FTF 2016
TECHNOLOGY FORUM

LAYERS OF VARIOUS ARCHITECTURE SUPPORT AND COMMERCIAL SOFTWARE DELIVERY

FTF-DES-N1843

ZHENHUA LUO
LETITIA DUMITRESCU

LAUREN POST
FTF-DES-N1843
MAY 19, 2016

AGENDA

- Yocto Project Introduction
- NXP Unified Yocto Project Layer
- i.MX BSP Delivery Mechanism
- QorIQ SDK Delivery Mechanism
- QorIQ Yocto Project SDK Usage
- QorIQ SDK Installer

YOCTO PROJECT INTRODUCTION

Yocto Project Introduction: What is the Yocto Project?

- An open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture.
- Collaboration among many hardware manufacturers, open-source operating systems vendors, and electronics companies to bring some order to the chaos of embedded Linux development. Including NXP, Intel, TI, Wind River, Mentor Graphics are contributing to the Yocto Project
- NXP is an active part of the upstream community and is a full Yocto Project Member and a member of the Advisory Board

Official site: www.yoctoproject.org

Yocto Project Introduction: Building Blocks

- **OpenEmbedded Core**: core metadata and build information to build baseline embedded systems
- **Poky**: Yocto Project example distribution which integrates all the required pieces and makes an official release. Poky contains common components, toolchain and bitbake engine
- **Bitbake**: parses metadata and runs tasks. At the core of Poky is the bitbake task executor together with various types of configuration files
- **Metadata**: .bb files are usually referred to as 'recipes‘
- **Class**: (.bbclass) Contain information which is useful to share between metadata files
- **Configuration**: (.conf) files define various configuration variables which govern what Poky does
- **Layers**: Sets of common recipes such as poky, meta-nxp

Yocto Project Introduction: Yocto Project Releases

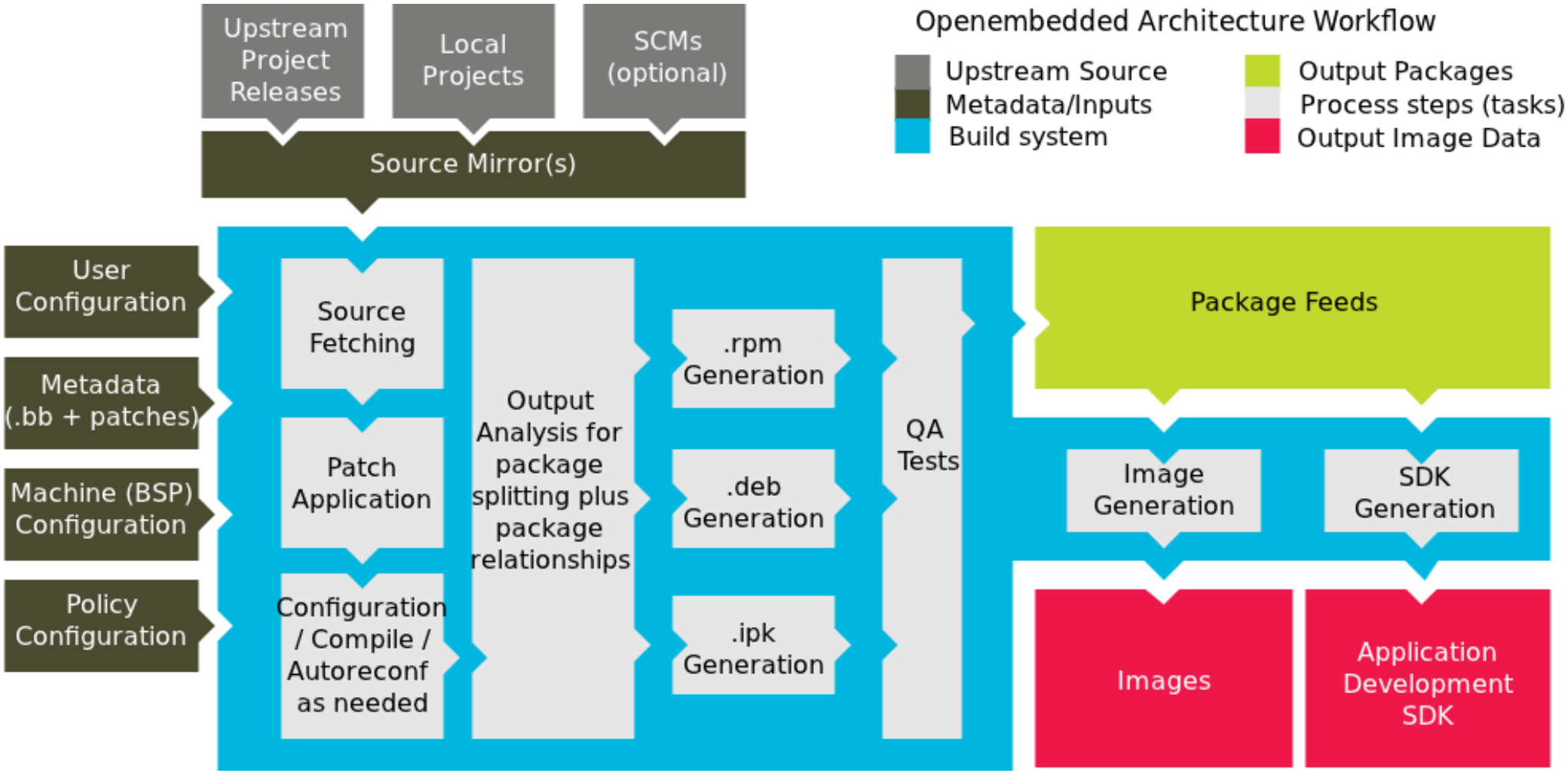
- Yocto Project has releases on 6 month cycles approximately every April and October. Releases are named in month of release.

- **Next Release**: Yocto Project 2.2 (Master) in October 2016

- **Latest Release**: Yocto Project 2.1 (Krogoth) in April 2016

- **Past releases**:
  - Yocto Project 2.0 (Jethro) in October 2015
  - Yocto Project 1.8 (Fido) in April 2015
  - Yocto Project 1.7 (Dizzy) in October 2014
  - Yocto Project 1.6 (Daisy) in April 2014
  - Yocto Project 1.5 (Dora) in October 2013
  - Yocto Project 1.4 (Dylan) in April 2013
  - Yocto Project 1.3 (Danny) in October 2012
  - Yocto Project 1.2 (Denzil) in April 2012

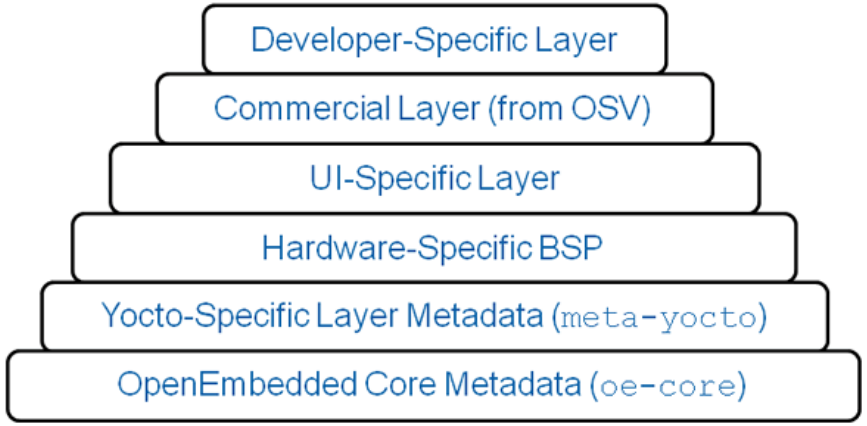
Yocto Project Introduction: Architecture Workflow



Yocto Project Introduction: Layers

The OpenEmbedded build system supports organizing Metadata into multiple layers. Layers allow you to isolate different types of customizations from each other.

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



Source: https://www.yoctoproject.org/tools-resources/projects/openembedded-core

Let's check out: http://layers.openembedded.org/layerindex/

Yocto Project Introduction: Documentation

The Yocto Project documentation is available on the Yocto Project website:

- https://www.yoctoproject.org/documentation

Recommended Reading:

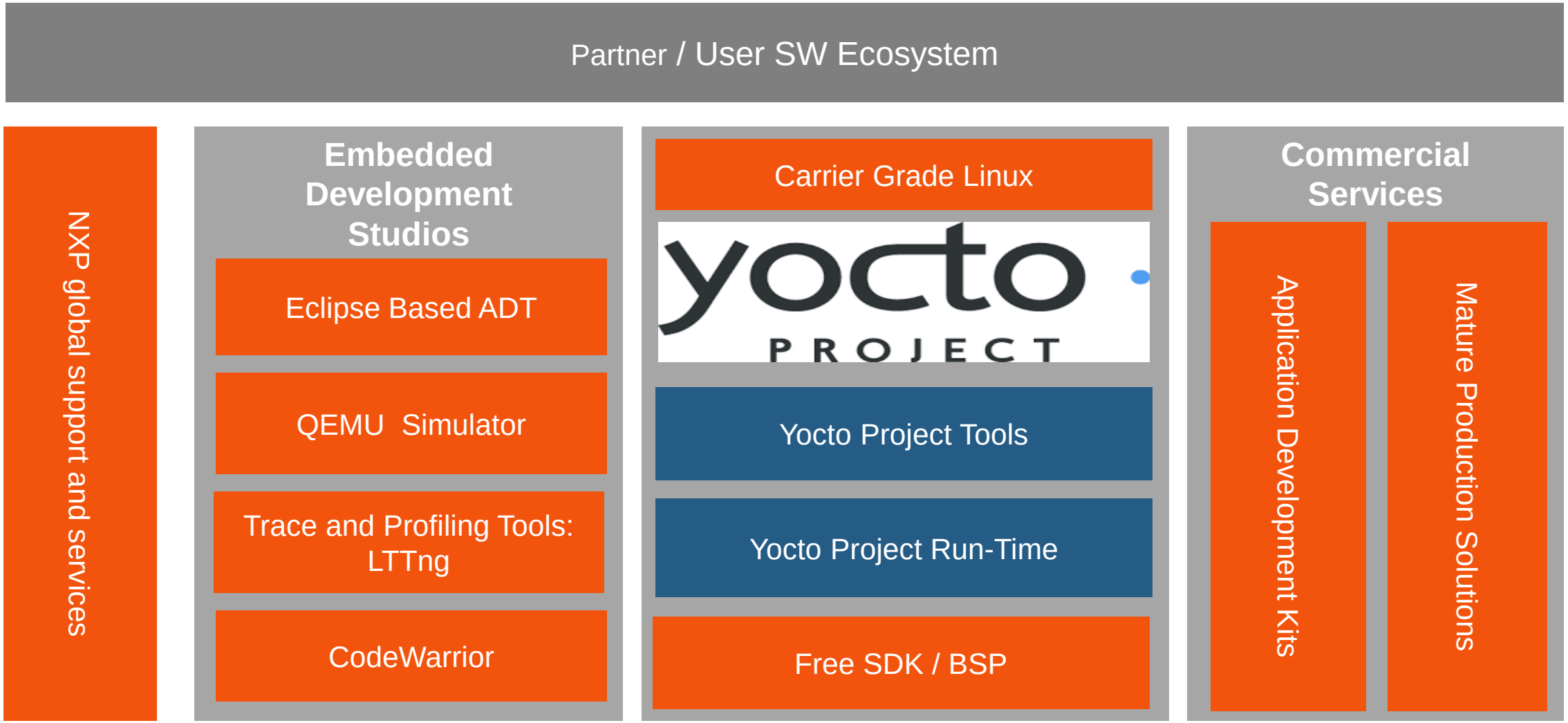
- BitBake User Manual Link
- Yocto Project Reference Manual Link
- Yocto Project Complete Manual Link

NXP UNIFIED YOCTO PROJECT LAYER

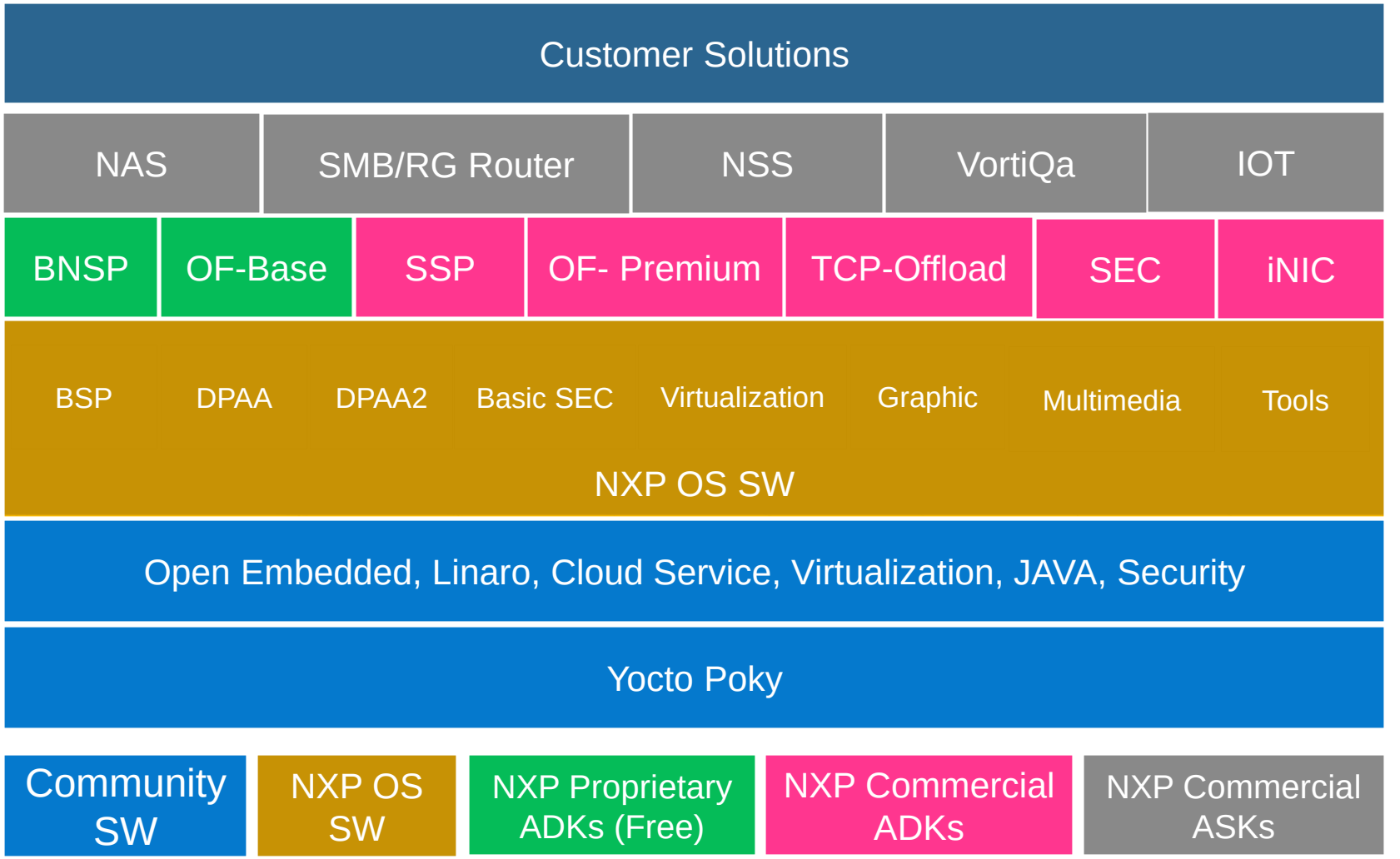
NXP Unified Yocto Project Layer: Benefits

- **B** Provide unified build environment for NXP products
- **E** Share the efforts of NXP products
- **N** Work with Yocto Ecosystem more closely
- **E** Collaborate with community for development and testing
- **F** Simplify the usage with single layer
- **I** Reduce the maintain effort with less layer combinations
- **T** Align with NXP upstream strategy
- **S** Collaborate with Linaro to use LAVA for regression testing

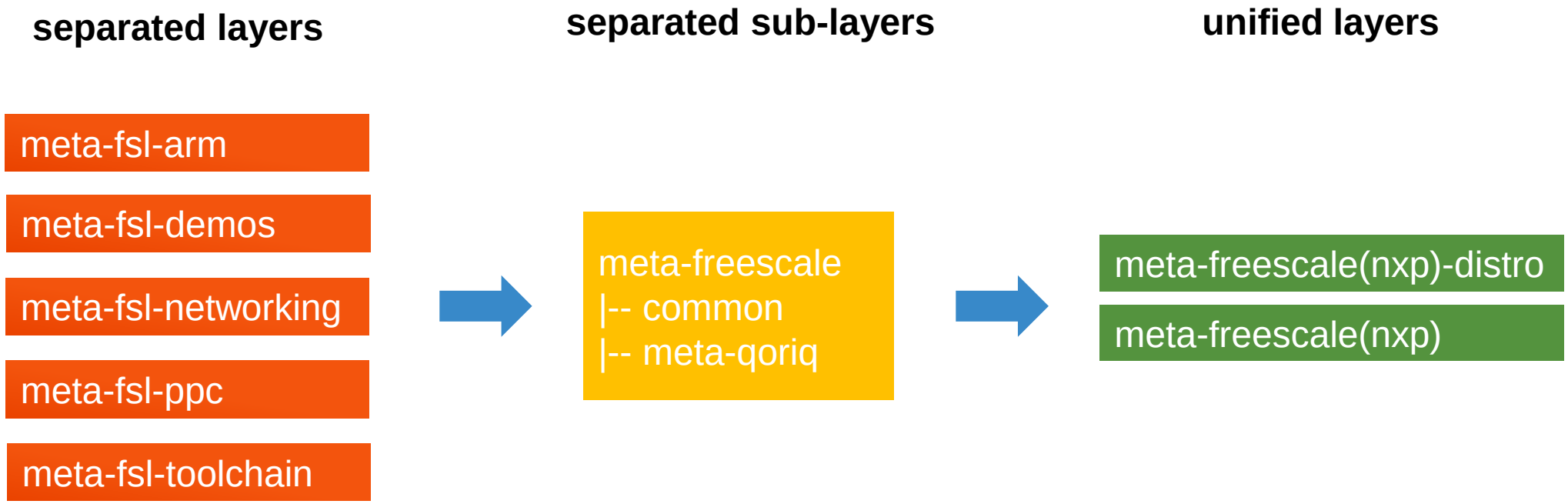
NXP Unified Yocto Project Layer: Ecosystem SW Architecture



NXP Unified Yocto Project Layer: Layers Architecture



NXP Unified Yocto Project Layer: Structure Evolvement



NXP Unified Yocto Project Layer: Transition

Supported Architectures

- ARMv7, ARMv8, PowerPC, PowerPC64

QorIQ Processors

- ARM® and Power Architecture® cores consolidated with common kernel and U-boot
- SDK 1.9 is the first SDK of unified layer
- SDK 2.0 is on-going
- The meta-fsl-arm and meta-fsl-ppc are planned to be deprecated in H2-2016

NXP Unified Yocto Project Layer: Transition

i.MX Processors

- i.MX has more community engagement than QorIQ and requires community signoff of meta-freescale before it can replace meta-fsl-arm.
- Waiting on QorIQ changes in meta-freescale to be signed off by community.
- Until then using existing release process as described in i.MX Yocto Project User's Guide.

I.MX BSP MECHANISM

i.MX Release Model

i.MX continues to release with the same release model as documented in our release documents using the repo manifest model.

- Each release is based on an official Yocto Project base which is updated each April and October.
- Each year we upgrade our kernel after LTS version is announced by kernel.org
- Current kernel is 4.1
- Each GA release is supported for 1 year.
- Patches are released in the same GA release so refreshing a GA release using repo sync will pull in the latest patches for that GA release.

i.MX Release

Release	3.14.52	4.1.15	Future
Kernel	3.14.52	4.1.15	4.1.X
Patch Updates	3.14.52-1.1.1 Graphics p8.4	4.1.15-1.1.0 i.MX 6UL 4.1.15-1.1.1 i.MX 6UL 4.1.15-1.2.0 i.MX 7D (soon)	
Graphics	3.14.52-1.1.0 – v5p7.4 3.14.52-1.1.1 – v5p8.4	4.1.15-1.1.0 – v5p8.3 4.1.15-1.2.0 – v5p8.4	V6 graphics with both 32 and 64bit support
Multimedia	4.0.8 Gstreamer 1.4	4.0.9 Gstreamer 1.6	4.1.0 with 64bit support Gstreamer 1.8
Supported Boards	i.MX 6Q/QP/D/DL/SL/SX i.MX 6UL	i.MX 6Q/QP/D/DL/SL/SX i.MX 6UL i.MX 7Dual with 4.1.15-1.2.0	New silicon
U-Boot	2015.04	2015.04	2016.03

Release Instructions

Each release manifest has a README with instructions to download

http://git.freescale.com/git/cgit.cgi/imx/fsl-arm-yocto-bsp.git/tree/README?h=imx-4.1.15-1.0.0_ga

Instructions

repo init –u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git –b <manifest branch>

repo sync

MACHINE=<imx machine> DISTRO=<distro> source ./fsl-setup-release.sh –b <build dir>

Example

repo init –u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git –b imx-4.1.15-1.0.0_ga

repo sync

MACHINE=imx6qsabresd DISTRO=fsl-imx-x11 source ./fsl-setup-release.sh –b bld-x11

i.MX Graphics

Current graphics supports the 5.0.11 and Wayland 1.9

Graphical backends are supported using DISTROs provided in the BSP for

- X11, Xwayland and Framebuffer
- Specify the distro on setup or change in conf/local.conf

Versions

- 3.14.52 and 4.1.15 both support same graphics 5.0.11
- 3.14.52 is **last release** to support software floating point
- Next GA release for new GPU silicon will be v6 and support 64bit and 32 bit.

i.MX Additional Layers

Some i.MX content is released via additional layers not part of the default build

Layer	Description	Download
meta-fsl-bcmdhd	Bluetooth WiFI	nxp.com download
meta-fsl-aacp	AAC Plus audio decoder	nxp.com download
meta-fsl-microsoft	Microsoft audio decoders and parsers	Restricted access Requires Microsoft License
,eta-fsl-ac3	AC3 audio decoder	Restricted access Requires Dolby License
meta-fsl-ddp	DD-Plus audio decoder	Restricted access Requires Dolby License
meta-fsl-real	Real Networks audio decoder and parser	Restricted access Requires Real Networks License
meta-fsl-eink	E Ink display for 7Dual	Requires E Ink license

i.MX Demos

Demos are not supported but show how to integrate some new features.

Each demo is based on 4.1.15-1.0.0 GA release on git.freescale.com in imx section.

Look at README for each for information on how to setup.

Demo	Description	External
AGL	Autograde Linux on i.MX 6QP SABRE SD/AUTO using Wayland Weston backend	http://git.freescale.com/git/cgit.cgi/imx/meta-nxp-agl.git/
Genivi	Genivi on on i.MX 6Q/QP SABRE SD/AUTO using Wayland backend	http://git.freescale.com/git/cgit.cgi/imx/meta-nxp-genivi.git/
XBMC	XBMC using on i.MX 6QP SABRESD using Framebuffer backend	http://git.freescale.com/git/cgit.cgi/imx/meta-nxp-xbmc.git/
IOTG	Internet of Things Gateway using i.MX 6UltraLite EVK – router and access point capabilities and supporting applications	http://git.freescale.com/git/cgit.cgi/imx/meta-nxp-iotg.git/

QORIQ SDK DELIVERY MECHANISM

QorIQ SDK Delivery Mechanism: Release Format

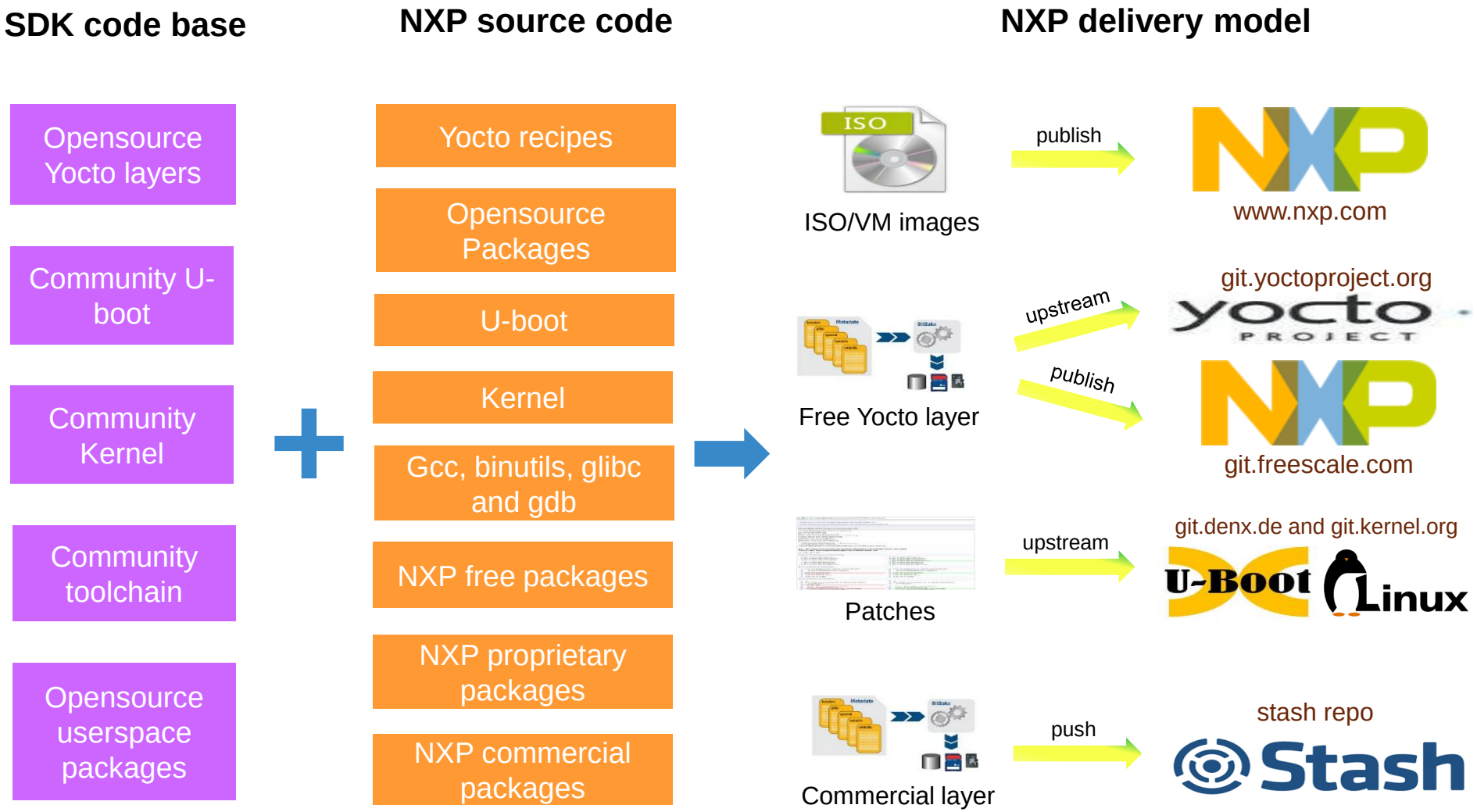
Currently there are several release formats for Yocto SDK.

- SDK ISOs: everything is included in ISOs, offline build is supported. The ISOs are available in http://www.nxp.com
- VM image: the Yocto build environment and standalone build environment is setup in the VM image, the VM images are available in http://www.nxp.com
- NXP public GIT: the source code of NXP packages and Yocto layers are published in http://git.freescale.com
- Yocto community GIT: the NXP layers of free bits are upstreamed in http://git.yoctoproject.org/cgit/cgit.cgi/

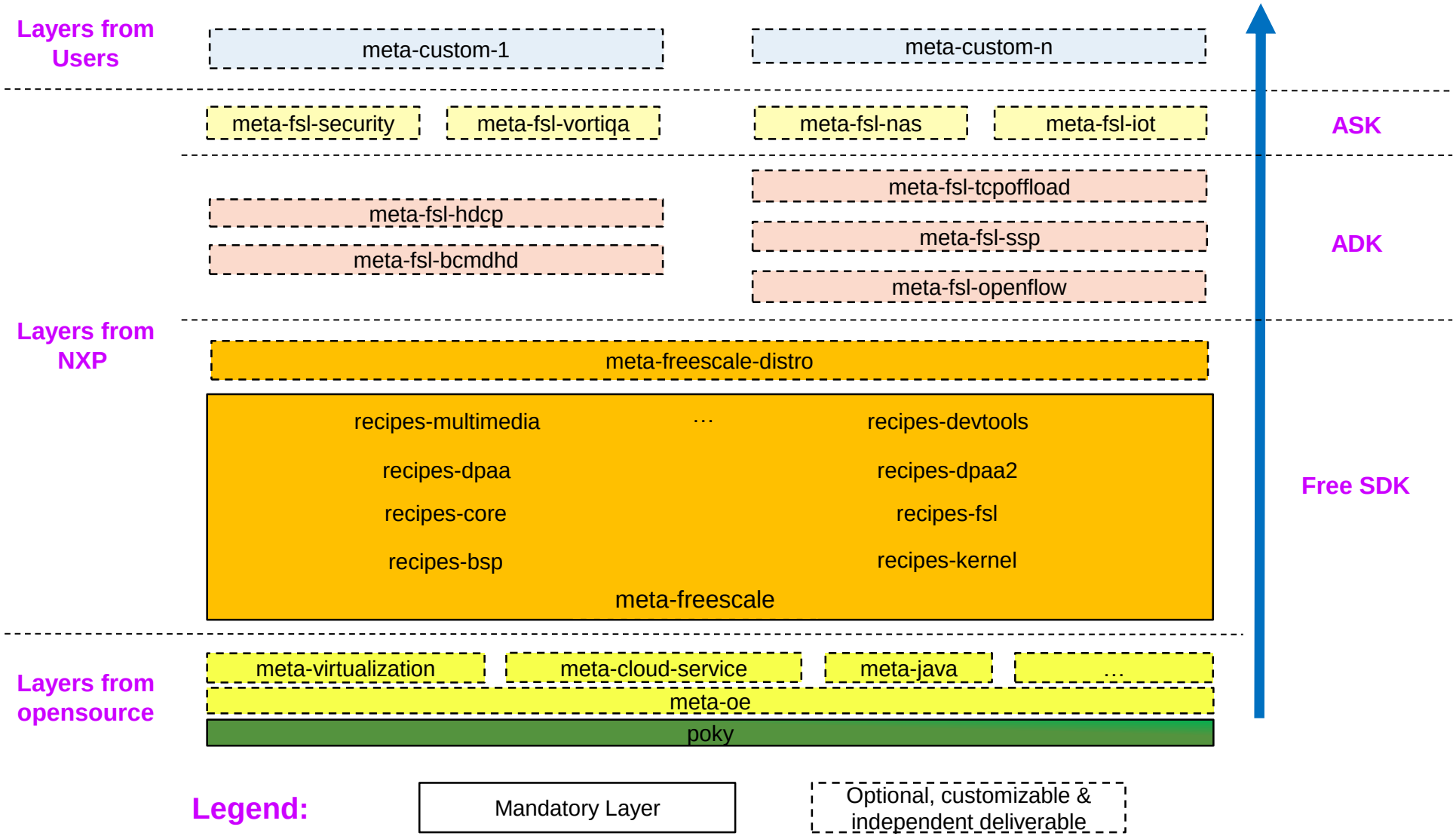
QorIQ SDK Delivery Mechanism: Software Model (1)

- Multi-tiered software offering
  - **SDK: (Software Development Kit)** Enablement offering (no-charge) includes general Linux functionality, performance enhanced, with networking (or target market) differentiation.
  - **ADK: (Application Development Kit)** Commercial offering (paid), sits on top of SDK, adds differentiating capability (typically targeted at a function or capability - additional features, Integration, optimization that reduces customer time to market for specific vertical application.)
  - **ASK: (Application Solutions Kit)** Complete SW offering (paid) – typically 80%+ of software required for a specific application – “Broadband Home Router” in a box.
- SDK is released via NXP unified layer, ADK and ASK are delivered through separated layers which can be plugged in free SDK.

QorIQ SDK Delivery Mechanism: Software Model (2)



Yocto Project Introduction: Software Model (3) for QorIQ



QorIQ SDK Delivery Mechanism: Free SDK Release History

- SDK 1.4 is based off Yocto 1.4 "dylan" release
- SDK 1.5 is based off Yocto 1.4.1 “dylan” release
- SDK 1.6 is based off Yocto 1.6 “daisy” release
- SDK 1.7 is based off Yocto 1.6.1 “daisy” release
- SDK 1.8 is based off Yocto 1.6.1 “daisy” release
- SDK 1.9 is based off Yocto 1.8.1 “fido”
  - The NXP unified layer is used for this release
- SDK 2.0 will be based off Yocto 2.0 “jethro”
  - The NXP unified layer is used
- The NXP unified layer is planned to replace meta-fsl-arm and meta-fsl-ppc with meta-freescale in Yocto community in H2/2016

QorIQ SDK Delivery Mechanism: Commercial SDK

The QorIQ LS2 SDK supports the following 5 commercial ADKs which are managed by dedicated Yocto layers.

- meta-fsl-bnsp: basic network security support
- meta-fsl-openflow-base: basic openflow support
- meta-fsl-openflow-premium: premium openflow support
- meta-fsl-ssp: switch supplementary support
- meta-fsl-tcpoffload: tcp offload support

The commercial layers can be used based on the free LS2 EAR6 release. The usage instructions can be found in the README of each ADK.

QorIQ SDK Delivery Mechanism: Repo

- Repo is a tool that Google built on top of Git
- Repo helps manage multiple Git repositories
- Repo is not meant to replace Git
- The repo command is an executable Python script that you can put anywhere in your path
- SDK can be delivered via a xml file
- Repo will be used for SDK delivery in future

Repo repository and cheat sheet: Link

YOCTO PROJECT
QORIQ
SDK USAGE

Yocto Project QorIQ SDK Usage: Install SDK

- **Download the SDK ISOs from http://www.nxp.com**

- **Install SDK**
  - Mount SDK ISO
    - $ sudo mount –o loop <yocto-sdk>.iso /mnt
  - Install SDK into a specified folder
    - $ cd /mnt
    - $ ./install

Yocto Project QorIQ SDK Usage: Setup Build Environment

- **$ cd <sdk-install-dir>**
- **$ ./sources/meta-freescale/scripts/host-prepare.sh** # run if yocto runs on the host for the first time
- **$ . ./fsl-setup-env -m <machine> -j <bitbake_jobs> -t <make_threads> -l**
  - Examples:
  - $ . ./fsl-setup-env -m p4080ds -j 4 -t 4 -l
  - $ . ./fsl-setup-env -m ls1021atwr -j 4 -t 4
  - $ . ./fsl-setup-env -m t2080rdb-64b

NOTE

- “-l”: deletes the work area after making packages.
- -j: the number of jobs to have the make program itself spawn during the compile stage.
- -t: the maximum number of bitbake tasks (or threads) that can be issued in parallel.
- **$ . ./fsl-setup-env –h # show the supported boards**

Yocto Project QorIQ SDK Usage: Build Images

- $ bitbake <target-image>
  - The following is the supported image.
  - fsl-image-minimal: contains basic packages to boot up a board.
  - fsl-image-core: contains common open source packages and FSLspecific packages.
  - fsl-image-virt: contains packages for virtualization support.
  - fsl-image-mfgtool: contains all the user space apps needed to deploy the
  - fsl-image-full image to a USB stick, hard drive, or other large physical media, it contains all packages in the full package list
  - fsl-toolchain: the cross compiler binary package.
  - package-name(u-boot): build a specific package.

NOTE: the images will be put in build_<board>/tmp/deploy/images/<machine>.

Yocto Project QorIQ SDK Usage: Configure and Rebuild u-boot

- **Modify u-boot source code**
  - $ bitbake -c patch u-boot
  - $ cd tmp/work/<board>-fsl-linux/u-boot-qoriq/<ver>/git/ and do change
- **Modify u-boot config**
  - $ modify UBOOT_MACHINES in sources/meta-freescale/conf/machine/<machine>.conf
    - e.g. UBOOT_CONFIG = “sdcard nor”
- **Rebuild u-boot image**
  - $ cd build_<machine>
  - $ bitbake -c compile -f u-boot
  - $ bitbake u-boot
- **Note**: u-boot image can be found in build_<board>/tmp/deploy/images/<machine>

Yocto Project QorIQ SDK Usage: Configure and Rebuild Kernel (1)

- **Modify kernel source code**
  - $ bitbake -c patch virtual/kernel
  - $ cd tmp/work/<board>-fsl-linux/linux-qoriq/<ver>/git/ and do change
- **Modify kernel defconfig**
  - $ modify KERNEL_DEFCONFIG in sources/meta-freescale/conf/machine/<machine>.conf
- **Change kernel dtb**
  - $ update KERNEL_DEVICETREE in sources/meta-freescale/conf/machine/<machine>.conf

Yocto Project QorIQ SDK Usage: Configure and Rebuild Kernel (2)

- **Do menuconfig**
  - $ bitbake -c menuconfig virtual/kernel

  NOTE: terminal type can be configured by defining OE_TERMINAL.
  - E.g. OE_TERMINAL = "screen"
- **Rebuild kernel image**
  - $ cd build_<machine>
  - $ bitbake -c compile -f virtual/kernel
  - $ bitbake virtual/kernel
- **NOTE**: kernel images can be found in build_<board>/tmp/deploy/images/<machine>/

Yocto Project QorIQ SDK Usage: Build Custom RootFS (1)

- **Packages in rootfs can be customized by editing corresponding recipe**
  - fsl-image-mfgtool: meta-freescale/recipes-fsl/images/fsl-image-mfgtool.bb
  - fsl-image-core: meta-freescale/recipes-fsl/images/fsl-image-core.bb
  - fsl-image-full: meta-freescale/recipes-fsl/images/fsl-image-full.bb
  - fsl-image-minimal: meta-freescale/recipes-fsl/images/fsl-image-minimal.bb
  - fsl-image-virt: meta-freescale/recipes-fsl/images/fsl-image-virt.bb
  - fsl-toolchain: meta-freescale/recipes-fsl/images/fsl-toolchain.bb
- **The rootfs type can be customized by setting IMAGE_FSTYPES variable in above recipes**
  - Supported rootfs type: jffs2 cramfs ext2 ext2.gz ext2.gz.u-boot ext2.bz2.u-boot ext3 ext3.gz.u-boot ext2.lzma live squashfs squashfs lzma ubi tar tar.gz tar.bz2 tar.xz cpio cpio.gz cpio.xz cpio.lzma, etc.

Yocto Project QorIQ SDK Usage: Build Custom RootFS (2)

- **Where are the package source and patches?**
  - The package source tarball is in <sdk-install-dir>/downloads folder
  - The patches are in corresponding recipe folder of each package.
- **How to specify the preferred version of package?**
  - PREFERRED_VERSION_<pkg> is used to configure the required version of a package.
  - If PREFERRED_VERSION is not defined for package, Yocto will build the newest version.
    - For example: downgrade samba from 3.4.0 to 3.1.0
    - Add PREFERRED_VERSION_samba = “3.1.0” in meta-freescale/conf/machine/<machine>.conf
- **Rebuild rootfs**
  - $ bitbake <target-rootfs>

Yocto Project QorIQ SDK Usage: Documents

Yocto documents

- External website: http://www.yoctoproject.org/documentation
- External wiki: https://wiki.yoctoproject.org/wiki/Main_Page
- Online SDK documents: https://freescale.sdlproducts.com/LiveContent/web/pub.xql?c=t&action=home&pub=QorIQ_SDK_1.9&lang=en-US
- Documents in SDK ISO: SDK UM, Yocto Reference Manual

QORIQ
SDK INSTALLER

QorIQ SDK Installer: Background

- GUI for managing Yocto SDK deployment is a customer requirement
- Yocto is a command line driven process
  - Need an alternative for this complex and error prone solution
- Hard to customize QorIQ Linux SDK content
  - Need a simplified alternative for installation and customization
- No integration with existing NXP CodeWarrior offering

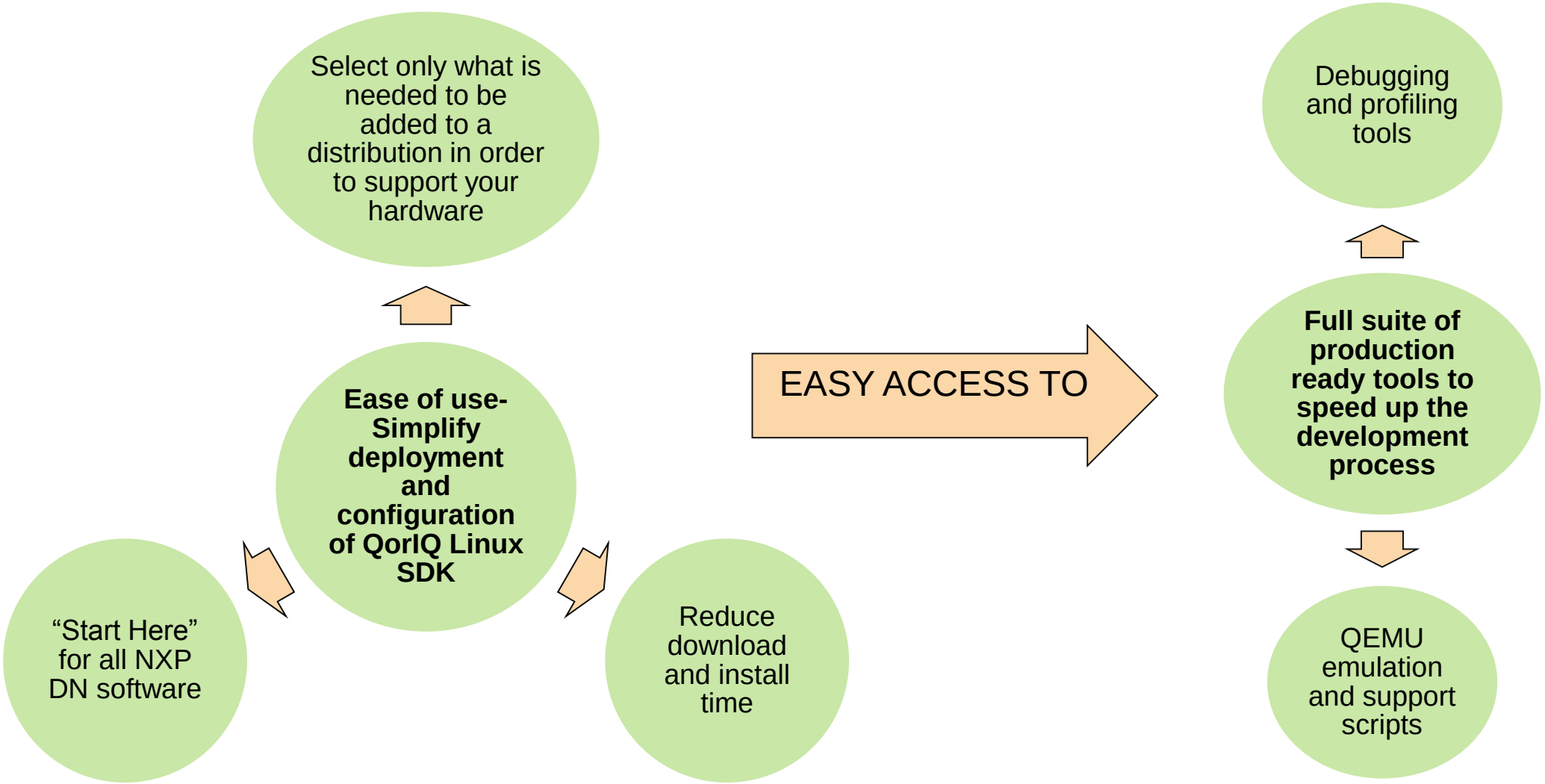
QorIQ SDK Installer: Customer benefits (1)

- **Simplify downloading the desired NXP SDK**
  - Present up to date and relevant information (version, description, etc)
  - Package integrity checks
- **Increase usability by providing options for**
  - Using either the official SDK location or 3rd party ones
  - Allow easy-install by way of SDK ISO images
  - Allow custom install via git repositories hosting SDK layers:
    - User selection for a subset of the SDK (layers)
    - Reduced total size due to only bringing desired components
    - Faster download due to size reduction and possibility of parallelization

QorIQ SDK Installer: Customer benefits (2)

- **Friendly UI for easy-use**
- **Perform requirements check**
  - Verify if dependencies are installed
  - Offer to install dependencies if missing
  - Check permissions (non-root, apt-get access)
- **Automatic host-prepare invocation**
- **Allow users to set the current SDK as CW ADT default**

QorIQ SDK Installer: Customer Benefits (3)



QorIQ SDK Installer: Install, Configure and Build With Codewarrior (1)

From command line to GUI

To install the repo tool

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ PATH=${PATH}:~/bin
```

To setup i.MX 3.14.28-1.0.0 GA release or QorIQ 1.7 release

```
$ mkdir fsl-yocto-bsp
$ cd fsl-yocto-bsp
$ repo init -u git://git.freescale.com/yocto/fsl-yocto-repo.git -b yocto-fsl-20150622
$ repo sync
```

- The machine name tells setup script which board you will be using for development

```
$ MACHINE=<machine-name> source fsl-setup-env.sh -b <build-directory>
```

- Example using Freescale i.MX6Q SABRE Auto and Framebuffer

```
$ DISTRO=imx-release-fb MACHINE=imx6qsabreauto source fsl-setup-env.sh -b build-fb
```

- Need to know by heart all the commands
- Need to configure the repo tool
- Need to create the directories structure

- Manually run the configuration Scripts
- Type the machine name – error prone

QorIQ SDK Installer: Install, Configure and Build With Codewarrior (2)

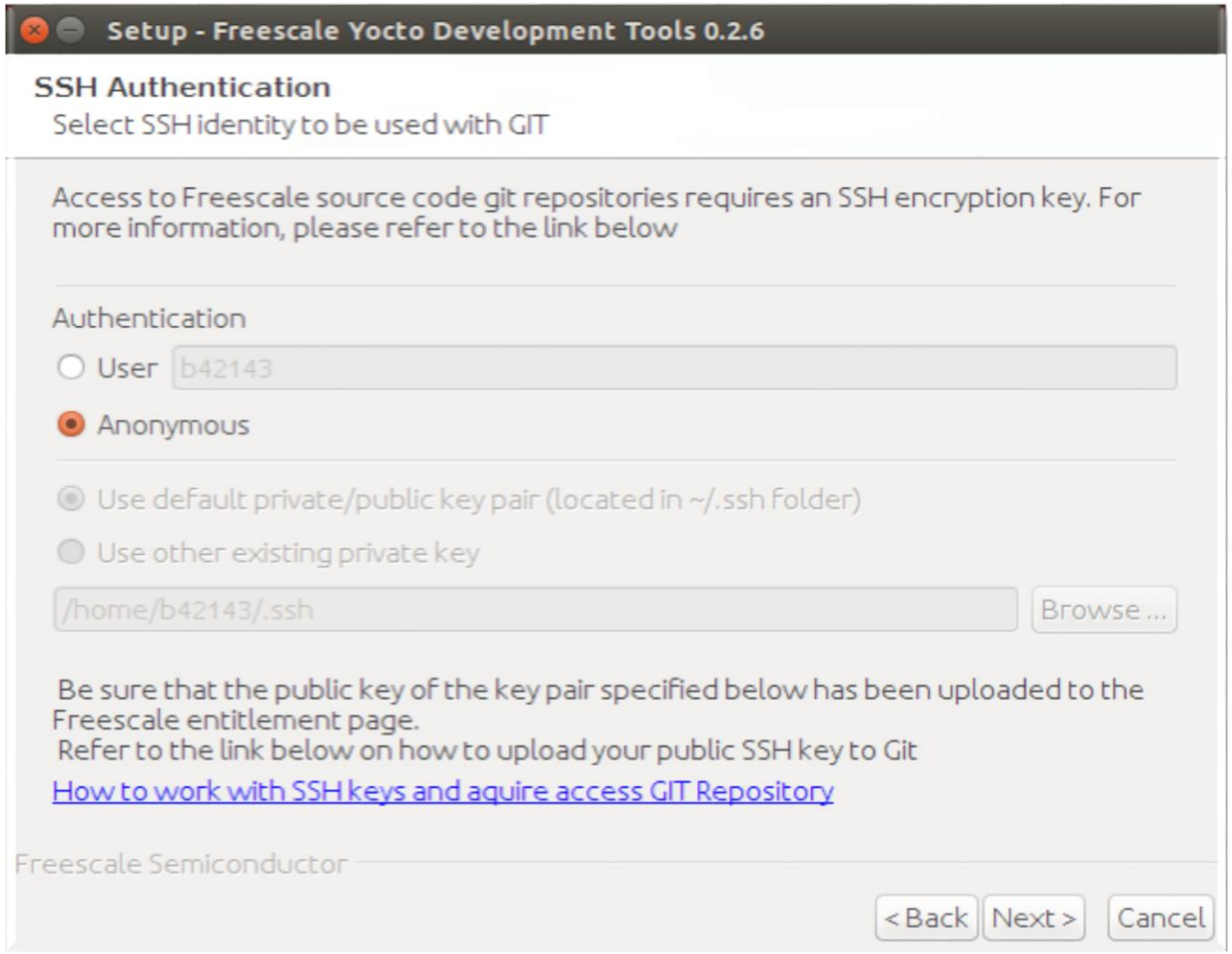
From command line to GUI

```
<?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://git.openembedded.org" name="oe" />
  <remote fetch="git://git.freescale.com/imx" name="fsl-release" />
  <remote fetch="git://github.com/OSSystems" name="OSSystems"/>

  <project remote="yocto" revision="bee7e3756adf70edaeabe9d43166707aab84f581" name="poky" path="sources/poky"/>
  <project remote="yocto" revision="af392c22bf6b563525ede4a81b6755ff1dd2c1c6" name="meta-fsl-arm" path="sources/meta-fsl-arm"/>

  <project remote="oe" revision="eb4563b83be0a57ede4269ab19688af6baa62cd2" name="meta-openembedded" path="sources/meta-openembedded"/>

  <project remote="freescale" revision="6bc2400f3045e27dc1a4a65cb28bfb0e32403bb7" name="fsl-community-bsp-base" path="sources/base">
      <copyfile dest="README" src="README"/>
      <copyfile dest="setup-environment" src="setup-environment"/>
  </project>

  <project remote="freescale" revision="07ad83db0fb67c5023bd627a61efb7f474c52622" name="meta-fsl-arm-extra" path="sources/meta-fsl-arm
  <project remote="freescale" revision="5a12677ad000a926d23c444266722a778ea228a7" name="meta-fsl-demos" path="sources/meta-fsl-demos"/

  <project remote="OSSystems" revision="fc3969f63bda343c38c40a23f746c560c4735f3e" name="meta-browser" path="sources/meta-browser" />

  <project remote="fsl-release" name="meta-fsl-bsp-release" path="sources/meta-fsl-bsp-release" revision="dora_3.10.17-1.0.0_GA">
      <copyfile src="imx/tools/fsl-setup-release.sh" dest="fsl-setup-release.sh" />
  </project>
  </manifest>
```

Manually modify the manifest file if not all the layers need to be used.

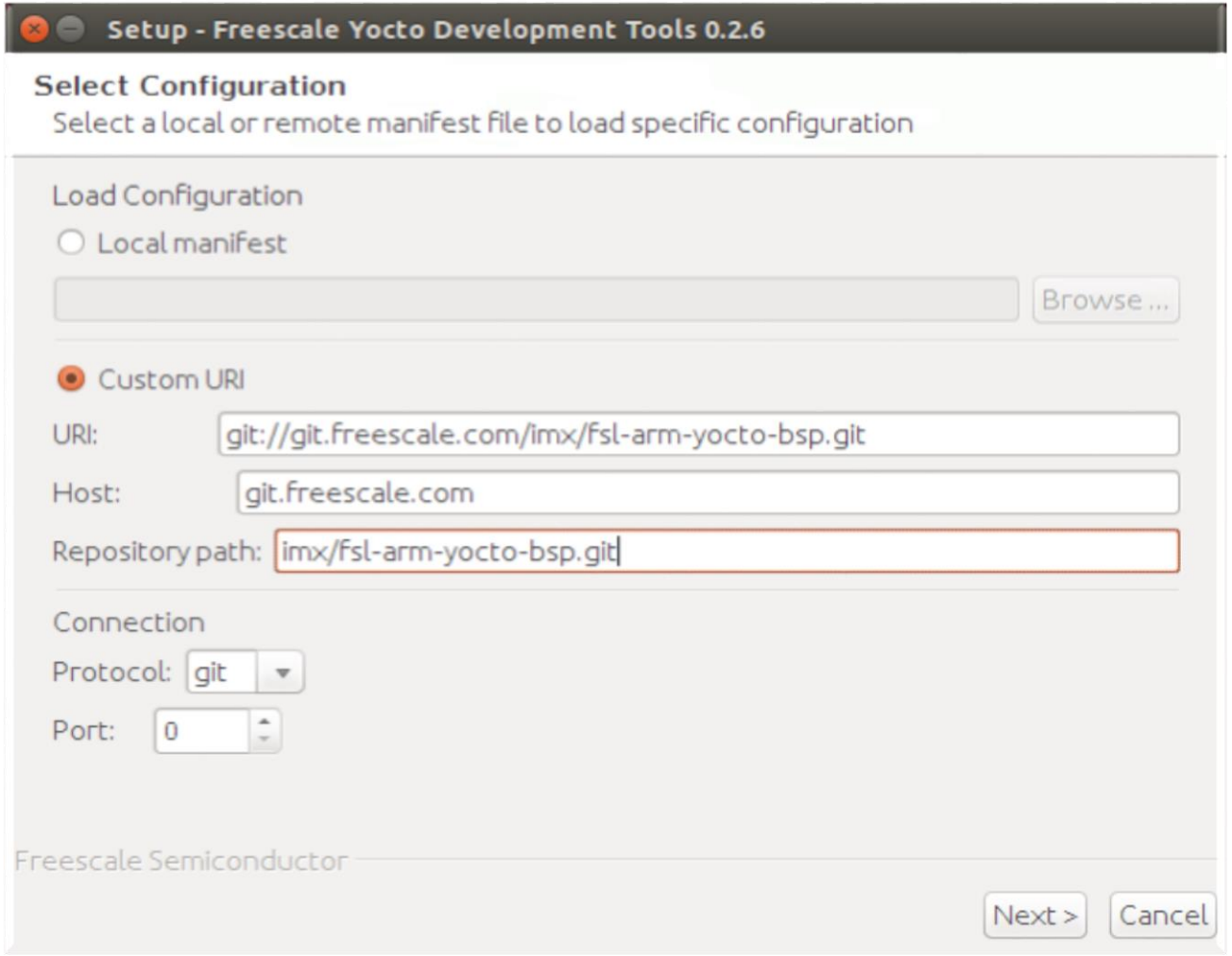
QorIQ SDK Installer: SDK installer GUI (1)

- Secure access to commercial differentiating capabilities
- Log on NXP portal to register and obtain access to those layers



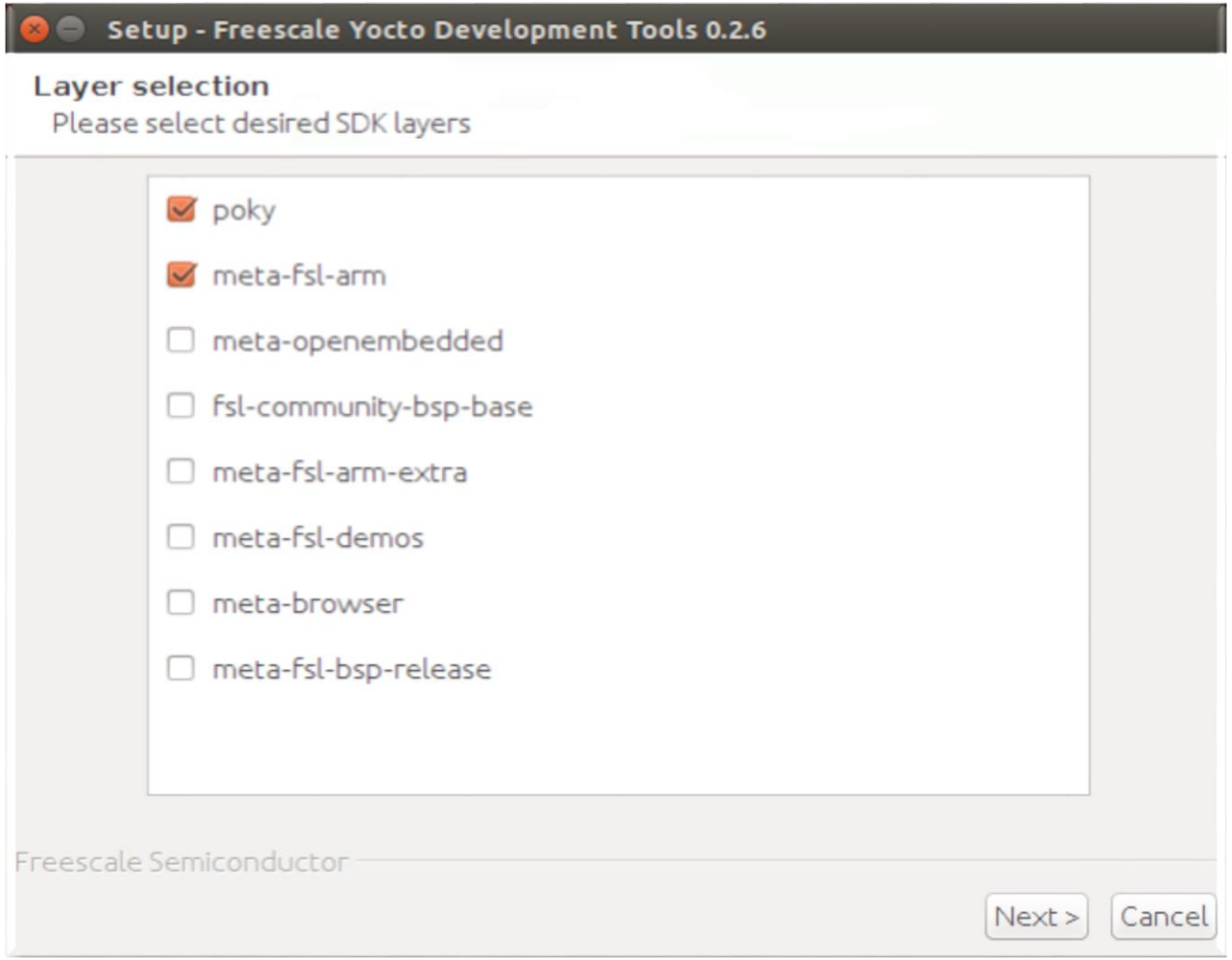
QorIQ SDK Installer: SDK installer GUI (2)

- Select a local custom manifest or use a remote repository
- Multiple options for the protocol to be used (git, http)



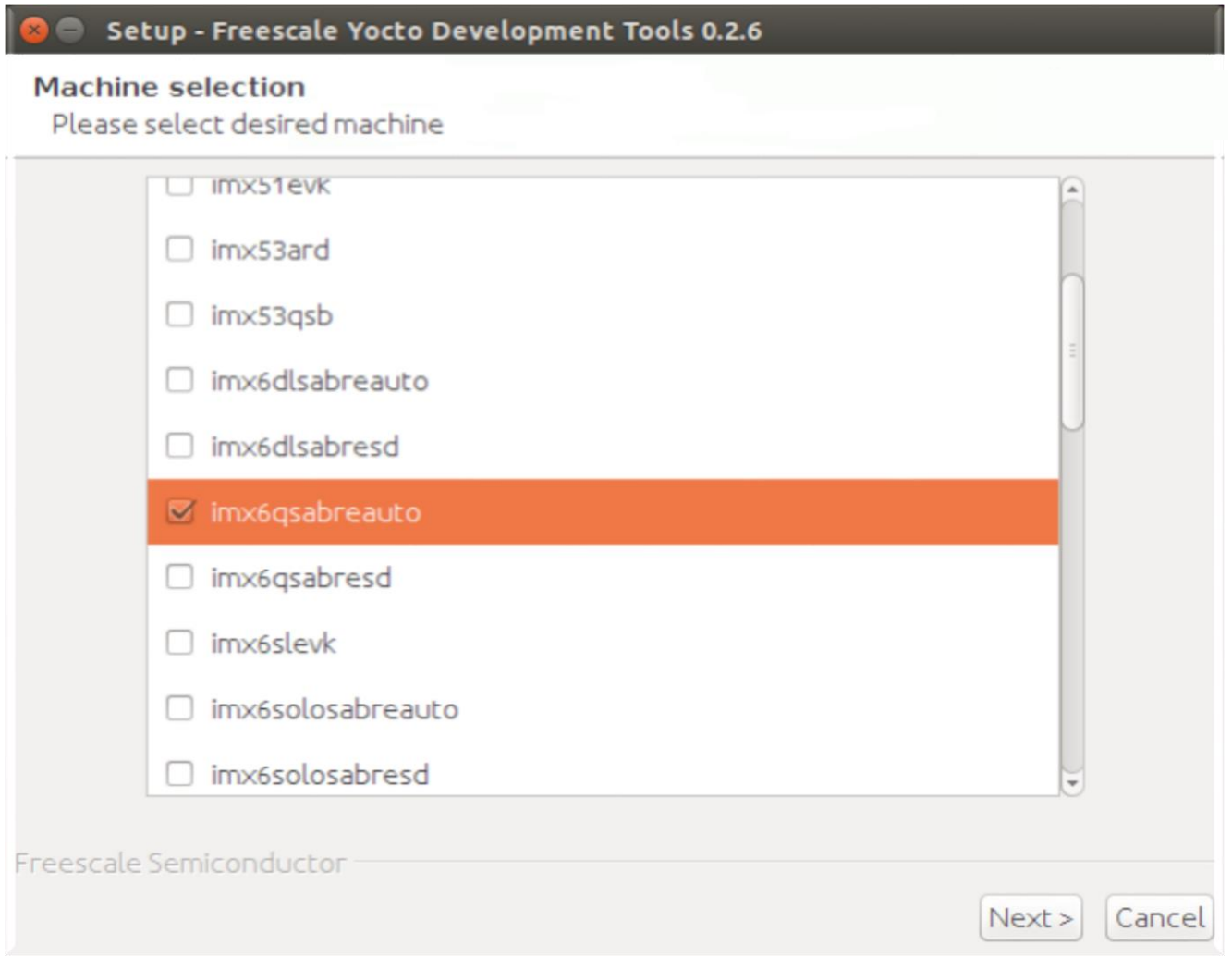
QorIQ SDK Installer: SDK Installer GUI (3)

- GUI listing of the layers to be installed based on parsing the repo manifest
- Allows selection for the needed layers only



QorIQ SDK Installer: SDK Installer GUI (4)

- GUI listing of the configurations available
- Allow selection for the configuration to be built

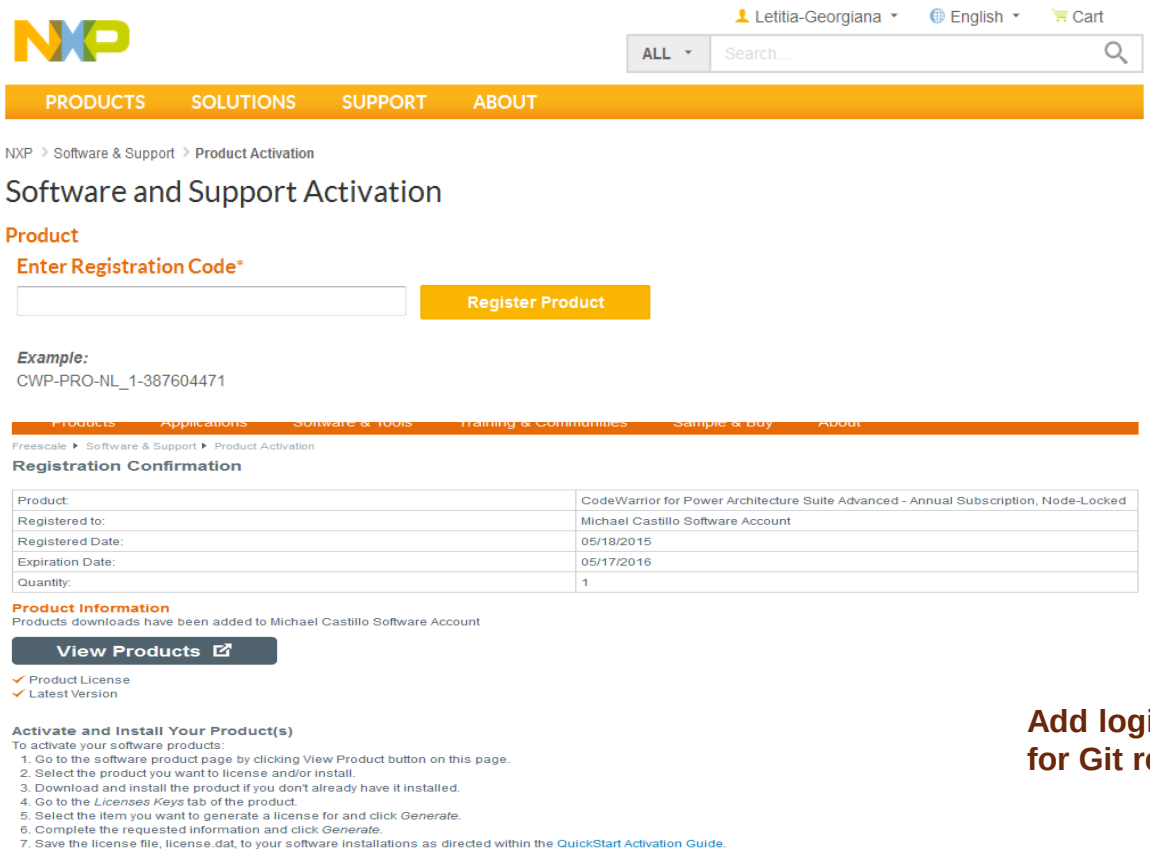


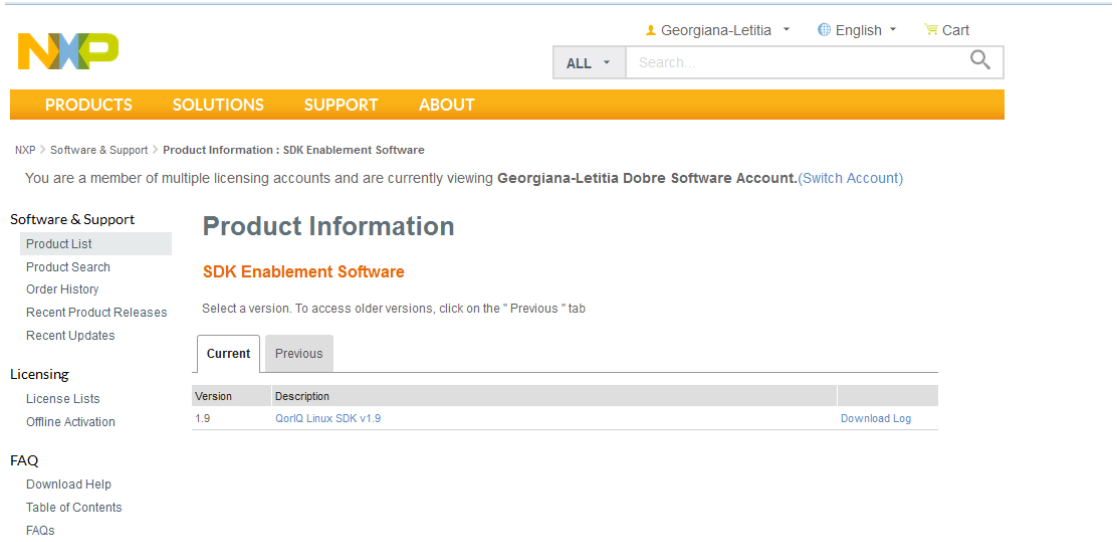
QorIQ SDK Installer: Business Perspective (1)

Create strong relationship with open source community and partners

QorIQ SDK Installer: Business Perspective

Subscribe for access to a specific package –know the customers through their feedback

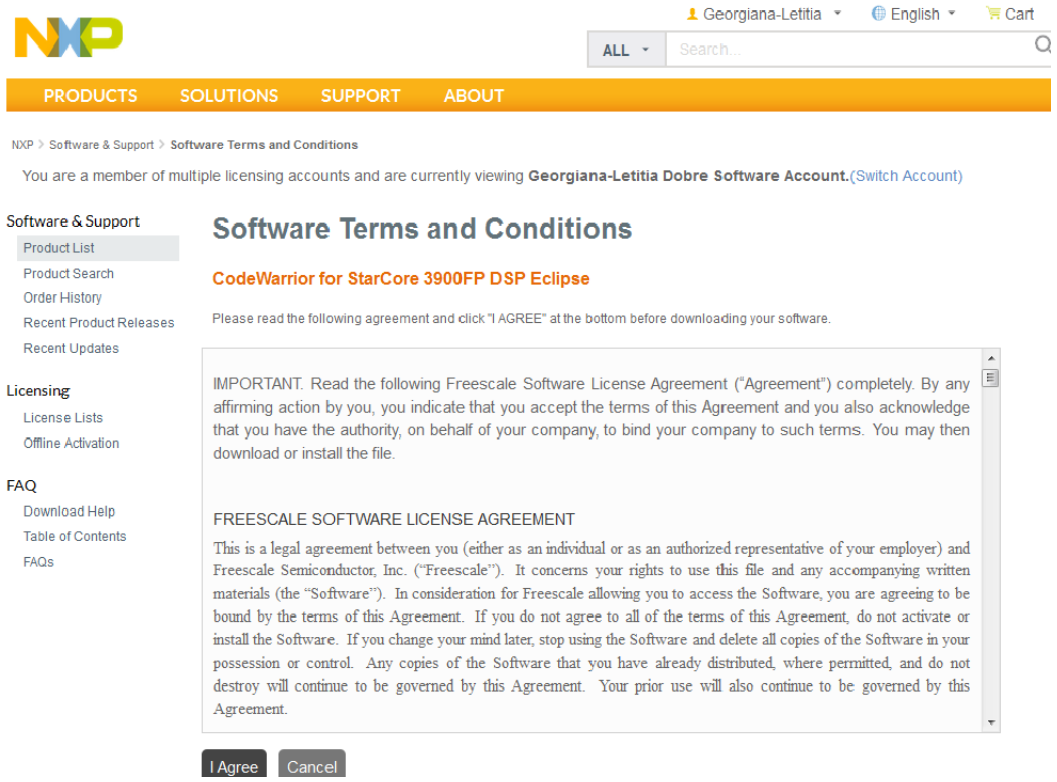


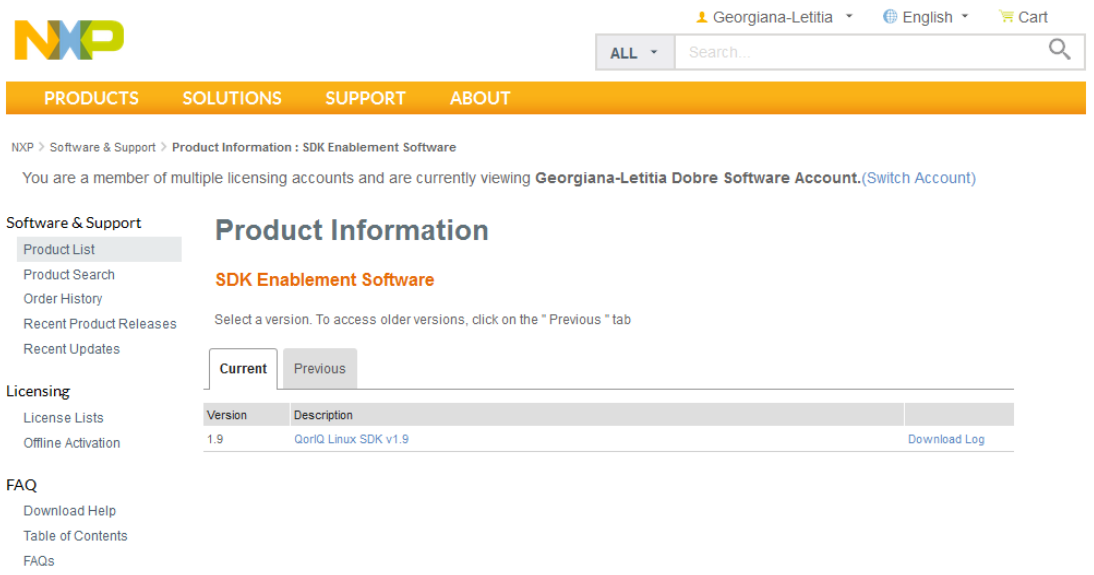


Add logic to customize the message for Git repository software instructions

QorIQ SDK Installer: Registration and Licensing

- Directs user to sign EULA if required
- Once signed the user is presented with available files.
- Only one file will be present providing the instructions on where to go to submit the SSH Key
- Once submitted the document will provide instructions to the Git repository





Add documentation for ssh keys generation and submission

QorIQ SDK Installer: Authentication (1)

- Screen will be available to submit SSH Key
- Once SSH Key is submitted, installer can identify if the user has access to an active entitlement that includes a Git repository.
- Assign the user's SSH key to the NXP Flexera GIT repositories if not already there
- Authorize user access to the specific GIT repository
- Return links to the active Git repositories for the user.
  For example: $ git clone ssh://sw@git.freescale.com/ppc/sw1.git
- When new SW is purchased and the SSH key already exists
- Middleware copies SSH key over to GIT
- If a user wants to change their SSH Key, the activation screens will include a method to do this.

QorIQ SDK Installer: Challenges

- NXP IT infrastructure support
- Integrating tools from different communities
- License mixing – combine enablement offering (no-charge) - SDK with commercial offering requiring different types of license agreements
- Enable layering NPIs over Community
- Enable layering NPIs over NXP BSPs

SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT