**Chapter 30. Images**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-images)

The OpenEmbedded build system provides several example images to satisfy different needs. When you issue the bitbakecommand you provide a “top-level” recipe that essentially begins the build for the type of image you want.

**Note**

Building an image without GNU General Public License Version 3 (GPLv3), GNU Lesser General Public License Version 3 (LGPLv3), and the GNU Affero General Public License Version 3 (AGPL-3.0) components is only supported for minimal and base images. Furthermore, if you are going to build an image using non-GPLv3 and similarly licensed components, you must make the following changes in the local.conf file before using the BitBake command to build the minimal or base image:

1. Comment out the EXTRA\_IMAGE\_FEATURES line

2. Set INCOMPATIBLE\_LICENSE = "GPL-3.0 LGPL-3.0 AGPL-3.0"

From within the poky Git repository, you can use the following command to display the list of directories within the [Source Directory](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#source-directory)that containe image recipe files:

$ ls meta\*/recipes\*/images/\*.bb

Following is a list of supported recipes:

* build-appliance-image: An example virtual machine that contains all the pieces required to run builds using the build system as well as the build system itself. You can boot and run the image using either the [VMware Player](http://www.vmware.com/products/player/overview.html) or [VMware Workstation](http://www.vmware.com/products/workstation/overview.html). For more information on this image, see the [Build Appliance](http://www.yoctoproject.org/documentation/build-appliance) page on the Yocto Project website.
* core-image-base: A console-only image that fully supports the target device hardware.
* core-image-clutter: An image with support for the Open GL-based toolkit Clutter, which enables development of rich and animated graphical user interfaces.
* core-image-full-cmdline: A console-only image with more full-featured Linux system functionality installed.
* core-image-lsb: An image that conforms to the Linux Standard Base (LSB) specification. This image requires a distribution configuration that enables LSB compliance (e.g. poky-lsb). If you build core-image-lsb without that configuration, the image will not be LSB-compliant.
* core-image-lsb-dev: A core-image-lsb image that is suitable for development work using the host. The image includes headers and libraries you can use in a host development environment. This image requires a distribution configuration that enables LSB compliance (e.g. poky-lsb). If you build core-image-lsb-dev without that configuration, the image will not be LSB-compliant.
* core-image-lsb-sdk: A core-image-lsb that includes everything in the cross-toolchain but also includes development headers and libraries to form a complete standalone SDK. This image requires a distribution configuration that enables LSB compliance (e.g. poky-lsb). If you build core-image-lsb-sdk without that configuration, the image will not be LSB-compliant. This image is suitable for development using the target.
* core-image-minimal: A small image just capable of allowing a device to boot.
* core-image-minimal-dev: A core-image-minimal image suitable for development work using the host. The image includes headers and libraries you can use in a host development environment.
* core-image-minimal-initramfs: A core-image-minimal image that has the Minimal RAM-based Initial Root Filesystem (initramfs) as part of the kernel, which allows the system to find the first “init” program more efficiently. See the[PACKAGE\_INSTALL](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-PACKAGE_INSTALL) variable for additional information helpful when working with initramfs images.
* core-image-minimal-mtdutils: A core-image-minimal image that has support for the Minimal MTD Utilities, which let the user interact with the MTD subsystem in the kernel to perform operations on flash devices.
* core-image-rt: A core-image-minimal image plus a real-time test suite and tools appropriate for real-time use.
* core-image-rt-sdk: A core-image-rt image that includes everything in the cross-toolchain. The image also includes development headers and libraries to form a complete stand-alone SDK and is suitable for development using the target.
* core-image-sato: An image with Sato support, a mobile environment and visual style that works well with mobile devices. The image supports X11 with a Sato theme and applications such as a terminal, editor, file manager, media player, and so forth.
* core-image-sato-dev: A core-image-sato image suitable for development using the host. The image includes libraries needed to build applications on the device itself, testing and profiling tools, and debug symbols. This image was formerly core-image-sdk.
* core-image-sato-sdk: A core-image-sato image that includes everything in the cross-toolchain. The image also includes development headers and libraries to form a complete standalone SDK and is suitable for development using the target.
* core-image-testmaster: A "master" image designed to be used for automated runtime testing. Provides a "known good" image that is deployed to a separate partition so that you can boot into it and use it to deploy a second image to be tested. You can find more information about runtime testing in the "[Performing Automated Runtime Testing](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#performing-automated-runtime-testing)" section in the Yocto Project Development Manual.
* core-image-testmaster-initramfs: A RAM-based Initial Root Filesystem (initramfs) image tailored for use with the core-image-testmaster image.
* core-image-weston: A very basic Wayland image with a terminal. This image provides the Wayland protocol libraries and the reference Weston compositor. For more information, see the "[Wayland](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#wayland)" section.
* core-image-x11: A very basic X11 image with a terminal.

**Chapter 31. Features**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features)

[31.1. Machine Features](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features-machine)

[31.2. Distro Features](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features-distro)

[31.3. Image Features](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features-image)

[31.4. Feature Backfilling](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features-backfill)

This chapter provides a reference of shipped machine and distro features you can include as part of your image, a reference on image features you can select, and a reference on feature backfilling.

Features provide a mechanism for working out which packages should be included in the generated images. Distributions can select which features they want to support through the [DISTRO\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-DISTRO_FEATURES) variable, which is set or appended to in a distribution's configuration file such as poky.conf, poky-tiny.conf, poky-lsb.conf and so forth. Machine features are set in the[MACHINE\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-MACHINE_FEATURES) variable, which is set in the machine configuration file and specifies the hardware features for a given machine.

These two variables combine to work out which kernel modules, utilities, and other packages to include. A given distribution can support a selected subset of features so some machine features might not be included if the distribution itself does not support them.

One method you can use to determine which recipes are checking to see if a particular feature is contained or not is to grepthrough the [Metadata](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#metadata) for the feature. Here is an example that discovers the recipes whose build is potentially changed based on a given feature:

$ cd poky

$ git grep 'contains.\*MACHINE\_FEATURES.\**feature*'

**31.1. Machine Features**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features-machine)

The items below are features you can use with [MACHINE\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-MACHINE_FEATURES). Features do not have a one-to-one correspondence to packages, and they can go beyond simply controlling the installation of a package or packages. Sometimes a feature can influence how certain recipes are built. For example, a feature might determine whether a particular configure option is specified within the[do\_configure](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-tasks-configure) task for a particular recipe.

This feature list only represents features as shipped with the Yocto Project metadata:

* ***acpi:*** Hardware has ACPI (x86/x86\_64 only)
* ***alsa:*** Hardware has ALSA audio drivers
* ***apm:*** Hardware uses APM (or APM emulation)
* ***bluetooth:*** Hardware has integrated BT
* ***efi:*** Support for booting through EFI
* ***ext2:*** Hardware HDD or Microdrive
* ***irda:*** Hardware has IrDA support
* ***keyboard:*** Hardware has a keyboard
* ***pcbios:*** Support for booting through BIOS
* ***pci:*** Hardware has a PCI bus
* ***pcmcia:*** Hardware has PCMCIA or CompactFlash sockets
* ***phone:*** Mobile phone (voice) support
* ***qvga:*** Machine has a QVGA (320x240) display
* ***rtc:*** Machine has a Real-Time Clock
* ***screen:*** Hardware has a screen
* ***serial:*** Hardware has serial support (usually RS232)
* ***touchscreen:*** Hardware has a touchscreen
* ***usbgadget:*** Hardware is USB gadget device capable
* ***usbhost:*** Hardware is USB Host capable
* ***vfat:*** FAT file system support
* ***wifi:*** Hardware has integrated WiFi

**31.2. Distro Features**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features-distro)

The items below are features you can use with [DISTRO\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-DISTRO_FEATURES) to enable features across your distribution. Features do not have a one-to-one correspondence to packages, and they can go beyond simply controlling the installation of a package or packages. In most cases, the presence or absence of a feature translates to the appropriate option supplied to the configure script during the[do\_configure](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-tasks-configure) task for the recipes that optionally support the feature.

Some distro features are also machine features. These select features make sense to be controlled both at the machine and distribution configuration level. See the [COMBINED\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-COMBINED_FEATURES) variable for more information.

This list only represents features as shipped with the Yocto Project metadata:

* ***alsa:*** Include ALSA support (OSS compatibility kernel modules installed if available).
* ***api-documentation:*** Enables generation of API documentation during recipe builds. The resulting documentation is added to SDK tarballs when the bitbake -c populate\_sdk command is used. See the "[Adding API Documentation to the Standard SDK](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#adding-api-documentation-to-the-standard-sdk)" section in the Yocto Project Software Development Kit (SDK) Developer's Guide for more information.
* ***bluetooth:*** Include bluetooth support (integrated BT only).
* ***bluez5:*** Include BlueZ Version 5, which provides core Bluetooth layers and protocols support.

**Note**

The default value for the [DISTRO FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-DISTRO_FEATURES) variable includes "bluetooth", which causes bluez5 to be backfilled in for bluetooth support. If you do not want bluez5 backfilled and would rather use bluez4, you need to use the[DISTRO\_FEATURES\_BACKFILL\_CONSIDERED](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-DISTRO_FEATURES_BACKFILL_CONSIDERED) variable as follows:

DISTRO\_FEATURES\_BACKFILL\_CONSIDERED = "bluez5"

Setting this variable tells the OpenEmbedded build system that you have considered but ruled out using the bluez5 feature and that bluez4 will be used.

* ***cramfs:*** Include CramFS support.
* ***directfb:*** Include DirectFB support.
* ***ext2:*** Include tools for supporting for devices with internal HDD/Microdrive for storing files (instead of Flash only devices).
* ***ipsec:*** Include IPSec support.
* ***ipv6:*** Include IPv6 support.
* ***irda:*** Include IrDA support.
* ***keyboard:*** Include keyboard support (e.g. keymaps will be loaded during boot).
* ***ldconfig:*** Include support for ldconfig and ld.so.conf on the target.
* ***nfs:*** Include NFS client support (for mounting NFS exports on device).
* ***opengl:*** Include the Open Graphics Library, which is a cross-language, multi-platform application programming interface used for rendering two and three-dimensional graphics.
* ***pci:*** Include PCI bus support.
* ***pcmcia:*** Include PCMCIA/CompactFlash support.
* ***ppp:*** Include PPP dialup support.
* ***ptest:*** Enables building the package tests where supported by individual recipes. For more information on package tests, see the "[Testing Packages With ptest](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#testing-packages-with-ptest)" section in the Yocto Project Development Manual.
* ***smbfs:*** Include SMB networks client support (for mounting Samba/Microsoft Windows shares on device).
* ***systemd:*** Include support for this init manager, which is a full replacement of for init with parallel starting of services, reduced shell overhead, and other features. This init manager is used by many distributions.
* ***usbgadget:*** Include USB Gadget Device support (for USB networking/serial/storage).
* ***usbhost:*** Include USB Host support (allows to connect external keyboard, mouse, storage, network etc).
* ***wayland:*** Include the Wayland display server protocol and the library that supports it.
* ***wifi:*** Include WiFi support (integrated only).
* ***x11:*** Include the X server and libraries.

**31.3. Image Features**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features-image)

The contents of images generated by the OpenEmbedded build system can be controlled by the [IMAGE\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-IMAGE_FEATURES) and[EXTRA\_IMAGE\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-EXTRA_IMAGE_FEATURES) variables that you typically configure in your image recipes. Through these variables, you can add several different predefined packages such as development utilities or packages with debug information needed to investigate application problems or profile applications.

The following image features are available for all images:

* ***allow-empty-password:*** Allows Dropbear and OpenSSH to accept root logins and logins from accounts having an empty password string.
* ***dbg-pkgs:*** Installs debug symbol packages for all packages installed in a given image.
* ***debug-tweaks:*** Makes an image suitable for development (e.g. allows root logins without passwords and enables post-installation logging). See the 'allow-empty-password', 'empty-root-password', and 'post-install-logging' features in this list for additional information.
* ***dev-pkgs:*** Installs development packages (headers and extra library links) for all packages installed in a given image.
* ***doc-pkgs:*** Installs documentation packages for all packages installed in a given image.
* ***empty-root-password:*** Sets the root password to an empty string, which allows logins with a blank password.
* ***package-management:*** Installs package management tools and preserves the package manager database.
* ***post-install-logging:*** Enables logging postinstall script runs to the /var/log/postinstall.log file on first boot of the image on the target system.

**Note**

To make the /var/log directory on the target persistent, use the [VOLATILE\_LOG\_DIR](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-VOLATILE_LOG_DIR) variable by setting it to "no".

* ***ptest-pkgs:*** Installs ptest packages for all ptest-enabled recipes.
* ***read-only-rootfs:*** Creates an image whose root filesystem is read-only. See the "[Creating a Read-Only Root Filesystem](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#creating-a-read-only-root-filesystem)" section in the Yocto Project Development Manual for more information.
* ***splash:*** Enables showing a splash screen during boot. By default, this screen is provided by psplash, which does allow customization. If you prefer to use an alternative splash screen package, you can do so by setting the SPLASH variable to a different package name (or names) within the image recipe or at the distro configuration level.
* ***staticdev-pkgs:*** Installs static development packages, which are static libraries (i.e. \*.a files), for all packages installed in a given image.

Some image features are available only when you inherit the [core-image](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-classes-core-image) class. The current list of these valid features is as follows:

* ***eclipse-debug:*** Provides Eclipse remote debugging support.
* ***hwcodecs:*** Installs hardware acceleration codecs.
* ***nfs-server:*** Installs an NFS server.
* ***perf:*** Installs profiling tools such as perf, systemtap, and LTTng. For general information on user-space tools, see the [Yocto Project Software Development Kit (SDK) Developer's Guide](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#sdk-manual).
* ***ssh-server-dropbear:*** Installs the Dropbear minimal SSH server.
* ***ssh-server-openssh:*** Installs the OpenSSH SSH server, which is more full-featured than Dropbear. Note that if both the OpenSSH SSH server and the Dropbear minimal SSH server are present in IMAGE\_FEATURES, then OpenSSH will take precedence and Dropbear will not be installed.
* ***tools-debug:*** Installs debugging tools such as strace and gdb. For information on GDB, see the "[Debugging With the GNU Project Debugger (GDB) Remotely](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#platdev-gdb-remotedebug)" section in the Yocto Project Development Manual. For information on tracing and profiling, see the Yocto Project Profiling and Tracing Manual.
* ***tools-sdk:*** Installs a full SDK that runs on the device.
* ***tools-testapps:*** Installs device testing tools (e.g. touchscreen debugging).
* ***x11:*** Installs the X server.
* ***x11-base:*** Installs the X server with a minimal environment.
* ***x11-sato:*** Installs the OpenedHand Sato environment.

**31.4. Feature Backfilling**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-features-backfill)

Sometimes it is necessary in the OpenEmbedded build system to extend [MACHINE\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-MACHINE_FEATURES) or [DISTRO\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-DISTRO_FEATURES) to control functionality that was previously enabled and not able to be disabled. For these cases, we need to add an additional feature item to appear in one of these variables, but we do not want to force developers who have existing values of the variables in their configuration to add the new feature in order to retain the same overall level of functionality. Thus, the OpenEmbedded build system has a mechanism to automatically "backfill" these added features into existing distro or machine configurations. You can see the list of features for which this is done by finding the [DISTRO\_FEATURES\_BACKFILL](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-DISTRO_FEATURES_BACKFILL) and [MACHINE\_FEATURES\_BACKFILL](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-MACHINE_FEATURES_BACKFILL) variables in the meta/conf/bitbake.conf file.

Because such features are backfilled by default into all configurations as described in the previous paragraph, developers who wish to disable the new features need to be able to selectively prevent the backfilling from occurring. They can do this by adding the undesired feature or features to the [DISTRO\_FEATURES\_BACKFILL\_CONSIDERED](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-DISTRO_FEATURES_BACKFILL_CONSIDERED) or [MACHINE\_FEATURES\_BACKFILL\_CONSIDERED](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-MACHINE_FEATURES_BACKFILL_CONSIDERED)variables for distro features and machine features respectively.

Here are two examples to help illustrate feature backfilling:

* ***The "pulseaudio" distro feature option***: Previously, PulseAudio support was enabled within the Qt and GStreamer frameworks. Because of this, the feature is backfilled and thus enabled for all distros through the DISTRO\_FEATURES\_BACKFILLvariable in the meta/conf/bitbake.conf file. However, your distro needs to disable the feature. You can disable the feature without affecting other existing distro configurations that need PulseAudio support by adding "pulseaudio" toDISTRO\_FEATURES\_BACKFILL\_CONSIDERED in your distro's .conf file. Adding the feature to this variable when it also exists in the DISTRO\_FEATURES\_BACKFILL variable prevents the build system from adding the feature to your configuration'sDISTRO\_FEATURES, effectively disabling the feature for that particular distro.
* ***The "rtc" machine feature option***: Previously, real time clock (RTC) support was enabled for all target devices. Because of this, the feature is backfilled and thus enabled for all machines through the MACHINE\_FEATURES\_BACKFILL variable in themeta/conf/bitbake.conf file. However, your target device does not have this capability. You can disable RTC support for your device without affecting other machines that need RTC support by adding the feature to your machine'sMACHINE\_FEATURES\_BACKFILL\_CONSIDERED list in the machine's .conf file. Adding the feature to this variable when it also exists in the MACHINE\_FEATURES\_BACKFILL variable prevents the build system from adding the feature to your configuration'sMACHINE\_FEATURES, effectively disabling RTC support for that particular machine.