Document Number:IMXLXYOCTOUG
Rev L3.10.17_1.0.0-ga, 05/2014

Freescale Yocto Project User's Guide

Contents

1 Overview

The Yocto Project is an open-source collaboration focused on embedded Linux development. For more information regarding Yocto Project, see the Yocto Project page: www.yoctoproject.org/.

The Freescale Yocto Project Community BSP is a development community outside of Freescale providing support for i.MX boards on the Yocto Project environment. Freescale i.MX joined the Yocto Project community providing a release based on the Yocto Project framework . This document describes how to build an image for an i.MX Freescale board by using a Yocto Project build environment.

For this release, Freescale provides an additional layer called the Freescale BSP Release, known as meta-fsl-bsp-release, to integrate a new Freescale release with the Freescale Yocto Project Community BSP. References to community are for all layers in Yocto Project except the meta-fsl-bsp-release. The meta-fsl-bsp-release layer 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-fsl-arm and meta-fsl-demos layers in the Yocto Project, in addition to a stable release such as Yocto Project 1.5.

Release L3.10.17_1.0.0 GA is released for Yocto Project 1.5 (Dora). The same recipes for Yocto Project 1.5 are going to be upstreamed to be available on Yocto Project release 1.6. The Yocto Project release cycle lasts roughly six months.

For releases subsequent to 1.5, the meta-fsl-bsp-release layer is not required. The Freescale Yocto Project Community BSP will provide support. For example, imx53qsb and imx28evk are some Freescale machines supported in the Freescale Yocto Project Community BSP . The community also has non Freescale machine configurations. All machine references in this document are related to the Freescale machine configuration files only.

The contents of the Freescale BSP Release layer are recipes and machine configurations. Yocto Project recipes contain the mechanism to build and package a component. In many cases, other layers implement recipes or include files and the Freescale release layer provides updates to the recipes by either appending to a current recipe, or including a component and updating with patches or source locations. Most Freescale release layer recipes are very small because they use what the community has provided and update what is needed for each new package version that is unavailable in the other layers.

Freescale i.MX boards are configured in the meta-fsl-arm layer. This includes U-Boot, kernel, and machine specific details.

Freescale kernel and U-Boot releases are available through Freescale public git servers. However, several components are released as packages on the Freescale mirror. The package-based recipes pull from the Freescale mirror instead of a git location and build and package also. Starting with the L3.5.7_1.0.0-alpha release, packages which are released as binary are built as a hardware floating point. In a few cases, we have provided a software floating point version. The package selection is determined by using the DEFAULTTUNE setting.

Freescale also provides image recipes that include all the components needed for a system image to boot. Components can be built individually or through an image recipe which pulls in all the components required in an image into one build process.

The architecture of the Freescale Release Layer is as follows:

Freescale release layer

- meta-fsl-bsp-release
    - updates for meta-fsl-arm, poky, and meta-openembedded layers
    - updates for meta-fsl-demos

Yocto Project community layer

- meta-fsl-arm
- meta-fsl-demos
- meta-fsl-community-base
- meta-openembedded
- poky

1.1 End user license agreement

During the setup environment process of Freescale Yocto Project Community BSP, the Freescale i.MX End User License Agreement (EULA) is displayed. To continue, users must agree to the conditions of this license. The agreement to the terms allows the Yocto Project build to untar packages from the Freescale mirror. Please read this license agreement carefully during the setup process, because once accepted, all the further work in the Yocto Project environment is tied to this accepted agreement.

2 Features

Freescale Yocto Project Release layers have the following features:

- Linux kernel recipe
    - The kernel recipe resides in the recipes-kernel folder and integrates a Freescale kernel from the source downloaded from the Freescale git server.

    - L3.10.17_1.0.0-ga is a Linux kernel that Freescale has released only for the Yocto Project. Previous BSP releases based on Linux version 3.0.35 are released with ltib.
    - Freescale L3.10.17_1.0.0-ga supports a device tree. This change adds the device tree settings in the i.MX 6 machine configuration files.
- U-Boot recipe
    - The U-Boot recipe resides in the recipes-bsp folder and integrates a Freescale uboot-imx.git from the source downloaded from the Freecale git server.
    - Certain i.MX boards use different U-Boot versions.
    - Freescale release L3.10.17_1.0.0-ga for the i.MX 6 devices uses an updated v2013.04 Freescale version. This version has not been updated for other i.MX Freescale hardware.
    - Releases based on Linux version 2.6.35 for imx5qsb uses v2009.08 from meta-fsl-arm.
    - The Freescale Yocto Project Community BSP uses u-boot-fslc v2013.10 from the mainline but this is only supported by the community 3.0.35 kernel and will not work for the 3.10.17 kernel.
    - Freescale release L3.10.17_1.0.0-ga requires using the Freescale v2013-04 U-Boot release.
    - The Freescale Yocto Project Community BSP updates U-Boot versions frequently, so the information above might change as new U-Boot versions are integrated to meta-fsl-arm layers and updates from Freescale u-boot-imx releases are integrated into the mainline.
- Graphics recipes
    - Graphics recipes reside in recipes-graphics.
    - Graphics recipes integrate the Freescale graphics package release. For the i.MX 6 boards, the gpu-viv-bin-mx6q recipes package the graphic components for each backend – X11, frame buffer (fb), Direct Frame Buffer (directfb), Wayland backend and Weston compositor (weston).
        - Xorg-driver integrates our xserver-xorg.
        - For i.MX 5, the amd-gpu-bin provides packages for X11 and frame buffer backends.
- i.MX package recipes

  imx-lib, imx-test, and firmware-imx reside in recipes-bsp and pull from the Freescale mirror to build and package into image recipes.

- Multimedia recipes
    - Multimedia recipes reside in recipes-multimedia.
    - Recipes include libfslcodec, libfslparser, libvpuwrap, and gstreamer that pull from the Freescale mirror to build and package into image recipes.
    - Some recipes are provided for restricted codecs for which packages are not on the Freescale mirror. These packages are available separately.
- Core recipes

  Some recipes for rules, such as udev, provide updated i.MX rules to be deployed in the system. These recipes are usually updates of policy recipes and are used for customization only. Releases only provide updates if needed.

- Demo recipes

  Demonstration recipes reside in the meta-fsl-demos directory. This layer contains image recipes and recipes for customization, such as touch calibration, or recipes for demonstration applications, such as glcubes-demo on imx53qsb.

3 Host Setup

To get the Yocto Project expected behavior in a Linux Host Machine, the packages and utilities described below must be installed. An important consideration is the hard disk space required in the host machine. For example, when building on a machine running Ubuntu, the minimum hard disk space required is about 50 GB for the X11 backend. It is recommended that at least 120 GB be provided, which is enough to compile any backend.

The recommended minimum Ubuntu version is 12.04 or later. Earlier versions may cause the Yocto Project build setup to fail, because it requires python versions only available starting wtih Ubuntu 12.04.

3.1 Host packages

A Freescale Yocto Project Community BSP build requires that some packages be installed for the build that are documented under the Yocto Project.

You can go to Yocto Project Quick Start and check for the packages that must be installed for your build machine.

Essential Yocto Project host packages:

```
$ sudo apt-get install wget git-core unzip texinfo libsdl1.2-dev \
gawk diffstat build-essential chrpath
```

i.MX layers host packages for a Ubuntu 12.04 host setup:

```
$ sudo apt-get install sed cvs subversion coreutils texi2html \
docbook-utils python-pysqlite2 help2man make gcc g++ desktop-file-utils \
libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc xterm
```

3.2 Setting up the repo utility

Repo is a tool built on top of Git that makes it easier to manage projects that contain multiple repositories, which do not need to be on the same server. Repo complements very well the layered nature of the Yocto Project, making it easier for customers to add their own layers to the BSP.

To install the “repo” utility, perform these steps:

1. Create a bin folder in the home directory.

   ```
   $ mkdir ~/bin (this step may not be needed if the bin folder already exists)
   $ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo  > ~/bin/repo
   $ chmod a+x ~/bin/repo
   ```

2. Add the following line to the `.bashrc` file to ensure that the `~/bin` folder is in your PATH variable.

   ```
   export PATH=~/bin:$PATH
   ```

4 Yocto Project Setup

A Yocto Project directory contains a "sources" directory, which contains the recipes used to build, one or more build directories, and a set of scripts used to set up the environment.

The recipes used to build the project come from both the community and Freescale. The following commands are used to download the Freescale Yocto Project Community BSP recipe layers. The Yocto Project layers are downloaded and placed in the "sources" directory. This sets up the recipes that are used to build the project.

For this example, a directory called "fsl-release-bsp" is created for the project.

```
$ mkdir fsl-release-bsp
$ cd fsl-release-bsp
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ repo init -u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-3.10.17-1.0.0_ga
```

```
$ repo sync
$ MACHINE=< machine name> source fsl-setup-release.sh -b <build directory> -e <backend fb,
dfb, wayland, x11>
```

When this process is completed, the source code is checked out at `fsl-release-bsp/sources`.

You can perform repo synchronization, with the command `repo sync`, periodically to update to the latest codes.

5 Image Build

This section provides the detailed information and process for building an image.

5.1 Choosing a Freescale Yocto Project image

Yocto Project provides some images which are available on different layers. Poky provides some images, meta-fsl-arm provides others, and new image recipes are provided in the meta-fsl-bsp-release layer. New releases starting with 3.5.7 provided image recipes for each graphical back end, such as fsl-image-x11 for X11, fsl-image-fb for frame buffer, fsl-image-dfb, DirectFB, and fsl-image-weston Wayland. Each image packages a variety of applications. The following table lists various images, their contents, and the layers that provide the image recipes.

Table 1. Freescale Yocto Project images

Image name	Target	Provided by layer
core-image-minimal	A small image that only allows a device to boot.	poky
core-image-base	A console-only image that fully supports the target device hardware.	poky
core-image-sato	Image with Sato, a mobile environment and visual style for mobile devices. The image supports X11 with a Sato theme, Pimlico applications and contains terminal, editor and file manager.	poky
fsl-image-test	Builds contents core-image-base plus Freescale test applications and multimedia components.	meta-fsl-arm
fsl-image-x11	Builds contents of core-image-sato with Freescale test applications and multimedia with hardware accelerated X11.	meta-fsl-bsp-release/imx/meta-fsl-demos
fsl-image-fb	Builds a hardware accelerated Frame buffer image with graphics. Requires specific parameters on fsl-setup-release.sh -e fb.	meta-fsl-bsp-release/imx/meta-fsl-demos
fsl-image-dfb	Builds a hardware accelerated direct frame buffer image with graphics. Requires specific parameters on fsl-setup-release.sh -e dfb.	meta-fsl-bsp-release/imx/meta-fsl-demos
fsl-image-weston	Builds a hardware accelerated Wayland backend image with the Weston compositor. Requires specific parameters on fsl-setup-release.sh -e wayland.	meta-fsl-bsp-release/imx/metafsl-demos
fsl-image-manufacturing	Builds a minimal image to flash a device with the manufacturing tool.	meta-fsl-bsp-release/imx/meta-fsl-demos

Note: The Wayland graphical back end is not provided as an image. Wayland with X11 is not recommended for builds, so the Weston compositor should be used for a non-X11 image as described above.

5.2 Machine configurations

In the meta-fsl-bsp-release layer, Freescale provides new or updated machine configurations that overlay the meta-fsl-arm machine configurations. These files are copied into the `meta-fsl-arm/conf/machine` directory by the `fsl-setup-release.sh script`.

In the meta-fsl-bsp-release layer, a consolidated machine configuration is provided, imx6qdlsolo. This is for Freescale's use to provide a common image with all the device trees for i.MX 6Quad, 6DualLite, and Solo (not SoloLite) in one image. Do not use this image for build. Instead, use the machine configuration file for the target device.

The following are all the Freescale machine configuration files that can be selected:

- imx6dlsabreauto
- imx6dlsabresd
- imx6qsabreauto
- imx6qsabresd
- imx6slevk
- imx6solosabreauto
- imx6solosabresd

The command below is used to set up a directory and configuration files for the specified machine and backend.

```
$ MACHINE=<machine file> source fsl-setup-release.sh -b <build dir> -e <backend>
```

The EULA is required to be accepted for the first time. After that, the acceptance is logged and EULA acceptance is not required again.

MACHINE=<machine configuration file> is the machine file in `meta-fsl-arm/conf/machine`. The setup script checks for a valid machine. Without setting MACHINE, the setup script assumes imx6qsabresd as the default. The i.MX machine files are provided in `meta-fsl-arm/conf/machine` and `meta-fsl-bsp-release/imx/meta-fsl-arm/conf/machine`. The MACHINE configuration can also be changed in `local.conf`.

The following is a part of a `local.conf` created from the setup-environment script:

```
MACHINE ??= 'imx6qsabresd'
DISTRO ?= 'poky'

ACCEPT_FSL_EULA = "1"
```

The `fsl-setup-release.sh` script integrates the Freescale Yocto Project Release layer into the Yocto Project build by inserting the layer into the `<build dir>/conf/bblayers.conf` file.

This setup script has the following optional parameters:

- -b sets the build directory.

  `-b <build dir >`
- -e sets the graphical back end for frame buffer and direct fb images. X11 is default if no backend is set.
  - `-e fb`
  - `-e dfb`
  - `-e wayland`
  - `-e x11`

5.3 Bitbake options

The bitbake command used to build an image is `bitbake <image name>`. Additional parameters can be used for specific activities described below. Bitbake provides various useful options for developing a single component. To run with a bitbake parameter, the command looks like this:

```
bitbake  <parameter> <component>
```

`<component>` is a desired build package.

The following table provides some bitbake options.

Table 2. Bitbake options

Bitbake paramater	Description
-c fetch	Will fetch if the downloads state is not marked as done.
-c cleanall	Will clean the entire component build directory. All the changes in the build directory will be lost. The rootfs and state of the component are also cleared.
-c deploy	Will deploy an image or component to the rootfs.
-k	Continues building components even if a build break occurs.
-c compile -f	If the source is changed, the Yocto Project might not rebuild unless using this option when the state is marked as already deployed. Use this option to force a recompile after the image is deployed.
-g	Lists a dependency tree for an image or component.
-DDD	Turns on debug 3 levels deep. Each D adds another level of debug.

5.4 U-Boot configuration

Release 3.10.17_1.0.0 GA provides the U-Boot configurations in the main machine configuration file and no longer provides separate machine configruation files for each U-Boot variation. The configuration is specified by using the UBOOT_CONFIG settings.

This requires setting UBOOT_CONFIG in `local.conf`; otherwise, the U-Boot build will default to SD boot.

Release L3.10.17_1.0.0 GA provides U-Boot configuration settings in each machine configuration file. The UBOOT_CONFIG must be set; otherwise, the default SD is used. These can be built separately by using the following commands (change MACHINE to the correct target):

U-Boot type	Build setup	Build command
U-Boot EIM-NOR	`$ echo "UBOOT_CONFIG = \"eimnor\"" >> conf/local.conf`	`$ MACHINE=imx6dlsabreauto bitbake -c deploy u-boot-imx`
U-Boot SPI-NOR	`$ echo "UBOOT_CONFIG = \"spinor\"" >> conf/local.conf`	`$ MACHINE=imx6qsabreauto bitbake -c deploy u-boot-imx`
U-Boot NAND	`$ echo "UBOOT_CONFIG = \"nand\"" >> conf/local.conf`	`$ MACHINE=imx6solosabreauto bitbake -c deploy u-boot-imx`
U-Boot SATA	`$ echo "UBOOT_CONFIG = \"sata\"" >> conf/local.conf`	`$ MACHINE=imx6qsabresd bitbake -c deploy u-boot-imx`

For more information on Uboot configuration, see the *Freescale U-Boot User's Guide*.

5.5 Building an image

Yocto Project build uses the bitbake command. For example, `bitbake <component>` builds the selected component. Each component build has multiple tasks, such as fetching, configuration, compilation, packaging, and deploying to the target rootfs. The bitbake image build gathers all the components required by the image and build in order of dependency per task. The first build is the toolchain along with the tools required for the components to build.

The following command is an example on how to build an image:

```
$ bitbake <image name>
```

5.6 Build scenarios

The following are build setup scenarios for various configurations.

```
$ mkdir fsl-community-bsp
$ cd fsl-community-bsp
$ repo init -u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-3.10.17-1.0.0_ga
$ repo sync
```

5.6.1 X-11 image on i.MX 6Quad Sabre-SD

```
$ MACHINE=imx6qsabresd source fsl-setup-release.sh –b build-x11 -e x11
$ bitbake fsl-image-x11
```

If bitbake displays an error indicating that fsl-image-x11 cannot be found, this means that `fsl-setup-release.sh` was not run. This script hooks in the meta-fsl-bsp-release layer with the Yocto Project layer system. Build scripts used to generate prebuilt images for this release are provided in the meta-fsl-bsp-release layer in imx/tools. Use these for additional examples.

5.6.2 FB image on i.MX 6Quad Sabre-AI

```
$ MACHINE=imx6qsabreauto source fsl-setup-release.sh –b build-fb –e fb
$ bitbake fsl-image-fb
```

If bitbake displays an error indicating that fsl-image-fb cannot be found, this means that `fsl-setup-release.sh` was not run. This script hooks in the meta-fsl-bsp-release layer with the Yocto Project layer system. Starting from the 3.10 release, this image recipe checks for conflicts in the distro features, so if the fsl-setup-release does not configure DISTRO_FEATURES in `local.conf` without X11, DirectFB, and Wayland, the image recipe will fail with an error message.

5.6.3 DFB image on i.MX 6Solo Sabre-SD

```
$ MACHINE=imx6solosabresd source fsl-setup-release.sh –b build-dfb –e dfb
$ bitbake fsl-image-dfb
```

If bitbake shows an error indicating that fsl-image-dfb cannot be found, this means that `fsl-setup-release.sh` is not run. This script hooks in the meta-fsl-bsp-release layer with the Yocto Project layer system. Starting from the 3.10 release, this image recipe checks for conflicts in the distro features, so if the fsl-setup-release does not configure DISTRO_FEATURES in `local.conf` without X11, Wayland, and with DirectFB, the image recipe will fail with an error message.

5.6.4 Wayland image on i.MX 6Solo Sabre-SD

```
$ MACHINE=imx6solosabresd source fsl-setup-release.sh -b build-wayland -e wayland
$ bitbake fsl-image-weston
```

This image recipe checks for conflicts in the distro features, so if the fsl-setup-release does not configure DISTRO_FEATURES in `local.conf` without X11 or DirectFB, and with Wayland, the image recipe will fail with an error message.

5.6.5 Restarting a build environment

If a new terminal window is opened or the machine is rebooted after a build directory is set up, the setup environment script should be used to set up the environment variables and run a build again. The full `fsl-setup-release.sh` is not needed.

```
source setup-environment <build-dir>
```

5.6.6 Chromium browser on X11

This release provides a Chromium recipe for X11 for integrated hardware accelerated rendering. This section describes how to integrate Chromium into your rootfs and enable hardware accelerated rendering of WebGL. The Chromium browser requires additional layers added in the `fsl-release-setup.sh` script.

In `local.conf`, you can perform the following operations:

- Add Chromium into your list.

  ```
  CORE_IMAGE_EXTRA_INSTALL += "chromium"
  ```
- Add the commercial white list into `local.conf`.

  ```
  LICENSE_FLAGS_WHITELIST="commercial"
  ```
- To enable rendering of WebGL, perform the following steps after boot, and add the following lines to `/etc/profile or /usr/bin/google-chrome`.

  ```
  export LD_PRELOAD=/usr/lib/libGAL.so:/usr/lib/libEGL.so:/usr/lib/libGLSLC.so:/usr/lib/
  libCLC.so:/usr/lib/libGLESv2.so
  ```

- Start the Chromium browser.

  ```
  $google-chrome --use-gl=egl
  ```

5.6.7 Building kernel with other defconfigs

The kernel linux-imx supports building with other defconfigs. The default defconfig is imx_v7_defconfig in the `arch/arm/configs` path. If another defconfig by another name in the `arch/arm/configs` path is to be used, add `FSL_KERNEL_DEFCONFIG = "newname_defconfig"`, without the path needed in `local.conf`. The 3.10.17 linux-imx recipe will look for this defconfig and use it instead of the default defconfig. The manufacturing defconfig is

imx_v7_mfg_defconfig and this is required for the fsl-image-manufacturing image for the manufacturing tools support. The final Kernel build will have the name of the defconfig used for the Kernel build appended to uImage and deployed into the images directory. The default uImage name will also be deployed.

6 Image Deployment

Once a build is complete, the created image resides in `<build directory>/tmp/deploy/images`. An image is specific to the machine set in the environment setup. Each image build creates a U-Boot, kernel, and image type based on the IMAGE_FSTYPES defined in the machine configuration file. Most machine configurations provide an SD card image, ext3 and tar.bz2. The ext3 is the root file system.

6.1 Flashing an SD card image

An SD card image provides the full system to boot with U-Boot and kernel. To flash an SD card image, run the following command:

```
$ sudo dd if=<image name>.sdcard  of=/dev/sd<partition> bs=1M && sync
```

You can also use the `.gz` file directly:

```
$ gunzip –c <image name>.sdcard.gz | sudo dd of=/dev/sd<partition> bs=1M
```

6.2 Flashing rootfs only

To flash rootfs, run the following commands:

```
$ sudo mount /dev/sd<partition> /mnt/card
$ sudo tar --numeric-owner –jxf <image name>.tar.bz2 -C /mnt/card
```

6.3 MFGTool

A manufacturing tool kernel is built by using the imx_v7_mfg_defconfig while the default kernel is built by using the imx_v7_defconfig.

For more details on how to use the manufacturing tool, go to "How to Enter Serial Download Mode for MFGTool" in the i.MX User Guide.

Appendix A Frequently Asked Questions

A.1 Local configuration tuning

A Yocto Project build can take considerable build resources both in time and disk usage, especially when building in multiple build directories. There are methods to optimize this, for example, use a shared sstate cache and downloads directory. These can be set at any location in `local.conf`.

```
DL_DIR="/opt/freescale/yocto/imx/download"
SSTATE_DIR="/opt/freescale/yocto/imx/sstate-cache"
```

These directories should have appropriate permissions. The shared sstate helps when multiple build directories are set, each of which uses a shared cache to minimize the build time. A shared download directory minimizes the fetch time. Without these settings, Yocto Project defaults to the build directory for the sstate cache and downloads.

Every package downloaded in the DL_DIR directory is marked with a `<package name>.done`. To avoid fetching, touch the `.done` file for the package name.

A.2 Recipes

Each component is built by using a recipe. For new components, a recipe must be created to point to the source (SRC_URI) and specify patches, if applicable. The Yocto Project environment builds from a makefile in the location specified by the SRC_URI in the recipe. When a build is established from auto tools, a recipe should inherit autotools and pkgconfig. Makefiles must allow CC to be overridden by Cross Compile tools to get the package built with Yocto Project.

Some components have recipes but need additional patches or updates. This can be accomplished by using a bbappend recipe. This appends to an existing recipe details about the updated source. For example, a bbappend recipe to include a new patch should have the following contents:

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"
SRC_URI += file://<patch name>.patch
```

FILESEXTRAPATHS_prepend tells Yocto Project to look in the directory listed to find the patch listed in SRC_URI.

Tip: If a bbappend recipe is not picked up, check the fetch log. Sometimes a git version of the recipe is being used over the version of the bbappend files.

A.3 How to select additional packages

Additional packages can be added to images as long as there is a recipe provided for that package.

A.3.1 Updating an image

An image is a set of packages and the environment configuration.

An image file (such as fsl-image-gui.bb) defines the packages that go inside the file system. Root file systems, kernels, modules, and the U-Boot binary are available in `build/tmp/deploy/images/`.

Note: You can build packages without including it in an image, but you must rebuild the image if you want the package installed automatically on a rootfs.

A.3.2 Package group

A package group is a set of packages that can be included on any image.

A package group can contain a procedure of compilation and installation for a set of packages. For example, a multimedia task could determine, according to the machine, whether the VPU package is built or not, so the selection of multimedia packages may be automated for every board supported by the BSP, and only the multimedia task is included on the images.

Additional packages can be installed by adding the following line in <build dir>/local.conf.

```
CORE_IMAGE_EXTRA_INSTALL += "<package_ name1 package_name2>"
```

A.3.3 Preferred version

The preferred version is used to specify the preferred recipe to use for a specific component. Sometimes a component might have multiple recipes in different layers and a preferred version points to a specific version to use.

In the meta-fsl-bsp-release layer, in layer.conf, preferred versions are set for all the recipes to provide a static system for a production environment. These preferred version settings are used for formal Freescale releases but are not essential for future development.

Preferred versions also help when previous versioning may cause confusion about which recipe should be used. For example, previous recipes for imx-test and imx-lib used a year-month versioning which has changed to <kernel-version> versioning. Without a preferred version, an older version might be picked up. Recipes that have _git versions are usually picked over other recipes, unless a preferred version is set. To set a preferred version, put the following in local.conf.

```
PREFERRED_VERSION_<component>_<soc family> = "<version>"
```

For example, imx-lib would be:

```
PREFERRED_VERSION_imx-lib_mx6 = "3.10.17-1.0.0"
```

A.3.4 Preferred provider

The preferred provider is used to specify the preferred provider for a specific component. A component can have multiple providers. For example, the Linux kernel can be provided by Freescale or by kernel.org and preferred provider states the provider to use.

For example, U-Boot is provided by both the community and Freescale. A community provider is u-boot-fslc. A Freescale provider is u-boot-imx. To state a preferred provider, put the following in local.conf:

```
PREFERRED_PROVIDER_<component>_<soc family> = "<provider>"
PREFERRED_PROVIDER_u-boot_mx6 = "u-boot-imx"
```

A.3.5 SoC family

The SoC family documents a class of changes that apply to a specific set of system chips. In each machine configuration file, the machine is listed with a specific SoC family. For example, i.MX 6DualLite Sabre-SD is listed under the MX 6 and MX 6DualLite SoC families. i.MX 6Solo Sabre-auto is listed under the MX 6 and MX 6Solo SoC families. Some changes can be targeted to a specific SoC family in local.conf to override a change in a machine configuration file. The following is an example of a change to an mx6dlsabresd kernel setting.

```
KERNEL_DEVICETREE_mx6dl = "imx6dl-sabresd.dts"
```

SoC families are useful when making a change that is specific only for a class of hardware. For example, i.MX 28 EVK does not have a Video Processing Unit (VPU), so all the settings for VPU should use MX 5 or MX 6 to be specific to the right class of chips.

A.3.6 Bitbake logs

Bitbake logs the build and package processes in the temp directory in `tmp/work/<toolchain>/<component>/temp`.

If a component fails to fetch a patch, the log showing the errors is in the file log.do_fetch.

If a component fails to compile, the log showing the errors is in the file log.do_compile.

Sometimes a component does not deploy as expected. Check the package directory under the build component directory.

Appendix B References

- For details on Boot Switches, see Chapter 4 "How to Boot the i.MX Board" in the *i.MX_6_<Board>_User_Guide*.
- For how to download images by U-boot, see Section 6.2 "Downloading Images by U-Boot" in the *i.MX_6_<Board>_User_Guide*.
- For how to set up an SD/MMC card, see Chapter 7 "Using a Linux Host to Set Up an SD/MMC Card" in the *i.MX_6_<Board>_User_Guide*.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: IMXLXYOCTOUG
Rev. L3.10.17_1.0.0-ga
05/2014