**5.2. Customizing Images[¶](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html" \l "usingpoky-extend-customimage" \o "Permalink)**

You can customize images to satisfy particular requirements. This section describes several methods and provides guidelines for each.

**5.2.1. Customizing Images Using local.conf**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#usingpoky-extend-customimage-localconf)

Probably the easiest way to customize an image is to add a package by way of the local.conf configuration file. Because it is limited to local use, this method generally only allows you to add packages and is not as flexible as creating your own customized image. When you add packages using local variables this way, you need to realize that these variable changes are in effect for every build and consequently affect all images, which might not be what you require.

To add a package to your image using the local configuration file, use the [IMAGE\_INSTALL](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-IMAGE_INSTALL) variable with the \_append operator:

IMAGE\_INSTALL\_append = " strace"

Use of the syntax is important - specifically, the space between the quote and the package name, which is strace in this example. This space is required since the \_append operator does not add the space.

Furthermore, you must use \_append instead of the += operator if you want to avoid ordering issues. The reason for this is because doing so unconditionally appends to the variable and avoids ordering problems due to the variable being set in image recipes and.bbclass files with operators like ?=. Using \_append ensures the operation takes affect.

As shown in its simplest use, IMAGE\_INSTALL\_append affects all images. It is possible to extend the syntax so that the variable applies to a specific image only. Here is an example:

IMAGE\_INSTALL\_append\_pn-core-image-minimal = " strace"

This example adds strace to the core-image-minimal image only.

You can add packages using a similar approach through the [CORE\_IMAGE\_EXTRA\_INSTALL](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-CORE_IMAGE_EXTRA_INSTALL) variable. If you use this variable, onlycore-image-\* images are affected.

**5.2.2. Customizing Images Using Custom IMAGE\_FEATURES and EXTRA\_IMAGE\_FEATURES**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#usingpoky-extend-customimage-imagefeatures)

Another method for customizing your image is to enable or disable high-level image features by using 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. Although the functions for both variables are nearly equivalent, best practices dictate usingIMAGE\_FEATURES from within a recipe and using EXTRA\_IMAGE\_FEATURES from within your local.conf file, which is found in the[Build Directory](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#build-directory).

To understand how these features work, the best reference is meta/classes/core-image.bbclass. This class lists out the available [IMAGE\_FEATURES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-IMAGE_FEATURES) of which most map to package groups while some, such as debug-tweaks and read-only-rootfs, resolve as general configuration settings.

In summary, the file looks at the contents of the IMAGE\_FEATURES variable and then maps or configures the feature accordingly. Based on this information, the build system automatically adds the appropriate packages or configurations to the [IMAGE\_INSTALL](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-IMAGE_INSTALL)variable. Effectively, you are enabling extra features by extending the class or creating a custom class for use with specialized image .bb files.

Use the EXTRA\_IMAGE\_FEATURES variable from within your local configuration file. Using a separate area from which to enable features with this variable helps you avoid overwriting the features in the image recipe that are enabled with IMAGE\_FEATURES. The value of EXTRA\_IMAGE\_FEATURES is added to IMAGE\_FEATURES within meta/conf/bitbake.conf.

To illustrate how you can use these variables to modify your image, consider an example that selects the SSH server. The Yocto Project ships with two SSH servers you can use with your images: Dropbear and OpenSSH. Dropbear is a minimal SSH server appropriate for resource-constrained environments, while OpenSSH is a well-known standard SSH server implementation. By default, the core-image-sato image is configured to use Dropbear. The core-image-full-cmdline and core-image-lsb images both include OpenSSH. The core-image-minimal image does not contain an SSH server.

You can customize your image and change these defaults. Edit the IMAGE\_FEATURES variable in your recipe or use theEXTRA\_IMAGE\_FEATURES in your local.conf file so that it configures the image you are working with to include ssh-server-dropbear or ssh-server-openssh.

**Note**

See the "[Images](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#ref-images)" section in the Yocto Project Reference Manual for a complete list of image features that ship with the Yocto Project.

**5.2.3. Customizing Images Using Custom .bb Files**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#usingpoky-extend-customimage-custombb)

You can also customize an image by creating a custom recipe that defines additional software as part of the image. The following example shows the form for the two lines you need:

IMAGE\_INSTALL = "packagegroup-core-x11-base package1 package2"

inherit core-image

Defining the software using a custom recipe gives you total control over the contents of the image. It is important to use the correct names of packages in the [IMAGE\_INSTALL](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-IMAGE_INSTALL) variable. You must use the OpenEmbedded notation and not the Debian notation for the names (e.g. glibc-dev instead of libc6-dev).

The other method for creating a custom image is to base it on an existing image. For example, if you want to create an image based on core-image-sato but add the additional package strace to the image, copy the meta/recipes-sato/images/core-image-sato.bb to a new .bb and add the following line to the end of the copy:

IMAGE\_INSTALL += "strace"

**5.2.4. Customizing Images Using Custom Package Groups**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#usingpoky-extend-customimage-customtasks)

For complex custom images, the best approach for customizing an image is to create a custom package group recipe that is used to build the image or images. A good example of a package group recipe is meta/recipes-core/packagegroups/packagegroup-base.bb.

If you examine that recipe, you see that the [PACKAGES](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-PACKAGES) variable lists the package group packages to produce. The inherit packagegroup statement sets appropriate default values and automatically adds -dev, -dbg, and -ptest complementary packages for each package specified in the PACKAGES statement.

**Note**

The inherit packages should be located near the top of the recipe, certainly before the PACKAGES statement.

For each package you specify in PACKAGES, you can use [RDEPENDS](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-RDEPENDS) and [RRECOMMENDS](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-RRECOMMENDS) entries to provide a list of packages the parent task package should contain. You can see examples of these further down in the packagegroup-base.bb recipe.

Here is a short, fabricated example showing the same basic pieces:

DESCRIPTION = "My Custom Package Groups"

inherit packagegroup

PACKAGES = "\

packagegroup-custom-apps \

packagegroup-custom-tools \

"

RDEPENDS\_packagegroup-custom-apps = "\

dropbear \

portmap \

psplash"

RDEPENDS\_packagegroup-custom-tools = "\

oprofile \

oprofileui-server \

lttng-tools"

RRECOMMENDS\_packagegroup-custom-tools = "\

kernel-module-oprofile"

In the previous example, two package group packages are created with their dependencies and their recommended package dependencies listed: packagegroup-custom-apps, and packagegroup-custom-tools. To build an image using these package group packages, you need to add packagegroup-custom-apps and/or packagegroup-custom-tools to [IMAGE\_INSTALL](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-IMAGE_INSTALL). For other forms of image dependencies see the other areas of this section.

**5.2.5. Customizing an Image Hostname**[**¶**](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#usingpoky-extend-customimage-image-name)

By default, the configured hostname (i.e. /etc/hostname) in an image is the same as the machine name. For example, if [MACHINE](http://www.yoctoproject.org/docs/2.3.1/mega-manual/mega-manual.html#var-MACHINE)equals "qemux86", the configured hostname written to /etc/hostname is "qemux86".

You can customize this name by altering the value of the "hostname" variable in the base-files recipe using either an append file or a configuration file. Use the following in an append file:

hostname="myhostname"

Use the following in a configuration file:

hostname\_pn-base-files = "myhostname"

Changing the default value of the variable "hostname" can be useful in certain situations. For example, suppose you need to do extensive testing on an image and you would like to easily identify the image under test from existing images with typical default hostnames. In this situation, you could change the default hostname to "testme", which results in all the images using the name "testme". Once testing is complete and you do not need to rebuild the image for test any longer, you can easily reset the default hostname.

Another point of interest is that if you unset the variable, the image will have no default hostname in the filesystem. Here is an example that unsets the variable in a configuration file:

hostname\_pn-base-files = ""

Having no default hostname in the filesystem is suitable for environments that use dynamic hostnames such as virtual machines.