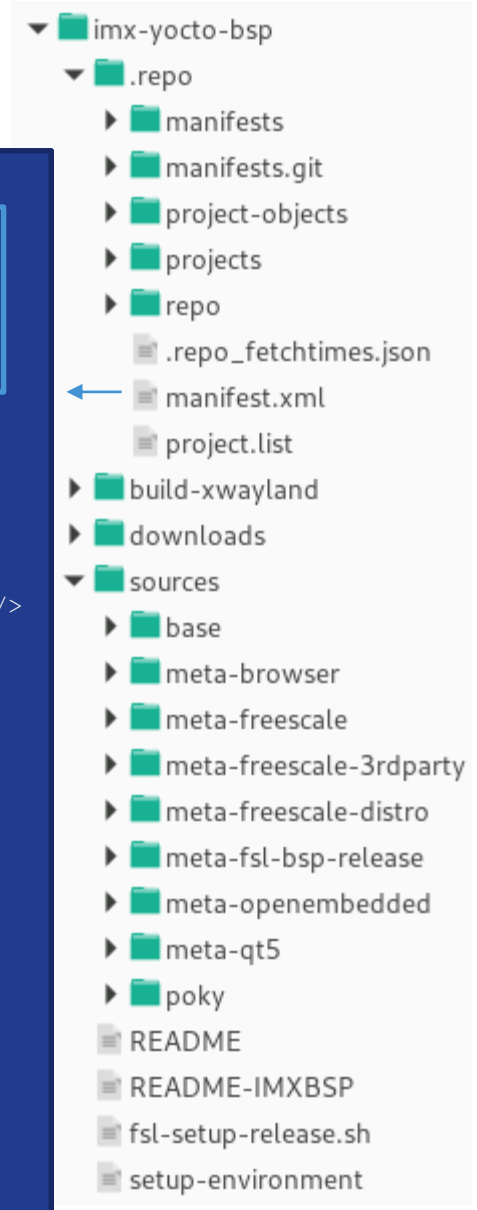
- Tool built on top of Git by Google
- Developed initially to manage multiple Android git repositories
- Not meant to replace git
- The repo command is an executable Python script that you can put anywhere in your path
- Manifest files are used to describe projects
- The i.MX BSPs have always used repo since migration to YP

Source: https://gerrit.googlesource.com/git-repo/
Repo with Android as reference: https://source.android.com/setup/develop/repo

BSP Organization

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

  <default sync-j="2"/>

  <remote fetch="git://git.yoctoproject.org" name="yocto"/>
  <remote fetch="git://github.com/Freescale" name="freescale"/>
  <remote fetch="git://github.com/openembedded" name="oe"/>
  <remote fetch="git://github.com/OSSystems" name="OSSystems"/>
  <remote fetch="git://github.com/meta-qt5"  name="QT5"/>
  <remote fetch="https://source.codeaurora.org/external/imx" name="CAF"/>

  <project remote="yocto" revision="0ec241873367e18f5371a3ad9aca1e2801dcd4ee" name="poky" path="sources/poky"/>
  <project remote="yocto" revision="49ac225a38f6d84519798e3264f2e4d19b84f70a" name="meta-freescale" path="sources/meta-freescale"/>

  <project remote="oe" revision="dacfa2b1920e285531bec55cd2f08743390aaf57" name="meta-openembedded" path="sources/meta-openembedded"/>

  <project remote="freescale" revision="70535e13dd2aabbad53243518f4cc5064d284592" name="fsl-community-bsp-base" path="sources/base">
            <linkfile dest="README" src="README"/>
            <linkfile dest="setup-environment" src="setup-environment"/>
  </project>

  <project remote="freescale" revision="1d6d5961dbf82624b28bb318b4950a64abc31d12" name="meta-freescale-3rdparty" path="sources/meta-
freescale-3rdparty"/>
  <project remote="freescale" revision="0ec6d7e206705702b5b534611754de0787f92b72" name="meta-freescale-distro" path="sources/meta-
freescale-distro"/>

  <!-- Use a newer version of chromium than is currently available on rocko -->
  <project remote="OSSystems" revision="d6f9aed41c73b75a97d71bff060b03a66ee087b1" name="meta-browser" path="sources/meta-browser" />
  <project remote="QT5" revision="32bb7d18a08d1c48873d7ab6332d4cc3815a4dff" name="meta-qt5" path="sources/meta-qt5" />

  <project remote="CAF" name="meta-fsl-bsp-release" path="sources/meta-fsl-bsp-release" revision="rocko-4.9.88-2.0.0_ga" >
     <linkfile src="imx/tools/fsl-setup-release.sh" dest="fsl-setup-release.sh"/>
     <linkfile src="imx/README" dest="README-IMXBSP"/>
  </project>

</manifest>
```

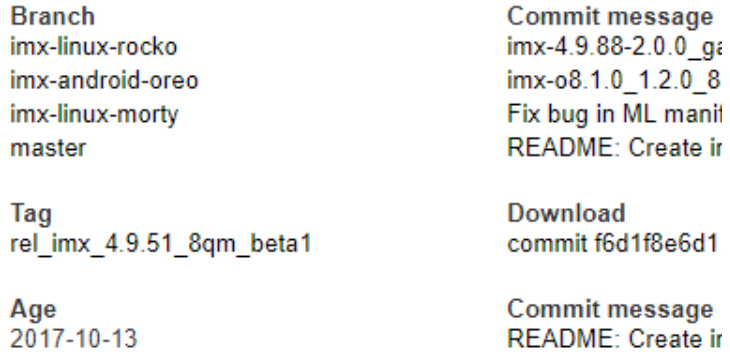
Example of how each layer is frozen at a specific commit hash.



manifest.xml -> manifests/imx-4.9.88-2.0.0_ga.xml

BSP Releases

- Repo manifest files
  - Git repository: https://source.codeaurora.org/external/imx/imx-manifest ==>
  - Same git repository for Android BSPs
  - One branch per YP and Android release
    - Multiple Alpha, Beta, GA, and demo manifest files under each branch
- Always check the Release Notes pdf for boards supported, U-Boot, Kernel, and Graphics library versions, features, known issues, etc.
- GA releases might be specific to one i.MX device family
  - Explicit in the release name/manifest file
  - Usually expected to work with other device families
  - Might be a better starting point for new products, as it may bring along additional patches, and newer Graphics lib, for instance
  - But… be aware, not fully tested on other the i.MX devices/families
  - There will likely be a consolidated GA eventually, tested on all currently supported devices
- Tested images are internal only (GPU vs non-GPU parts)



Yocto Project Releases: https://wiki.yoctoproject.org/wiki/Releases

Current BSP and Schedule

- Most recent YP BSP
  - Branch: imx-linux-rocko
    - ChangeLog
    - GraphicsChange
    - README
    - imx-4.9.88-2.0.0-edgescale-demo.xml
    - imx-4.9.88-2.0.0-iotg-greengrass-demo.xml
    - imx-4.9.88-2.0.0-kodi-demo.xml
    - imx-4.9.88-2.0.0_agl-demo.xml
    - **imx-4.9.88-2.0.0_ga.xml**
    - imx-4.9.88-2.0.0_genivi.xml
    - imx-4.9.88-2.1.0-8mm_alpha.xml
    - imx-4.9.88-2.2.0-8qxp_beta2.xml

- Lots of NPIs, schedule in flux
- Last 4.9.123 GA for i.MX8MM coming soon
- Following release will be 4.14-beta in Q4
  - 4.9 stops LTS in 2019
  - Consolidated release
  - First based on Sumo
  - OPTEE will be the default for i.MX 6/7/8

i.MX YP BSP manifest repository: https://source.codeaurora.org/external/imx/imx-manifest

New in Recent Releases

- Class added to enable i.MX-base machine overrides
  - Supports enabling common features on different machines more easily
- ARM v8 based products added
  - Additional complexity in the boot flow, and consequently in the image generation
- Multilib support added for i.MX 8 devices
  - The i.MX Yocto Project User's Guide explains how to enable
- Devices with DRM support are compatible with Liri
  - Currently only i.MX 8s

Documentation

- Available at nxp.com/imxtools
  - Overview tab
    - Documentation area (not the Documentation tab, which is not up to date – site has been under review)
- Documents available
  - **i.MX Linux Release Notes**
  - **i.MX Yocto Project User's Guide** – host setup, initialization, image generation… (how to start and customize)
  - i.MX Linux User's Guide – starting with pre-built binaries
  - i.MX Linux Reference Manual – driver level
  - i.MX Graphics User's Guide
  - i.MX VPU API
  - i.MX Porting Guide
- Check also the README file in the manifest repository (how to start info is available here as well)
  - https://source.codeaurora.org/external/imx/imx-manifest
- Pointers to latest documents also available at
  - https://imxdev.gitlab.io/imxdoc/software-links/

Wi-Fi/BT Murata Modules

- We don’t use the Murata Yocto Layer
- Recipes are included as part of the NXP BSP
- Cypress support will be added in coming 4.14 BSP release

NXP Distros and Images

- NXP distros target different graphics backends
  - fsl-imx-x11
  - fsl-imx-wayland
  - fsl-imx-xwayland
  - fsl-imx-fb

- Images
  - meta-freescale
    - fsl-image-mfgtool-initramfs
      - Same U-Boot and Kernel from other regular images
      - Moving to UUU
  - meta-freescale-distro
    - fsl-image-machine-test
    - fsl-image-multimedia
    - fsl-image-multimedia-full
  - meta-fsl-bsp-release
    - fsl-image-gui
    - fsl-image-qt5
    - fsl-image-qt5-validation-imx
    - fsl-image-validation-imx

Simple Tweaks Often Used

- Lots of good tips can be found in the following documents
  - Yocto Project Development Tasks Manual
    - https://www.yoctoproject.org/docs/2.5.1/dev-manual/dev-manual.html
  - i.MX Yocto Project User's Guide
    - https://www.nxp.com/docs/en/user-guide/i.MX_Yocto_Project_User's_Guide_Linux.pdf
- Bellow tweaks are used very often
  - How to add packages into an image for a quick test
    - Add package name to IMAGE_INSTALL or CORE_IMAGE_EXTRA_INSTALL in local.conf
  - How to conserve disk space during builds
    - Add a global inheritance from class rm_work in local.conf (INHERIT += "rm_work“)
  - How to share downloads and sstate folders
    - In local.conf, point DL_DIR to a common directory to share downloads between builds/hosts, and SSTATE_DIR to a common directory to share state files

Beyond the Reference

How to customize and further leverage the potential of i.MX/YP solutions.

FSL Community

- The main site is hosted at https://freescale.github.io
- Very well documented
- Pre built images available for NXP and 3rd party boards
- Development is centralized in the meta-freescale mailing list
  - https://lists.yoctoproject.org/listinfo/meta-freescale
- Community has their own forks of GSTreamer, U-Boot, Kernel
  - Hosted at Github
  - Not supported by NXP
- Community has a regular release schedule
  - Follows the main Yocto Project schedule - every 6 months (April and October)
- Releases use Repo and manifest files as well
  - https://github.com/Freescale/fsl-community-bsp-platform
  - Moving target – repo manifest points to branch heads, and not commit hashes

- *The meta-fsl-bsp-release layer from NXP aims to release the updated and new Yocto Project recipes and machine configurations for new releases that are not yet available on the existing meta-freescale and meta-freescale-distro layers in the Yocto Project*

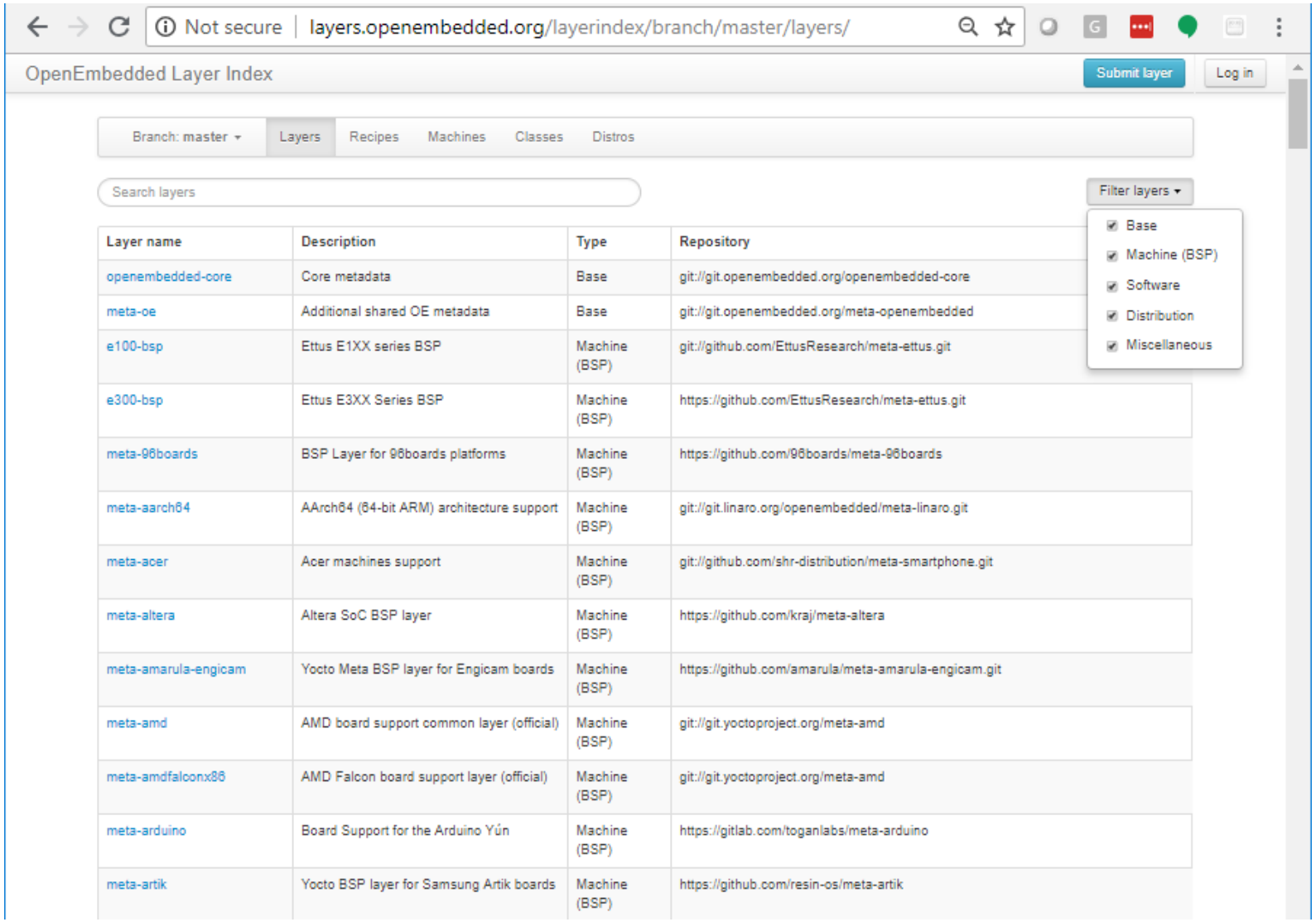
Customizing the BSP

- Repo use, and some steps below are optional
- Avoid reinventing the wheel, and add other external layers that help you solve your problem when recipes/functionalities you need are already available somewhere else
- Create an application layer
  - Customize busybox
  - Add GUI recipes
  - Add product images
  - Add custom distro – recommended
- Fork U-Boot, Kernel, and any other project necessary
- Create a custom BSP layer
  - Re-use the layers from the NXP release as baseline
  - Add recipes for the new U-Boot and Kernel forks
  - Add a new Machine configuration
- Create additional layer(s) as needed, depending on complexity

i.MX Yocto Project User's Guide: https://www.nxp.com/docs/en/user-guide/i.MX_Yocto_Project_User's_Guide_Linux.pdf (Chapter 7 and FAQ)
i.MX BSP Porting Guide: https://www.nxp.com/docs/en/user-guide/i.MX_BSP_Porting_Guide_Linux.pdf

Layer Index

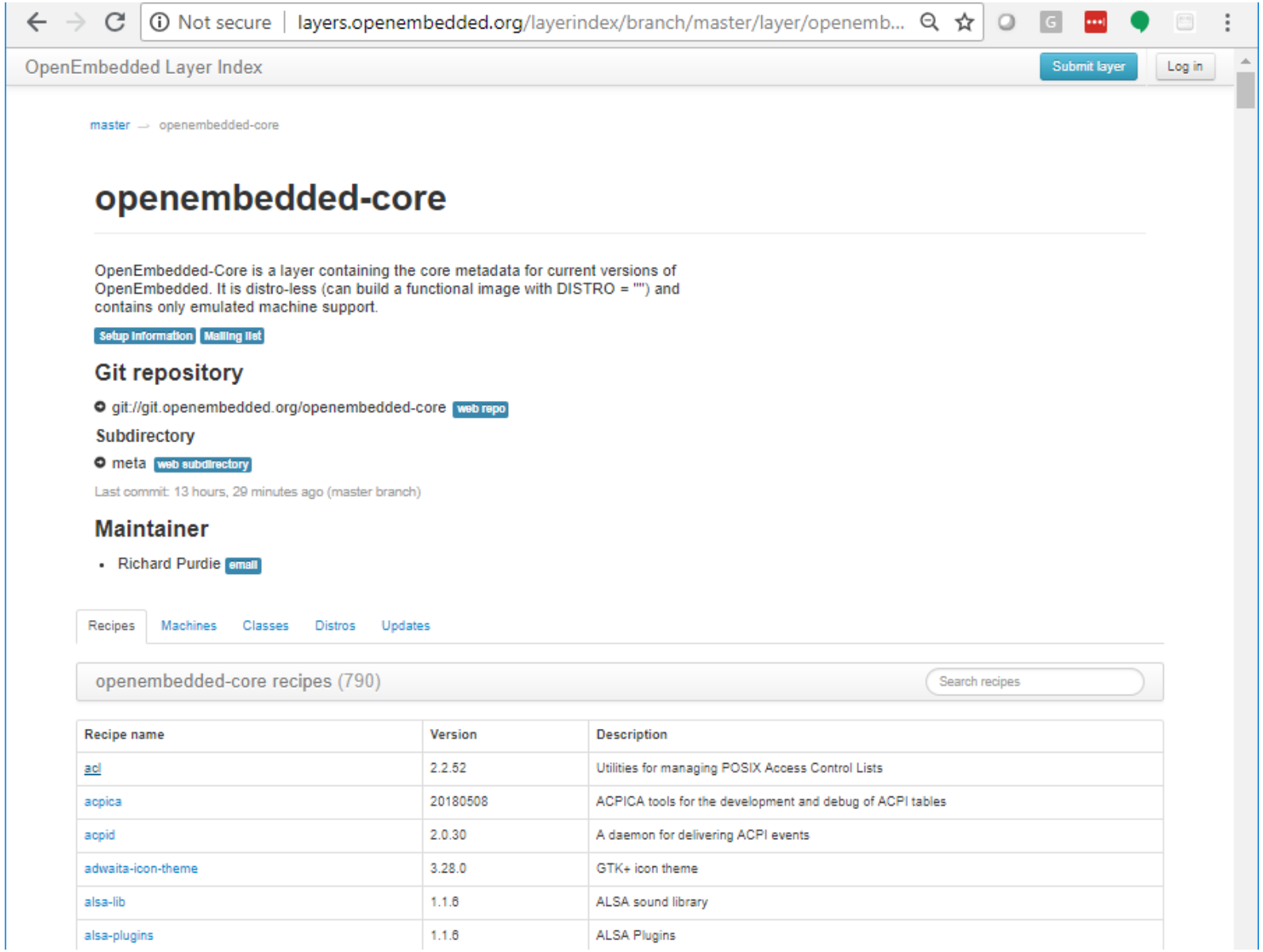
- The index project itself is available as a reference for custom index implementation: http://git.yoctoproject.org/cgit/cgit.cgi/layerindex-web/



Layer name	Description	Type	Repository
openembedded-core	Core metadata	Base	git://git.openembedded.org/openembedded-core
meta-oe	Additional shared OE metadata	Base	git://git.openembedded.org/meta-openembedded
e100-bsp	Ettus E1XX series BSP	Machine (BSP)	git://github.com/EttusResearch/meta-ettus.git
e300-bsp	Ettus E3XX Series BSP	Machine (BSP)	https://github.com/EttusResearch/meta-ettus.git
meta-96boards	BSP Layer for 96boards platforms	Machine (BSP)	https://github.com/96boards/meta-96boards
meta-aarch64	AArch64 (64-bit ARM) architecture support	Machine (BSP)	git://git.linaro.org/openembedded/meta-linaro.git
meta-acer	Acer machines support	Machine (BSP)	git://github.com/shr-distribution/meta-smartphone.git
meta-altera	Altera SoC BSP layer	Machine (BSP)	https://github.com/kraj/meta-altera
meta-amarula-engicam	Yocto Meta BSP layer for Engicam boards	Machine (BSP)	https://github.com/amarula/meta-amarula-engicam.git
meta-amd	AMD board support common layer (official)	Machine (BSP)	git://git.yoctoproject.org/meta-amd
meta-amdfalconx86	AMD Falcon board support layer (official)	Machine (BSP)	git://git.yoctoproject.org/meta-amd
meta-arduino	Board Support for the Arduino Yún	Machine (BSP)	https://gitlab.com/toganlabs/meta-arduino
meta-artik	Yocto BSP layer for Samsung Artik boards	Machine (BSP)	https://github.com/resin-os/meta-artik

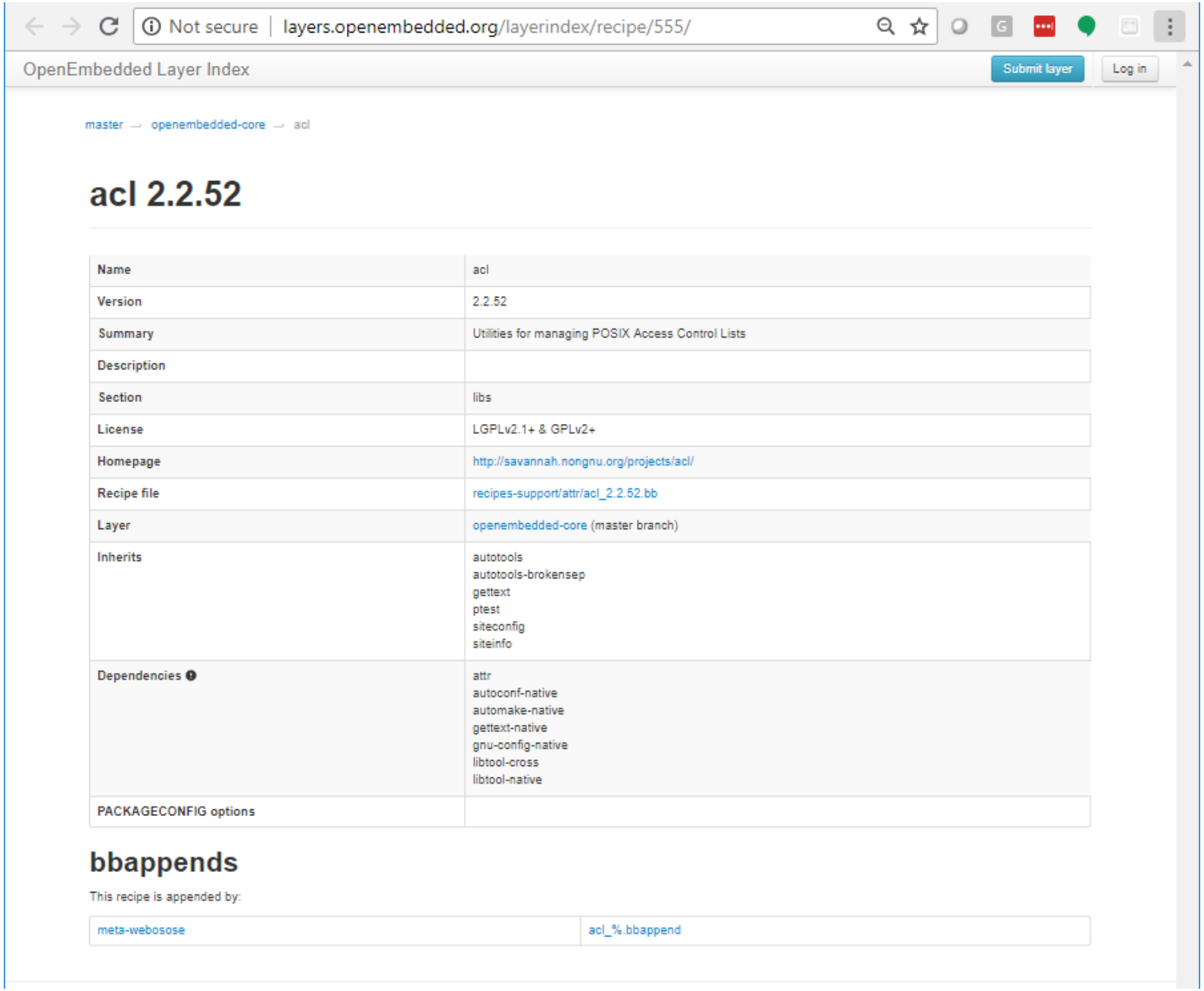
Layer Index

- Example of layer details



Layer Index

- Example of recipe details



Other Tools, Services, References for Development*

- Toaster
  - Enables configuration and builds via web interface
  - https://www.yoctoproject.org/docs/2.5.1/toaster-manual/toaster-manual.html
- Eclipse IDE, eSDK, devtool
  - These tools improve the development workflow
  - https://www.yoctoproject.org/docs/2.5.1/sdk-manual/sdk-manual.html
- CROPS
  - Cross platform enablement with containers
  - https://git.yoctoproject.org/cgit/cgit.cgi/crops/about/
- YP Upgrades by Dell EMC
  - Slides: https://events.static.linuxfound.org/sites/events/files/slides/Yocto-upgrades-ELC-2017.pdf
  - Video: https://www.youtube.com/watch?v=F5R8VXnfwYw
- O.S. Systems
  - BSP customization, build automation, OTA
  - https://www.ossystems.com.br
- Mender
  - Over-the-air software updates for connected Linux devices
  - https://mender.io

* Neither tested, nor supported by NXP

Some Takeaways

- Do no lose sight of the big picture of what the YP is and is not
- There's a steep learning curve to learn how to properly use the YP, but not everybody in the organization needs to learn, and you soon start benefiting from all it can offer
- There's a lot of good documentation in the YP site covering all sorts of topics
  - There are good videos online, and books as well

- NXP provides BSP layers to enable NXP HW
  - BSP packages as reference
  - Not all YP tools are enabled/tested
  - Support comes from NXP for BSP release layers, and basic components (U-Boot, Kernel, Graphics…), while support for YP in general comes from the community
- NXP's BSP release layer and documentation have evolved
  - It is easier to get started, and do more off the gate
  - More flexibility to support actual products

- If you are serious about using YP in production, learn how to go beyond where a BSP will get you, and how to implement tools and processes to keep SW for your products current

Q&A

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com

Backup

Terms For Reference (1)

- Configuration Files: Files which hold global definitions of variables, user defined variables and hardware configuration information. They tell the build system what to build and put into the image to support a particular platform.

- Recipe: The most common form of metadata. A recipe will contain a list of settings and tasks (instructions) for building packages which are then used to build the binary image. A recipe describes where you get source code and which patches to apply. Recipes describe dependencies for libraries or for other recipes, as well as configuration and compilation options. They are stored in layers.

- Layer: A collection of related recipes. Layers allow you to consolidate related metadata to customize your build, and isolate information for multiple architecture builds. Layers are hierarchical in their ability to override previous specifications. You can include any number of available layers from the Yocto Project and customize the build by adding your layers after them. The Layer Index is searchable for layers within Yocto Project.

- Metadata: A key element of the Yocto Project is the meta-data which is used to construct a Linux distribution, contained in the files that the build system parses when building an image. In general, Metadata includes recipes, configuration files and other information refering to the build instructions themselves, as well as the data used to control what things get built and to affect how they are built. The meta-data also includes commands and data used to indicate what versions of software are used, and where they are obtained from, as well as changes or additions to the software itself (patches or auxiliary files) which are used to fix bugs or customize the software for use in a particular situation. OpenEmbedded Core is an important set of validated metadata.

- OpenEmbedded-Core: oe-core is meta-data comprised of foundation recipes, classes and associated files that are meant to be common among many different OpenEmbedded-derived systems, including the Yocto Project. It is a curated subset of an original repository developed by the OpenEmbedded community which has been pared down into a smaller, core set of continuously validated recipes resulting in a tightly controlled and an quality-assured core set of recipes.

Terms For Reference (2)

- Poky: A reference embedded distribution and a reference test configuration created to 1) provide a base level functional distro which can be used to illustrate how to customize a distribution, 2) to test the Yocto Project components, Poky is used to validate Yocto Project, and 3) as a vehicle for users to download Yocto Project. Poky is not a product level distro, but a good starting point for customization. Poky is an integration layer on top of oe-core.

- Build System - "Bitbake": a scheduler and execution engine which parses instructions (recipes) and configuration data. It then creates a dependency tree to order the compilation, schedules the compilation of the included code, and finally, executes the building of the specified, custom Linux image (distribution). BitBake is a make-like build tool. BitBake recipes specify how a particular package is built. They include all the package dependencies, source code locations, configuration, compilation, build, install and remove instructions. Recipes also store the metadata for the package in standard variables. Related recipes are consolidated into a layer. During the build process dependencies are tracked and native or cross-compilation of the package is performed. As a first step in a cross-build setup, the framework will attempt to create a cross-compiler toolchain (Extensible SDK) suited for the target platform.

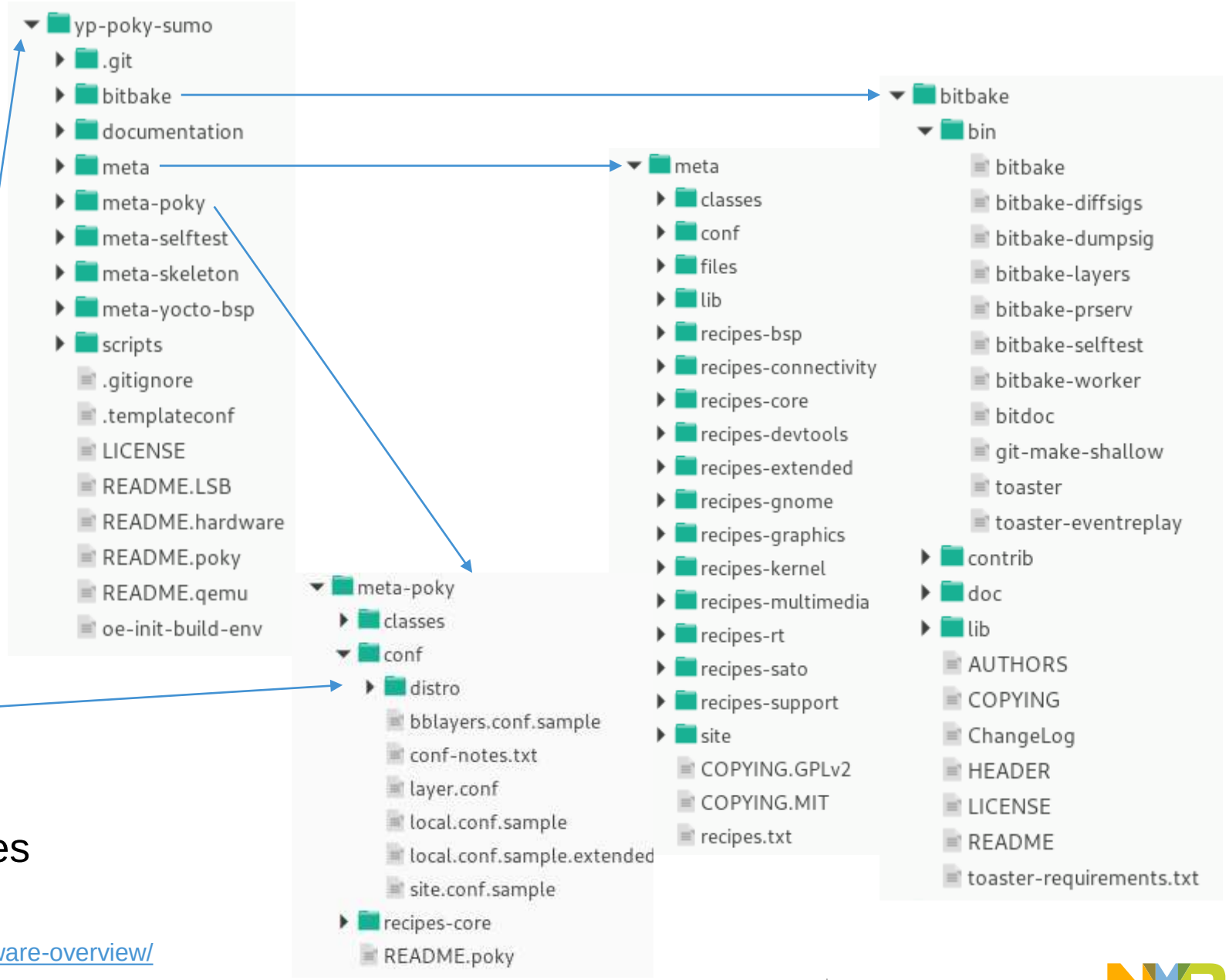
- Packages: The output of the build system used to create your final image.

- Extensible Software Development Kit (ESDK): A custom SDK for application developers that allows them to incorporate their library and programming changes back into the image to make their code available to other apps developers.

- Image: A binary form of a Linux distribution (operating system) intended to be loaded onto a device.

Building Blocks

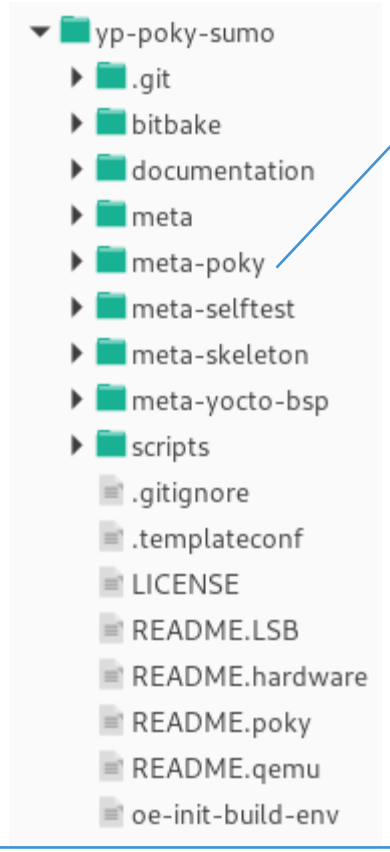
- Metadata
  - Recipes
    - .bb, .bbappend
  - Configuration
    - .conf
  - Classes
    - .bbclass
- Poky
  - Contains the build system
  - Reference distro
- Bitbake
  - Build tool used to bake recipes

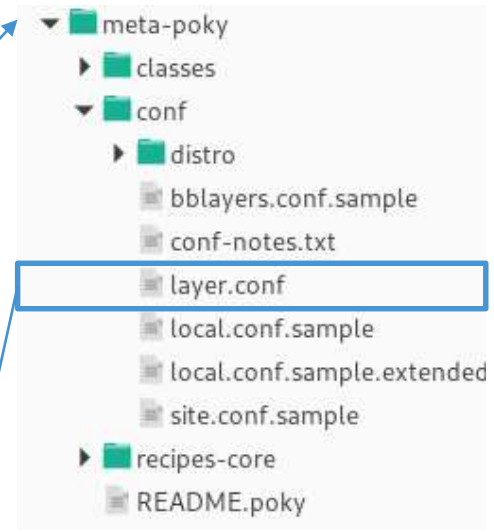


Terms for reference: https://www.yoctoproject.org/software-overview/

Layers

- Different types of metadata and customizations can be isolated into layers, re-usable across projects
- Example of layer types: Base, BSP, Distro, Middleware, GUI…
- Content from one layer can override content from another layer (.bbapend, configuration)
- Layers are prioritized
- Some layers may be incompatible
- Layer names start with meta- by convention






```
# We have a conf and classes directory, add to BBPATH
BBPATH =. "${LAYERDIR}:"

# We have recipes-* directories, add to BBFILES
BBFILES += "${LAYERDIR}/recipes-*/*/*.bb \
        ${LAYERDIR}/recipes-*/*/*.bbappend"

BBFILE_COLLECTIONS += "yocto"
BBFILE_PATTERN_yocto = "^${LAYERDIR}/"
BBFILE_PRIORITY_yocto = "5"

LAYERSERIES_COMPAT_yocto = "sumo"

# This should only be incremented on significant changes
that will
# cause compatibility issues with other layers
LAYERVERSION_yocto = "3"

LAYERDEPENDS_yocto = "core"

REQUIRED_POKY_BBLAYERS_CONF_VERSION = "2"
```

More on layers: http://www.yoctoproject.org/docs/1.6/dev-manual/dev-manual.html#understanding-and-creating-layers

Getting Started with the BSP

```
# Install the repo tool
$ mkdir ~/bin
$ curl https://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=${PATH}:~/bin

# Setup the BSP (most recent branch: imx-linux-rocko; default manifest file: imx-4.9.88-2.0.0_ga.xml)
$ mkdir imx-yocto-bsp
$ cd imx-yocto-bsp
$ repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-rocko -m imx-4.9.88-2.0.0_ga.xml
$ repo sync

# Initialize the build environment (using machine imx8mqevk and distro fsl-imx-xwayland as example)
# NOTE: first time initializing a build directory, it's necessary to use fsl-setup-release.sh with all arguments
# from the second time on, source setup-environment <build_dir> should be used instead
$ DISTRO=fsl-imx-xwayland MACHINE=imx8mqevk source fsl-setup-release.sh -b build-xwayland

# Agree to license terms to continue

# Bake an image (fsl-image-validation-imx, for example)
$ bitbake fsl-image-validation-imx
```

i.MX Linux Release Notes: https://www.nxp.com/docs/en/release-note/i.MX_Linux_Release_Notes.pdf (important info about supported features, machines, etc.)
i.MX Yocto Project User's Guide: https://www.nxp.com/docs/en/user-guide/i.MX_Yocto_Project_User's_Guide_Linux.pdf

Build Configuration

```
MACHINE ??= 'imx8mqevk'
DISTRO ?= 'fsl-imx-xwayland'
PACKAGE_CLASSES ?= "package_rpm"
EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
USER_CLASSES ?= "buildstats image-mklibs image-prelink"
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS ??= "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    STOPTASKS,/tmp,100M,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
    ABORT,${SSTATE_DIR},100M,1K \
    ABORT,/tmp,10M,1K"
PACKAGECONFIG_append_pn-qemu-native = " sdl"
PACKAGECONFIG_append_pn-nativesdk-qemu = " sdl"
CONF_VERSION = "1"

DL_DIR ?= "${BSPDIR}/downloads/"
ACCEPT_FSL_EULA = "1"
```

build-xwayland/conf/local.conf

```
LCONF_VERSION = "6"

BBPATH = "${TOPDIR}"
BSPDIR := "${@os.path.abspath(os.path.dirname(d.getVar('FILE', True))
+ '/../..')}"

BBFILES ?= ""
BBLAYERS = " \
  ${BSPDIR}/sources/poky/meta \
  ${BSPDIR}/sources/poky/meta-poky \
  \
  ${BSPDIR}/sources/meta-openembedded/meta-oe \
  ${BSPDIR}/sources/meta-openembedded/meta-multimedia \
  \
  ${BSPDIR}/sources/meta-freescale \
  ${BSPDIR}/sources/meta-freescale-3rdparty \
  ${BSPDIR}/sources/meta-freescale-distro \
"

# i.MX Yocto Project Release layers
BBLAYERS += " ${BSPDIR}/sources/meta-fsl-bsp-release/imx/meta-bsp "
BBLAYERS += " ${BSPDIR}/sources/meta-fsl-bsp-release/imx/meta-sdk "

BBLAYERS += " ${BSPDIR}/sources/meta-browser "
BBLAYERS += " ${BSPDIR}/sources/meta-openembedded/meta-gnome "
BBLAYERS += " ${BSPDIR}/sources/meta-openembedded/meta-networking "
BBLAYERS += " ${BSPDIR}/sources/meta-openembedded/meta-python "
BBLAYERS += " ${BSPDIR}/sources/meta-openembedded/meta-filesystems "
BBLAYERS += " ${BSPDIR}/sources/meta-qt5 "
```

build-xwayland/conf/bblayers.conf

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com