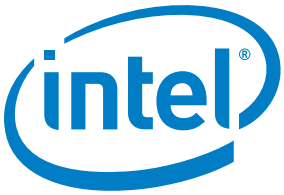

Using Package Manager to Efficiently Develop Yocto Project-based Systems

Package Manager White Paper

May 2015

Document Number: 332443 Rev.1 .0

Contents

History

Date	Revision	Description
May 2015	1.0	Initial release.

§

1.0 Introduction to Runtime Package Management

During development of an embedded Linux* system, you can update, add, remove, or query the packages on a target device at runtime. This is called runtime package management.

This white paper provides guidance to a system developer regarding runtime package management. It discusses the features of package managers available in the Yocto project and best practices in the development flow that could be taken in making Yocto project-based embedded Linux* development more efficient.

1.1 Terminology

Table 1. Terminology

Term	Description
RPM	Red Hat Package Manager software package format
DEB	Debian software package format
IPK	OpenEmbedded software package format
OPKG	Open PacKaGe Management

1.2 Reference Documents

Table 2. Reference Documents

Document	Document No./Location
Yocto Project Reference Manual	http://www.yoctoproject.org/docs/latest/ref-manual/ref-manual.htmll
Smart Package Manager	http://labix.org/smart/
Information about using createrepo to create standalone repository	https://plasmixs.wordpress.com/2014/04/05/package-management-with-smart-package-management/
Information about using opkg-make-index to create standalone repository	http://www.jumpnowtek.com/yocto/Managing-a-private-opkg-repository.html

§

2.0 Runtime Package Management

In order to use the runtime package management, you need to:

- Set up a package repository consisting of pre-compiled packages and metadata (see section 2.1, Package Repository).
- Make sure the package manipulation tools are on the target device (see section 2.2, Package Manipulation Tools on the Device).

The remainder of this section describes the process.

2.1 Package Repository

The package repository is storage with pre-compiled packages and metadata describing package contents. The repository can be generated during the bitbake image build or manually using package manipulation tools.

2.1.1 Package Feed Repository

During the process of building an image, the OpenEmbedded build system called BitBake executes ingredients in recipes to create packages and places them into the package feed repository. After all required packages for an image are ready, BitBake gets packages from the package feed repository to generate an image.

A variable PACKAGE_CLASSES is required for BitBake to know what package format is used for generating packages. It can be set in the configuration file conf/**local.conf** as one or more of following arguments:

- `package_ipk`
- `package_rpm`
- `package_deb`

The package feed repository can be found in `build/tmp/deply/`**`PACKAGE_CLASS`**, which is `rpm`, `deb`, or `ipk;` they can also be used as a package repository.

Please note that whenever you perform any sort of build step that can potentially generate a package or modify an existing package, it is recommended to re-generate the package index with the bitbake `package-index` command or a build image.

2.1.2 Standalone Repository

In addition to the package feed repository, you can also create an `rpm` or `ipk` package repository by using `createrepo` and `opkg-make-index` tools. Please refer to section 1.2, Reference Documents, for links to more information about using `createrepo` and `opkg-make-index`.

2.2 Package Manipulation Tools on the Device

A package manipulation tool is required to do runtime package management on the target:

- `opkg` is the tool to handle ipk package.
- `smartpm` is tool to handle rpm package.

Adding `opkg` or `smartpm` to the variable `IMAGE_INSTALL_append` can generate an image with those tools. After the image is up and running, you need to initialize the `package database` on the target.

In addition to package manipulation tools, you can include the package database to generate an image by adding `package-management` in the `IMAGE_FEATURES` of image recipe or `EXTRA_IMAGE_FEATURES` variable of `conf/`**`local.conf`**.

The `core-image-sato` recipe comes with `IMAGE_FEATURES += package-management` and so you do not need to do it in `conf/`**`local.conf`**. In the other hand, the `core-image-minimal` recipe does not include the package-management feature, so you need to enable it by adding `package-management` into the variable `EXTRA_IMAGE_FEATURES` in the `conf/`**`local.conf`** configuration file.

§

3.0 Using Runtime Package Management

To demonstrate runtime package management:

- This section uses `core-image-minimal` image recipe as an example with the following content in `conf/local.conf` :

  - `MACHINE ?= qemux86-64`
  - `PACKAGE_CLASSES = package_rpm package_ipk`
  - `IMAGE_INSTALL_append = opkg smartpm`
  - `MAGE_ROOTFS_EXTRA_SPACE = 131072`

- Build an image using the following command:

  ```
  source oe-init-build-env; bitbake core-image-minimal
  ```

- Run the generated image in QEMU with the following command:

  ```
  runqemu qemux86-64
  ```

- This package uses the Python SimpleHTTPServer module as HTTP server with IP address 192.168.7.1.

  Note: Please use your build machine's IP address instead.

- After the initial image build, run `bitbake dropbear` to generate a dropbear package. Then run `bitbake package-index` or `bitbake core-image-minimal` to update the package index of package feeds. Otherwise, package manager cannot find the dropbear package.

- In the examples in the following sections:
  - `yocto@VirtualBox`: means you are at the host machine
  - `root@qemux86-64`: means you are at the target machine

3.1 Smart Package Manager

This section uses smart package manager to do runtime package management. It starts from the repository and target setup. Then use smart package manager to install, update packages, and search which package contains the command you need.

3.1.1 Set up the Remote Repository

A web server can be set up to host the package feed as a remote repository. A package feed of rpm packages can be found in the path listed below after the bitbake image build:

```
$(TMPDIR)/deploy/rpm
```

Below is an example of a generated package feed:

```
yocto@VirtualBox:~/yocto/poky/build/deploy/rpm$ ls -l
total 328
drwxr-xr-x 3 yocto yocto   4096 Apr 26 11:43 all
drwxr-xr-x 3 yocto yocto 258048 Apr 26 12:38 core2-64
drwxr-xr-x 3 yocto yocto  69632 Apr 26 11:43 qemux86_64
```

To set up the web server on a build machine, use the Python `SimpleHTTPServer` module:

```
yocto@VirtualBox:~/yocto/poky/build/deploy/rpm $ python -m
SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

3.1.2 Configure the Target System

To get help information, run the `smart` command without any arguments:

```
root@qemux86-64:~# smart
Usage: smart command [options] [arguments]

Action commands:
    update
    ...
```

To get current package information on the target, run `smart stats`:

```
root@qemux86-64:~# smart stats
Updating cache...           ################################ [100%]

Installed Packages: 0
Total Packages: 0
Total Provides: 0
Total Requires: 0
Total Upgrades: 0
Total Conflicts: 0
```

Channel is used by smart to manage the repository. Run `smart channel --add...` commands to add repository.

```
root@qemux86-64:~# smart channel --add all type=rpm-md
baseurl=http://192.168.7.1:8000/all

Alias: all
Type: rpm-md
Base URL: http://192.168.7.1:8000/all
Include this channel? (y/N): y

root@qemux86-64:~# smart channel --add core2_64 type=rpm-md
baseurl http://192.168.7.1:8000/core2_64

Alias: core2_64
Type: rpm-md
Base URL: /media/realroot/rpm/corei7_64
Include this channel? (y/N): y

root@qemux86-64:~# smart channel --add qemux86_64 type=rpm-md
baseurl=http://192.168.7.1:8000/qemux86_64

Alias: qemux86_64
Type: rpm-md
Base URL: http://192.168.7.1:8000/qemux86_64

Include this channel? (y/N): y
```

To retrieve repository metadata, run “smart update”:

```
root@qemux86-64:~# smart update
Updating cache...          ############################## [100%]

Fetching Information for ‘all’...
-> http://192.168.7.1:800/all/repodata/repmod.xml
repmod.xml                 ############################## [ 16%]
-> http://192.168.7.1:800/all/repodata/filelists.xml.gz
-> http://192.168.7.1:800/all/repodata/primary.xml.gz
```

```
filelists.xml.gz            ################################ [ 25%]
primary.xml.gz              ################################ [ 33%]

Fetching Information for ‘qemux86_64’...
-> http://192.168.7.1:800/qemux86_64/repodata/repmod.xml
repmod.xml                  ################################ [ 50%]
-> http://192.168.7.1:800/qemux86_64/repodata/filelists.xml.gz
-> http://192.168.7.1:800/qemux86_64/repodata/primary.xml.gz
filelists.xml.gz            ################################ [ 58%]
primary.xml.gz              ################################ [ 66%]

Fetching Information for ‘core2_64’...
-> http://192.168.7.1:800/core2_64/repodata/repmod.xml
repmod.xml                  ################################ [ 83%]
-> http://192.168.7.1:800/core2_64/repodata/filelists.xml.gz
-> http://192.168.7.1:800/core26_64/repodata/primary.xml.gz
filelists.xml.gz            ################################ [ 91%]
primary.xml.gz              ################################ [100%]

Updating cache...           ################################ [100%]

Channels have 3777 new packages.
Saving cache...
```

Below is the output of `SimpleHTTPServer` during the smart update:

```
192.168.7.2 - - [26/Apr/2015 15:20:56] "GET
/all/repodata/repmod.xml HTTP/1.0" 200 -
192.168.7.2 - - [26/Apr/2015 15:21:01] "GET
/all/repodata/primary.xml.gz HTTP/1.0" 200 -
192.168.7.2 - - [26/Apr/2015 15:21:07] "GET
/all/repodata/filelists.xml.gz HTTP/1.0" 200 –
192.168.7.2 - - [26/Apr/2015 15:21:12] "GET
/qemux86_64/repodata/repmod.xml HTTP/1.0" 200 -
192.168.7.2 - - [26/Apr/2015 15:21:17] "GET
/qemux86_64/repodata/primary.xml.gz HTTP/1.0" 200 -
192.168.7.2 - - [26/Apr/2015 15:21:22] "GET
/qemux86_64/repodata/filelists.xml.gz HTTP/1.0" 200 -
```

```
192.168.7.2 - - [26/Apr/2015 15:21:28] "GET
/core2_64/repodata/repmod.xml HTTP/1.0" 200 -
192.168.7.2 - - [26/Apr/2015 15:21:33] "GET
/core2_64/repodata/primary.xml.gz HTTP/1.0" 200 -
192.168.7.2 - - [26/Apr/2015 15:21:38] "GET
/core2_64/repodata/filelists.xml.gz HTTP/1.0" 200 -
```

To update the statistics of the package repository; run `smart stats`:

```
root@qemux86-64:~# smart stats
Loading cache...
Updating cache...           ################################ [100%]

Installed Packages: 0
Total Packages: 3857
Total Provides: 6654
Total Requires: 1814
Total Upgrades: 3865
Total Conflicts: 26
```

3.1.3 Install the Package

To install the package, run `smart install package-name`:

```
root@qemux86-64:~# smart install dropbear
Loading cache...
Updating cache...           ################################ [100%]

Computing transaction...

Installing packages (1):
  dropbear-2014.63-r0@core2_64

139.0kB of package files are needed. 273.1kB will be used.

Confirm changes? (Y/n):y

Fetching packages...
```

```
-> http://192.168.7.1:800/core2_64/dropbear-2014.63-
r0.core2_64.rpm

Committing transaction...
Preparing...                ################################# [  0%]
Output from dropbear-2014.63-r0@core2_64
 Removing any system startup links for dropbear ...
   1:Installing dropbear    ################################# [100%]
update-alternatives: Linking //usr/bin/rcp to
/usr/sbin/dropbearmulti
update-alternatives: Linking //usr/bin/ssh to
/usr/sbin/dropbearmulti
 Adding system startup for /etc/init.d/dropbear.
Starting Dropbear SSH server: Generating key, this may take a
while…
Public key portion is:
ssh-rsa AAAAB3N...
Fingerprint: md5 7f:19:5a:...
dropbear.
```

The output of `SimpleHTTPServer` when running `smart install dropbear` is:

```
192.168.7.2 - - [26/Apr/2015 15:33:56] "GET /core2_64/dropbear-
2014.63-r0.core2_64 HTTP/1.0" 200 -
```

3.1.4 Update the Package

Modify the dropbeak recipe and then run `bitbake dropbear` and `bitbake package-index` to create a new dropbear package. For details, please see section 3.3, Creating a Layer for Package Upgrade. Then run `smart update` for a list of updated packages.

```
root@qemux86-64:~# smart update
...
Channels have 3 new packages:
    dropbear-2014.63-r1@core2_64
    dropbear-dbg-2014.63-r1@core2_64
    dropbear-dev-2014.63-r1@core2_64
```

```
Saving cache...

root@qemux86-64:~# smart install dropbear
...
```

3.1.5 Search for a Package Containing a Specific Command

In some cases, you want to know which package has the command you need. Smart package manager provides search command for this purpose. Run `smart search command-name` to list packages containing `command-name`. For example, `smart search ssh` will list package containing `ssh` command.

3.2 opkg Package Manager

This section uses `opkg` package manager to do runtime package management. It starts from repository and target setup. Then use of `opkg` package manager to install, update packages, and search which package containing command you need.

3.2.1 Set up the Remote Repository

Run `opkg` without any arguments for help information.

```
root@qemux86-64:~# opkg
opkg must have one sub-command argument
Usage: opkg [options] sub-command [arguments...]
Where sub-command is one of:

Package Manipulation:
    update                      Update list of available packages
    ...
```

Package feed of `ipk` packages can be found in the path listed below after the bitbake image build.

```
$(TMPDIR)/deploy/ipk
```

For example:

```
yocto@VirtualBox:~/yocto/poky/build/deploy/ipk$ ls -l
total 324
drwxr-xr-x 2 yocto yocto   4096 Apr 26 19:03 all
```

```
drwxr-xr-x 2 yocto yocto 241664 Apr 26 19:03 core2-64
-rw-r--r-- 1 yocto yocto      0 Apr 26 19:03 Packages
drwxr-xr-x 2 yocto yocto  69632 Apr 26 19:03 qemux86_64
```

Use Python `SimpleHTTPServer` module to setup web server on build machine:

```
yocto@VirtualBox:~/yocto/poky/build/deploy/ipk$ python -m
SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

3.2.2 Configure the Target System

`opkg` uses configuration file `/etc/opkg/`**`base-feeds.conf`** to know where to look for package repository and `/var/lib/`**`opkg`** folder to store package metadata on the target device. Below is an example for adding repository information into base-feeds.conf. Create `/var/lib/`**`opkg`** folder if it does not exist.

```
src/gz all http://192.168.7.1:8000/all
src/gz core2-64 http://192.168.7.1:8000/core2-64
src/gz qemux86_64 http://192.168.10.8:8000/qemux86_64
```

Run `opkg update` to retrieve package data:

```
root@qemux86-64:~# opkg update
Downloading http://192.168.7.1:8000/all/Packages.gz.
Inflating http://192.168.7.1:8000/all/Packages.gz.
Updated list of available packages in /var/lib/opkg/all.
Downloading http://192.168.7.1:8000/core2-64/Packages.gz.
Inflating http://192.168.7.1:8000/core2-64/Packages.gz.
Updated list of available packages in /var/lib/opkg/core2-64.
Downloading http://192.168.7.1:8000/qemux86-64/Packages.gz.
Inflating http://192.168.7.1:8000/qemux86-64/Packages.gz.
Updated list of available packages in /var/lib/opkg/qemux86-64.
```

Output of SimpleHTTPServer when run “opkg update”.

```
192.168.7.2 - - [26/Apr/2015 10:58:36] "GET /all/Packages.gz
HTTP/1.1" 200 -
192.168.7.2 - - [26/Apr/2015 10:58:41] "GET /core2-64/Packages.gz
HTTP/1.1" 200 -
```

```
192.168.7.2 - - [26/Apr/2015 10:58:52] "GET
/qemux86_64/Packages.gz HTTP/1.1" 200 -
```

3.2.3 Install the Package

Install a package with command `opkg install package-name`.

```
root@qemux86-64:~# opkg install dropbear
Installing dropbear (2014.63-r0) on root.
Downloading http://192.168.7.2:8000/core2-64/dropbear_2014.63-
r0_core2-64.ipk.
 Removing any system startup links for dropbear …
Configuring dropbear.
update-alternatives: Linking //usr/bin/rcp to
/usr/sbin/dropbearmulti
update-alternatives: Linking //usr/bin/ssh to
/usr/sbin/dropbearmulti
 Adding system startup for /etc/init.d/dropbear.
Starting Dropbear SSH server: Generating key, this may take a
while…
Public key portion is:
ssh-rsa AAAAB3N...
Fingerprint: md5 7f:19:5a:...
dropbear.
root@qemux86-64:~#
```

Output of `SimpleHTTPServer`:

```
192.168.7.2 - - [26/Apr/2015 11:36:59] "GET /core2-
64/dropbear_2014.63-r0_core2-64.ipk HTTP/1.1" 200 -
```

3.2.4 Update the Package

Modify dropbeak recipe and then run “bitbake dropbear” and “bitbake package-index” to create new dropbear package. For detail, please refer to section 3.3, Creating layer for Package Upgrade. Run following commands to check update.

```
root@qemux86-64:~# opkg update
root@qemux86-64:~# opkg list-upgradable
```

```
dropbear – 2014.63-r0 2014.63-r1
```

Run `opkg upgrade` to install the updated package.

```
root@qemux86-64:~# opkg upgrade
Upgrading dropbear from 2014.63-r0 to 2014.63-r1 on root.
...
```

3.2.5 Search for a Package Containing a Specific Command

In some cases, you want to know which package has the command you need. Opkg package manager provides search command for this purpose. Run `opkg search command-name` to list packages containing command-name. For example, `opkg search ssh` will list packages containing **ssh** command.

3.3 Creating a Layer for Package Upgrade

To use package upgrade function, this section uses yocto-layer tool to create a layer to host bbapend recipe.

```
yocto@VirtualBox:~/yocto/poky/$ yocto-layer create pkgmgr
Please enter the layer priority you'd like to use for the layer:
[default: 6]
Would you like to have an example recipe created? (y/n) [default:
n]
Would you like to have an example bbapend file created? (y/n)
[default: n]

New layer created in meta-pkgmgr.

Don't forget to add it to your BBLAYERS (for detail see meta-
pkgmgr\README).
```

Create folder meta-pkgmgr/recipes-core/dropbear with command below.

```
yocto@VirtualBox:~/yocto/poky/$ mkdir –p meta-pkgmgr/recipes-
core/dropbear/
```

Use following content to create dropbear_2014.63.bbappend in meta-pkgmgr/recipes-core/dropbear/.

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

PR := “r1”
```

To build recipes in `meta-pkgmgr`, it is required to add its path into `conf\`**`bbalyers.conf`**:

```
yocto@VirtualBox:~/yocto/poky/$ echo “BBLAYERS +=
/home/yocto/yocto/poky/meta-pkgmgr” >> build/conf/bblayers.conf
```

Run `bitbkae dropbear` and `bitbake package-index` to generate a new version of the dropbear package and update the package feed repository.

§