# How to create your own distribution

## Contents



Stable: 19.10.2020 - 14:20 / Revision: 19.10.2020 - 14:19

Stable: 19.10.2020 - 14:20 | Revision: 19.10.2020 - 14:20

A quality version of this page, approved on *19 October 2020*, was based off this revision.

Contents

# 1 Article purpose

The purpose of this article is to describe the basic steps required to create your own distribution.

## 2 Prerequisites

OpenSTLinux distribution must be installed and into the board Flash memory(ies).

# 3 Creating your own distribution

As recommended in Yocto user manual [1], you may create your own distribution in order not to alter any original distribution Metadata, while gaining more control over package alternative selections, compile-time options, and other low-level configurations.

The basic steps for creating a distribution are detailed in the below chapter.

More details can be found in *Yocto Mega manual/Creating your own distribution*[1].

## 3.1 Creating a layer for a new distro

Please read the How to create a new open embedded layer article.

## 3.2 Creating the distribution configuration file

Some configuration examples are provided in ST distribution under: **<path of OpenSTLinux distribution delivery>/meta-st/meta-st-openstlinux/conf/distro/*.conf**

## 3.3 Providing miscellaneous variables

Some miscellaneous variable examples are provided under : **<path of OpenSTLinux distribution delivery>/meta-st/meta-st-openstlinux/conf/distro/include/st-default-distro*.inc files**

All meta-st-openstlinux distro layer configuration files presented above are located here:

```
distro
├── include
│   ├── exception-gplv3.inc
│   ├── openstlinux.inc
│   ├── st-default-distro-providers.inc
│   ├── st-default-distro-rules.inc
├── openstlinux-eglfs.conf
├── openstlinux-weston.conf
├── openstlinux-x11.conf
└── [...]
```

## 3.4 Adding more to the layer if necessary

More add-on component examples:

- recipes for installing distro-specific configuration files
- any image recipes specific to user distribution
- a *psplash append file* for a branded splash screen
- any other append files to make custom changes

Some examples of such add-on components can be found in **<path of OpenSTLinux distribution delivery>/meta-st/meta-st-openstlinux, you will retrieve some examples of these addons.**

ST has already added some recipes (*bbappend) in openstlinux-weston distribution for configuring, patching, ... (non-exhaustive list shown below):

- recipes-benchmark for *glmark2*
- recipes-connectivity for *bluez5*
- recipes-core for *busybox*
- recipes-graphics for *weston-init*

...

Some other added components (*bb) are more specific: images, system services, ... (non-exhaustive list shown below):

- recipes-core for *psplash screen*, *systemd services*
- recipes-samples for example images
- recipes-security for *OP-TEE userland part*

...

## 3.5 Use of meta-st-stm32mp with a core image

If you want to use the meta-st-stm32mp layer with a core image (nodistro mode), please apply the following steps to manage the dependencies between layers:

```
PC $> source layers/openembedded-core/oe-init-build-env
PC $> bitbake-layers add-layer ../layers/meta-openembedded/meta-oe
PC $> bitbake-layers add-layer ../layers/meta-openembedded/meta-python
PC $> bitbake-layers add-layer ../layers/meta-st/meta-st-stm32mp
PC $> bitbake core-image-base or bitbake core-image-minimal
```

# 4 Reference list

- 1.01.1 Yocto Megamanual Creating your own distribution

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Open Portable Trusted Execution Environment
Stable: 19.10.2020 - 14:17 / Revision: 19.10.2020 - 14:16

A quality version of this page, approved on *19 October 2020*, was based off this revision.

This article describes how to create and add a new layer to your STM32MPU Embedded Software Distribution Package for any development platform of the STM32MP1 family (for example an STM32MP157 Evaluation board). This can be done in order to modify some of its installed software, or to add new applications. It first explains the OpenEmbedded principle for customization by distribution, and then gives a step-by-step approach to creating a new layer and adding it it to your distribution. Finally, it gives guidelines on how to upgrade (add, remove, configure, improve...) any piece of installed software.

Contents



# 1 Why create a layer?

The OpenEmbedded distribution comes with a set of layers that provide the different pieces of software used to build images.

Checking the layers included in your distribution is easy with the **bitbake-layers** command tool:

- From your top directory, source your BitBake env setup (if not already done):

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- List your layers and priority:

```
$ bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta                  /local/openstlinux-18-01-23/layers/openembedded-core/meta  5
meta-oe               /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe  6
...
```

When you want to integrate or modify a distribution , the **OpenEmbedded principle is to append modifications/addons in a dedicated place** outside of the already available distribution layers.
This place is a new layer that you add to the distribution layers and feed with your modifications or add-ons. Hence the first thing to do when you want to integrate your modifications into a distribution is to create a new layer.

## 2 How to proceed

## 2.1 Tools

The openembedded-core layer provides a utility tool called **bitbake-layers**, which performs actions associated with layers. It allows you to perform actions including the following:

- see the list of actual layers taken into account in your configuration
- add existing layers to your configuration
- create a new layer

For an exhaustive list of options and subcommands:

- From your top directory, source your BitBake env setup (if not already done):

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- Display bitbake-layers help:

```
$ bitbake-layers --help
NOTE: Starting bitbake server...
usage: bitbake-layers [-d] [-q] [-F] [--color COLOR] [-h] <subcommand> ...

BitBake layers utility

optional arguments:
  -d, --debug           Enable debug output
  -q, --quiet           Print only errors
  -F, --force           Force add without recipe parse verification
  --color COLOR         Colorize output (where COLOR is auto, always, never)
  -h, --help            Show this help message and exit

subcommands:
  <subcommand>
    show-layers         Show current configured layers
    show-overlayed      List overlayed recipes (where the same recipe exists
                        in another layer)
    show-recipes        List available recipes, showing the layer by which they
                        are provided
    show-appends        List bbappend files and recipe files to which they apply
    show-cross-depends  Show dependencies between recipes that cross layer
                        boundaries
    add-layer           Add a layer to bblayers.conf
    remove-layer        Remove a layer from bblayers.conf
    flatten             flatten layer configuration into a separate output
                        directory
    layerindex-fetch    Fetches a layer from a layer index along with its
                        dependent layers, and adds them to conf/bblayers.conf
    layerindex-show-depends
                        Find layer dependencies from layer index.
    create-layer        Create a basic layer

Use bitbake-layers <subcommand> --help to get help on a specific command
```

## 2.2 Creating a new layer

### 2.2.1 Available layers from configuration

You may first check the list of available layers from your configuration and their priority:

- From your top directory, source your BitBake env setup (if not already done):

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- Then get the list of layers for your configuration:

```
$ bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta-oe               /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe  6
meta-gnome            /local/openstlinux-18-01-23/layers/meta-openembedded/meta-gnome  7
meta-xfce             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-xfce  7
meta-initramfs        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-initramfs
8
meta-multimedia       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
multimedia  6
meta-networking       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
networking  5
meta-webserver        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
webserver  6
meta-filesystems      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
filesystems  6
meta-perl             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-perl  6
meta-python           /local/openstlinux-18-01-23/layers/meta-openembedded/meta-python  7
meta-st-stm32mp       /local/openstlinux-18-01-23/layers/meta-st/meta-st-stm32mp  6
meta-qt5              /local/openstlinux-18-01-23/layers/meta-qt5  7
meta-st-openstlinux   /local/openstlinux-18-01-23/layers/meta-st/meta-st-openstlinux  5
meta                  /local/openstlinux-18-01-23/layers/openembedded-core/meta  5
```

**Information**

priority**: indicates the order followed by bitbake to apply rules from layers. When two layers have a '.bbappend' file based on the same recipe, the rules defined by the layer having the lowest priority are applied first, and then the rules defined by the layer having the highest priority are applied. This allows a layer with higher priority than the other layers to override any rules defined by the other layers.**

For more information about priority see the documentation: Priority doc

### 2.2.2 Create the new layer

To create a new layer you need to specify:

- the location of the new layer
- the name of the new layer
- (optionally) the priority of the new layer.

As an example, let's create the new layer *meta-my-custo-layer* on *meta-st* directory with priority set to *7*:

```
$ bitbake-layers create-layer --priority 7 ../layers/meta-st/meta-my-custo-layer
NOTE: Starting bitbake server...
Add your new layer with 'bitbake-layers add-layer ../layers/meta-st/meta-my-custo-layer'
```

```
$ cd ../layers/meta-st
$ tree meta-my-custo-layer
meta-my-custo-layer
├── conf
│   └── layer.conf
├── COPYING.MIT
├── README
└── recipes-example
    └── example
        └── example.bb

3 directories, 4 files
```

### 2.2.3 Recommended actions on a new layer

- Update the README file
- Update the priority in the **conf/layer.conf** file (if it's not the expected one )
- Apply your own modifications and/or addons for the distribition

**Information**

Note: you can refer to OpenEmbedded - devtool to learn how to add new applications, modify existing applications and more.

## 2.3 Add the new layer to your configuration

### 2.3.1 Generic case

You may first check the list of available layers from your configuration and their priority:

- From your top directory, source your BitBake env setup (if not already done):

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- Then get the list of layers for your configuration:

```
$ bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta-oe               /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe  6
meta-gnome            /local/openstlinux-18-01-23/layers/meta-openembedded/meta-gnome  7
meta-xfce             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-xfce  7
meta-initramfs        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-initramfs
8
meta-multimedia       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
multimedia  6
meta-networking       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
networking  5
meta-webserver        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
webserver  6
meta-filesystems      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
filesystems  6
meta-perl             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-perl  6
meta-python           /local/openstlinux-18-01-23/layers/meta-openembedded/meta-python  7
meta-st-stm32mp       /local/openstlinux-18-01-23/layers/meta-st/meta-st-stm32mp  6
meta-qt5              /local/openstlinux-18-01-23/layers/meta-qt5  7
meta-st-openstlinux   /local/openstlinux-18-01-23/layers/meta-st/meta-st-openstlinux  5
meta                  /local/openstlinux-18-01-23/layers/openembedded-core/meta  5
```

- You can then add the new layer *meta-my-custo-layer* to your build configuration:

**Information**

Make sure that you run the command from your build directory

```
$ bitbake-layers add-layer ../layers/meta-st/meta-my-custo-layer/
```

- Then check that it has been properly enabled for your configuration:

```
$ bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta-oe               /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe  6
meta-gnome            /local/openstlinux-18-01-23/layers/meta-openembedded/meta-gnome  7
meta-xfce             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-xfce  7
meta-initramfs        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-initramfs
8
meta-multimedia       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
```

```
multimedia  6
meta-networking       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
networking  5
meta-webserver        /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
webserver  6
meta-filesystems      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-
filesystems  6
meta-perl             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-perl  6
meta-python           /local/openstlinux-18-01-23/layers/meta-openembedded/meta-python  7
meta-st-stm32mp       /local/openstlinux-18-01-23/layers/meta-st/meta-st-stm32mp  6
meta-qt5              /local/openstlinux-18-01-23/layers/meta-qt5  7
meta-st-openstlinux   /local/openstlinux-18-01-23/layers/meta-st/meta-st-openstlinux  5
meta                  /local/openstlinux-18-01-23/layers/openembedded-core/meta  5
meta-my-custo-layer   /local/openstlinux-18-01-23/layers/meta-st/meta-my-custo-layer  7
```

### 2.3.2 With STMicroelectronics Distribution Package

Pre-requisite:

- Your custom layers are integrated in git.

To apply the new layer on top of fresh Distribution Package delivery:

- Download your layer using the *git clone* command under the Distribution Package delivery folder tree
- Source your build environment:

```
$ source ./layers/meta-st/scripts/envsetup.sh
```

- Add your layer *meta-my-custo-layer* to your build configuration

```
bitbake-layers add-layer ../layers/meta-st/meta-my-custo-layer/
```

Initial ramdisk (https://en.wikipedia.org/wiki/Initial_ramdisk)