AN APPROACH TO DELIVER HARDWARE-DEPENDENT PACKAGES IN ORDER TO REDUCE EFFORT OF UPDATING AGL DISTRIBUTION IMAGES

AUTOMOTIVE LINUX SUMMIT JAPAN 2018

06/21/2018, TOKYO
KHIEM NGUYEN
SENIOR STAFF ENGINEER
RENESAS DESIGN VIETNAM
RENESAS ELECTRONICS CORPORATION
WHO AM I?

- Name: Khiem Trong. Nguyen (KHIEM Nguyen - キム グエン)
- Company: Renesas Design Vietnam

- Career: 10 years experiences in embedded software development
  - Development and verification for Mobile and In-vehicle software platform
  - Development for test automation solutions of In-vehicle software platform

- Email: khiem.nguyen.xt@renesas.com
ABOUT RENESAS AND RENESAS DESIGN VIETNAM

- Renesas Design Vietnam Co., Ltd. (RVC) was founded in October 2004, as one of the main design centers in Renesas group.
- Business line: Design of semiconductor for both hardware and software.
MOTIVATION
Traditional AGL development workflow and issues (1/2)

AGL Developers

Recipe files

AGL Images

Build and Deploy for each platforms

AGL-JTA (jta)

LAVA

Deploy

Build

Deploy

Deploy

Request

CI builds

Gerrit

Jenkins (Build)

ROOTFS

KERNEL

BOOTLOADER

APPS

Most update

Renesas R-Car Series
MOTIVATION
TRADITIONAL AGL DEVELOPMENT WORKFLOW AND ISSUES (2/2)

Take **9 hrs. 41 min total** from scheduled to completion for all machine configurations.

- Simulation on each platform (e.g. Starter Kit Pro)
  - Full build & deploy time: ~5 hours.
  - AGL developers: 81 (**)
    - Cost = 81*5 = **405 hours.**

Build system and distribution organization (*)


(**) 81 unique individuals and 33 unique companies have contributed to AGL since the start of the project [2]
MOTIVATION
AGL PROFILES – BUILD ONCE AND DEPLOY FOR EACH PLATFORM (1/2)

- Enable easy/flexible derivations of AGL for various purposes.
- Define a single location to control core functionalities.
- Deploy only the updated packages to optimize development cycle times early in the lifecycle.
- CI performance: avoid repeated full builds by composing binary packages.

AGL Profiles


Package Manipulation tools: DNF, Zypper, Smart, Apt.
## MOTIVATION

### SOFTWARE UPDATE SOLUTIONS

- Focus on value-added feature development by reducing integration effort.
- The comparison between Software update solutions [1] [2] [3]

<table>
<thead>
<tr>
<th>Key criteria</th>
<th>Run-time Installation (RPM)</th>
<th>Dual A/B rootfs (SWUpdate)</th>
<th>Incremental atomic updates (OSTree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robust and secure</td>
<td>Red</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Adapt existing environment</td>
<td>Green</td>
<td>Yellow</td>
<td>Red</td>
</tr>
<tr>
<td>Easy to get started</td>
<td>Green</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>Bandwidth consumption</td>
<td>Green</td>
<td>Red</td>
<td>Green</td>
</tr>
<tr>
<td>Downtime during update</td>
<td>Green</td>
<td>Green</td>
<td>Red</td>
</tr>
</tbody>
</table>

- Focus on value-added feature development by reducing integration effort.
- The comparison between Software update solutions [1] [2] [3]

### Table:

<table>
<thead>
<tr>
<th>Key criteria</th>
<th>Run-time Installation (RPM)</th>
<th>Dual A/B rootfs (SWUpdate)</th>
<th>Incremental atomic updates (OSTree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robust and secure</td>
<td>Red</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Adapt existing environment</td>
<td>Green</td>
<td>Yellow</td>
<td>Red</td>
</tr>
<tr>
<td>Easy to get started</td>
<td>Green</td>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>Bandwidth consumption</td>
<td>Green</td>
<td>Red</td>
<td>Green</td>
</tr>
<tr>
<td>Downtime during update</td>
<td>Green</td>
<td>Green</td>
<td>Red</td>
</tr>
</tbody>
</table>
Motivation
Software Update Solution - Why RPM Package?

- Used by OpenEmbedded.
- Many Package Manipulation tools: Smartpm, Dandified Yum (DNF), Zypper, etc.
- Be applicable for both updating on the device and SDK.
- Check Package License Information.
- Sign/re-sign packages created by using rpm-tool (*) or Yocto build system.
- Combine with other tools (e.g. SPDX tools).

- Simulation on each target
  - Full build & deploy time: ~5 hours.
  - AGL developers: 81
  - Current workflow
    - Cost = 81 * 5 = 405 hours.

- Use Binary Packaging Distribution
  - Cost = 5 + 1 * 81 = 86 hours.

Reduce ~ 78%

TECHNICAL OVERVIEW [1/3]
RUNTIME PACKAGE MANAGEMENT

- Create an update server for over-the-air (OTA) updates.
- Allows update the system at runtime using package level, for many use cases.
  - In-field security updates.
  - System, packages or configuration customization at runtime.
- Remote debugging.

Development Host
Maintains packages and updates
Remote Repository
RPM Packages Repo

via HTTP/HTTPS

On PC Host / On the device
Requests Package updates/upgrades/installs/removes on demand
From Yocto Project 2.3, Smart package manager is replaced by DNF package manager.

DNF is the next generation version of the Yellodog Updater Modified (YUM) and licensed under General Public License v2.

DNF advanced functionalities (*)

- It's actively maintained and supports python 3.x.
- An undocumented API—this meant more work for developers.
- Is used to manage package installation and removal, as well as dependency resolution.

<table>
<thead>
<tr>
<th>DNF version</th>
<th>Poky version</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0.0</td>
<td>Pyro (2.3)</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Rocko (2.4)</td>
</tr>
<tr>
<td>2.7.5</td>
<td>Sumo (2.5)</td>
</tr>
</tbody>
</table>

Recommend the most recent version

TECHNICAL OVERVIEW [3/3]

META-YOMO

- Is maintained by Ronan Le Martret (IOT.BZH’s member)
- Meta-yomo includes
  - SDK bootstrap: a SDK with DNF from Yocto to manage RPM repo and install dependencies for profile.
  - Scripts: allow to install a minimal set of packages on PC Host/on the target.

DEM0

- Install package-agl-demo-platform on AGL Eel 5.0.3.
- Demo for other use-cases (e.g. Updating SDK on PC Host)
  - Sanity testing after an update
    - Checksum files after installation.
    - Automated tests on the board.
    - Check that you can install future updates.
DEMO – SCREEN CAST

INSTALL PACKAGE-AGL-DEMO-PLATFORM ON AGL EEL 5.0.3/RENESAS R-CAR STARTER KIT PRO THROUGH RPMS

Agl-image-ivi ( .wic.xz ) + Packages repo Profile Demo = Agl-demo-platform
Installing packagegroup-agl-demo-platform [1/3]

Automatically asked authorization to install/download with all dependencies.

- Use 2 configuration files to define the package repositories available to the device:
  - Repositories: `/etc/yum.repos.d/*repo` file(s).
  - Main Configuration: `/etc/dnf/dnf.conf` file.
The packagegroup-agn-platform install has been successfully completed!
To show all the files installed by the package (e.g. homescreen)

```
rpm -ql homescreen-2017-1.0+git0+ec28d0bf46-r0.aarch64
```

```
/etc
/etc/agl-postinsts
/etc/agl-postinsts/10-homescreen-2017.sh
/usr
/usr/AGL
/usr/AGL/apps
/usr/AGL/apps/autoinstall
/usr/AGL/apps/autoinstall/homescreen-2017.wgt
```

To install AGL applications, need to execute some scripts in `/etc/agl-postins/* .sh`. 
Installing AGL applications by running run-agl-postinsts [3/3]

Reboot board after completing!
It’s the same

agl-demo-platform-m3ulcb.wic.xz

Renesas R-Car Starter Kit Pro
DEMO – SCREEN CAST
UPDATING SDK ON PC HOST

SDK bootstrap

SDK conf file

Install/Distribution

Native SDK sysroot

Target sysroot

environmen script

REPO

REPO
Initialize **SDK rootfs** by SDK bootstrap [1/3]

- SDK bootstrap use “m3ulcb-sdk-conf.sdk” configuration file.

- That *config* file includes:
  - Arch, Target repositories.
  - SDK native repositories.
  - Default packages/packages groups/profiles.

```bash
$ /ssd/demo/agl-sdk-bootstrap.sh
+ XDT_SDK_BOOTSTRAP=/ssd/demo/sdk-bootstrap
+ XDT_SDK=/ssd/demo/sdk-rootfs

# /ssd/demo/x86_64-sdk-bootstrap-5.0.3.sh -y -d /ssd/demo/sdk-bootstrap
yocto sdk bootstrap installer version 5.0.3
===========================================================================
You are about to install the SDK to "/ssd/demo/sdk-bootstrap". Proceed[Y/n]? Y
Extracting SDK ...... done
Setting it up... done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.
$ /ssd/demo/sdk-bootstrap/environment-setup-x86_64-aglsdk-linux

# source /ssd/demo/sdk-bootstrap/environment-setup-x86_64-aglsdk-linux

#### m3ulcb-sdk-conf.sdk ####

<table>
<thead>
<tr>
<th>Repo:</th>
<th>m3ulcb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>m3ulcb</td>
</tr>
<tr>
<td>Baseurl:</td>
<td><a href="http://192.168.10.53:5678">http://192.168.10.53:5678</a></td>
</tr>
<tr>
<td>TargetRepo:</td>
<td>aarch64-agl-linux,</td>
</tr>
<tr>
<td>HostRepo:</td>
<td>x86_64-linux</td>
</tr>
<tr>
<td>SDK_init_conf</td>
<td></td>
</tr>
<tr>
<td>TUNE_PKGARCH:</td>
<td>&quot;aarch64&quot;</td>
</tr>
<tr>
<td>TARGET_VENDOR:</td>
<td>&quot;-agl&quot;</td>
</tr>
<tr>
<td>SDK_ARCH:</td>
<td>&quot;x86_64&quot;</td>
</tr>
<tr>
<td>SDK_VENDOR:</td>
<td>&quot;aglsdk&quot;</td>
</tr>
<tr>
<td>SDK_VERSION:</td>
<td>&quot;5.0.3&quot;</td>
</tr>
<tr>
<td>SDK_OS:</td>
<td>&quot;linux&quot;</td>
</tr>
<tr>
<td>DISTRO:</td>
<td>&quot;poky-agl&quot;</td>
</tr>
<tr>
<td>MACHINE:</td>
<td>&quot;m3ulcb&quot;</td>
</tr>
</tbody>
</table>

* init-sdk-rootfs -l m3ulcb-sdk-conf.sdk -o /ssd/demo/sdk-rootfs
Done!
```
Use script `dnf4Native` to install `nativesdk` packages

Packages for native SDK

795 packages = agl-demo-platform-crosssdk native profile

```
$ cat package-native
nativesdk-gn	nativesdk-x2	nativesdk-lua	nativesdk-zlib
nativesdk-ninja	nativesdk-gperf	nativesdk-wayland
nativesdk-nss-dev	nativesdk-nsp-dev
nativesdk-wayland-dev	nativesdk-perl-modules
nativesdk-packagegroup-sdk-host	nativesdk-packagegroup-qt5-toolchain-host
packagegroup-cross-canadian
$ pkgs="\"cat package-native\""
$ {{XDT_SDK}}/m3ulcb-sdk/dnf4Native install $pkgs
Last metadata expiration check: 08:00:00 ago on Fri May 18 06:18:41 2018 UTC.
Dependencies resolved.

<table>
<thead>
<tr>
<th>Package</th>
<th>Arch</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>nativesdk-gn</td>
<td>x86_64</td>
<td>nativesdk 20179928.r4</td>
</tr>
<tr>
<td>nativesdk-gperf</td>
<td>x86_64</td>
<td>nativesdk 3.0.4-r0</td>
</tr>
<tr>
<td>nativesdk-libz</td>
<td>x86_64</td>
<td>nativesdk 1.2.11-r0</td>
</tr>
</tbody>
</table>
```

Complete!
Use script `dnf4Target` to install target packages.

Packages for target:

```
$ cat package-target
mc dnf rpm screen psplash run-postinsts omx-user-module
agl-desktop-config
packagegroup-agl-devel
packagegroup-core-ssh-openssh
packagegroup-core-tools-debug
packagegroup-core-eclipse-debug
packagegroup-core-tools-profile
kernel-modules
kernel-device-tree
kernel-module-vsp2
kernel-module-vspm
kernel-module-vspm-if
kernel-module-vspsvm
packagegroup-agl-image-ivi
```

$ pkgs="\`cat package-target\``

$ \
$(XT1_SDK)/m3ulcb-sdk/dnf4Target install $pkgs

```
arach64-agl-linux
```

Last metadata expiration check: 0:00:00 ago on Fri May 18 04:15:04 2018 UTC.

```
890 packages = agl-image-ivi profile
```

© 2018 Renesas Electronics Corporation. All rights reserved.
CONCLUSION
CONCLUSION

- Using RPM binary packages is a good approach for AGL development and collaboration.
  - Support applications/root filesystems update on development devices.
  - Easily update development environment (SDK) on the host machine.
  - Integrate with update server for over-the-air (OTA) updates.
- The approach is aligned with existing features on AGL software.
  - Easily move from one AGL profile to other AGL profiles
  - Generate the binary packages once and share to others
REFERENCES


IMPLEMENTATION [1/6]
BUILD CONFIGURATION

- To demonstrate runtime package management
  - Enabling Additional PR Server Features
    ```
    PRSERV_HOST = "localhost:0"
    INHERIT += buildhistory
    BUILDHISTORY_COMMIT = "1"
    ```
  - Use RPMs
    ```
    PACKAGE_CLASSES = "package_rpm"
    ```
  - Have package tools
    ```
    EXTRA_IMAGE_FEATURES += "package-management"
    ```
  - Build `agl-demo-platform`
    ```
    $ bitbake agl-demo-platform
    ```
- Update the package indices
  ```
  $ bitbake package-index
  ```
- Copy RPM and generate repositories
  - Use `Update-SDK-repo.py` tool [5] to copy and generate a local RPM repo.
  ```
  $ twlstd -n web --path rpm-dir \ 
  -p 8080
  ```
- Feed local RPM repo on HTTP web service
IMPLEMENTATION [2/6]
SIGNING PACKAGES IN A LOCAL RPM REPOSITORY

- To implement this function for the security and have to install Pulsar Linux SDK to use rpm-tool to sign all packages in the repository with your own private GPG keys at once.

```bash
$ rpm-tool --repo-sign --repo-dir /path/to/rpm-repo \
  --gpg-name "gpg-private-key-name" \ 
  --gpg-key /path/to/gpg-private-key \ 
  --gpg-pass "gpg-key-passphrase"
```

- For details on all command options, see The rpm-tool Command Option Reference (*)

This may be changed in the near feature. These instructions are tentative.

To build SDK-bootstrap:

- Add "meta-yomo" to your bblayers as below:

  BBLAYERS += "$\{YoMo\}/meta-yomo"

- Add runtime dependencies for init-sdk-rootfs and DNF by applying patch 0001

- Fix init-sdk-rootfs source code to be compatible with AGL v5.0.3 by applying patch 0002

- Run bitbake:

  $ bitbake sdk-bootstrap
This file is used to configure sdk rootfs

<table>
<thead>
<tr>
<th>Repo:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>m3ulcb</td>
</tr>
<tr>
<td>Baseurl:</td>
<td>file:///local_repo</td>
</tr>
<tr>
<td>TargetRepo:</td>
<td>aarch64-agl-linux,</td>
</tr>
<tr>
<td>HostRepo:</td>
<td>x86_64-linux</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDK_init_conf:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TUNE_PKGARCH:</td>
<td>&quot;aarch64&quot;</td>
</tr>
<tr>
<td>TARGET_VENDOR:</td>
<td>&quot;-agl&quot;</td>
</tr>
<tr>
<td>SDK_ARCH:</td>
<td>&quot;x86_64&quot;</td>
</tr>
<tr>
<td>SDK_VENDOR:</td>
<td>&quot;-aglsdk&quot;</td>
</tr>
<tr>
<td>SDK_VERSION:</td>
<td>&quot;5.0.3&quot;</td>
</tr>
<tr>
<td>SDK_OS:</td>
<td>&quot;linux&quot;</td>
</tr>
<tr>
<td>DISTRO:</td>
<td>&quot;poky-agl&quot;</td>
</tr>
<tr>
<td>MACHINE:</td>
<td>&quot;m3ulcb&quot;</td>
</tr>
</tbody>
</table>
IMPLEMENTATION [5/6]
INSTALL SDK ROOTFS

- Initial SDK rootfs

```bash
$ export XDT_SDK_BOOTSTRAP=/home/user/sdk-bootstrap
$ export XDT_SDK=/home/user/sdk-rootfs
$ ${DEPLOY_DIR}/sdk/x86_64-sdk-bootstrap-5.0.3.sh -y -d ${XDT_SDK_BOOTSTRAP}
$ source ${XDT_SDK_BOOTSTRAP}/environment-setup-x86_64-aglsdk-linux
$ init-sdk-rootfs -i m3ulcb-sdk-conf.sdk -o ${XDT_SDK}
```

- Use `dnf4Native` and `dnf4Target` to install native rootfs and target rootfs respectively

```bash
$ ${XDT_SDK}/m3ulcb-sdk/dnf4Native install nativesdk-packagegroup-sdk-host
$ ${XDT_SDK}/m3ulcb-sdk/dnf4Target install kernel-devsrc
```
IMPLEMENTATION [6/6]
TARGET CONFIGURATION (ON THE DEVICE)

- Make directory for repos config
  
  $ mkdir -p /etc/yum.repos.d

- Add the repo. The file name must end in '.repo' for instance *agl-eel.repo*

  [aarch64-agl-linux]
  name=aarch64-agl-linux
  baseurl=file:///aarch64-agl-linux/
  enabled=1

- Import GPG Keys to the Device

  $ rpm --import /path-to-public-key

- Install RPM packages

  $ dnf --y install packagegroup-agl-demo-platform

Files

- Cache Files
  /var/cache/dnf

- Main Configuration
  /etc/dnf/dnf.conf

- Repository
  /etc/yum.repos.d/