Brillo/Weave Part 1: High Level Introduction

Open IoT Summit
Bruce Beare
April 2016
Introduction to Brillo and Weave

- Introduction to Brillo
- Introduction to Weave
- Brillo Security Features
- Source Code Structure
- Intel Starter Boards
- Next Steps
Brillo/Weave

Brillo: Google’s OS for IoT MPU devices
- Targeted at smart homes
- Expanding to buildings and industry
- Supports MPU devices w/ min 35MB of RAM.

Weave: An open communications protocol
- Devices can connect and interact with one another, the internet, and your phone
- Ensures that access to your device and user data is secure and private.

Libuweave: A tiny Weave implementation
- Communications Stack intended to be portable to MCU based devices
Brillo: an Operating System based on Android

Simpler... Smaller... IoT Focused
- C/C++ environment
- Binder IPC
- No Java Applications, framework, runtime
- No Graphics
- 35MB Memory Footprint (minimum)
Weave* is:

- A communications platform for IoT devices
- Device setup, phone-to-device-to-cloud communication
- User interaction from mobile devices and the web
- Transports: 802.15.4 (zigbee, threads), BLE, WiFi, Ethernet, Others possible
- Schema Driven (json) Associates Weave XMPP requests with application function invocations
- Web apps may be written with Google* API support
- OAuth 2.0 Authentication, Google as AS
Weave* Device Schema

- Device manufacturers are encouraged to support a standardized schema for their device (toaster, fridge, thermostat, ...)
- IHV may extend the schema
- The user will be able to share control of their devices. Friends, 3rd party Weave (web) applications
- Cloud based apps will better integrate all of a user’s devices over time
Weave* command/data flow

- Weave (libweave)
- Binder
- Your App (service)
- Cloud APIs
- Local APIs
- Your Mobile Apps
Brillo Devices are Administered via the Google Cloud

- View Device Status
- Rename Devices
- Share Devices with apps, friends
- Send commands to Devices

FAQ: Do I have to use Google’s Cloud?

There’s nothing to prevent you from modifying Weave to use a different cloud service; but there’s nothing to make this easy either. It is unlikely that Google will release the cloud-side sources.

You could certainly add a completely different cloud back-end as well and develop a product with the Brillo OS but without Weave.
Registering (provisioning) a Brillo* Device

Connect to the Brillo’s WiFi AP via a Mobile Device
Provide information to the Device:
• OAuth* 2.0 identity
• WiFi Access Point
• PIN Code
Device contacts the Cloud
• Downloads updates
The Device is Ready
Brillo* Security Features & Requirements

Requirements are largely the same as Android:

• Google* Verified Boot
• HW Keymaster 1.0 (but limited to 2k keys)
• Cryptographic Chain of trust
• SE-Linux* Enforcement
• Limit root processes
Weave* Security Features

Weave provides link-level security and data privacy:

- SSL/TLS for Link Level Data privacy
- Device and Application access may be shared with friends or apps
- Oauth*  2.0 used for Authentication, crypto Certificates
- Google* provides the Authentication Server
## Brillo vs. Ostro

<table>
<thead>
<tr>
<th></th>
<th>Brillo</th>
<th>Ostro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Based on (Kernel)</td>
<td>Linux</td>
<td>Linux</td>
</tr>
<tr>
<td>Based on (Userland)</td>
<td>Android</td>
<td>Yocto</td>
</tr>
<tr>
<td>Over the Air Updates</td>
<td>Yes</td>
<td>Yes, but needs a cloud service to support</td>
</tr>
<tr>
<td>Compatibility with Ubuntu, Fedora Source Packages</td>
<td>Moderate (bionic library, build system)</td>
<td>High</td>
</tr>
<tr>
<td>Compatibility with Android Source Packages</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPC Mechanism</td>
<td>Binder</td>
<td>D-BUS, Linux IPC</td>
</tr>
<tr>
<td>Open Source</td>
<td>Yes (AOSP)</td>
<td>Yes (github)</td>
</tr>
</tbody>
</table>
## Weave vs. OCF/IoTivity

<table>
<thead>
<tr>
<th>Feature</th>
<th>Weave</th>
<th>IoTivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing/Available Cloud Infrastructure</td>
<td>Google</td>
<td>No</td>
</tr>
<tr>
<td>Open Source Cloud Infrastructure</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Certificate based Authentication</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Able to self-organize a network</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Request/Response Protocol</td>
<td>HTTPS (tcp)</td>
<td>CoAP (tcp, udp)</td>
</tr>
<tr>
<td>Multi-OS, Multi-Platform</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Gateway Support to non-native devices</td>
<td>Planned</td>
<td>Yes</td>
</tr>
<tr>
<td>Certification Available</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Brillo* Code Structure
Brillo* in the AOSP (Android*) Tree

Golden Rule

- Google* projects are upstream first

Board Support Package

- vendor/bsp/intel
  - Pre-builtons
    - Externally sourced, not part of AOSP
- hardware/bsp/intel
  - HALs, SOC Definitions
- device/intel/<board>
  - Board Definitions
- hardware/bsp/kernel/intel/*
  - Transition to a google common kernel

Android AOSP Projects: 506
Brillo AOSP Projects: 190
Projects common to both: 146
Projects Unique to Brillo: 42

- device/generic/brillo
- platform/external/gentoo/integration
- platform/external/gentoo/overlays/gentoo
- platform/external/gentoo/portage
- platform/external/shflags
- platform/system/peripheralmanager
- platform/tools/bdk
- product/google/common
- product/google/example-ledflasher

- platform/hardware/bsp/kernel/common/v4.4
- kernel/common

- device/intel/edison
- platform/hardware/bsp/intel
- platform/hardware/bsp/kernel/intel
- device/qcom/dragonboard
- platform/hardware/bsp/qcom
- platform/hardware/bsp/kernel/qcom
Brillo* BDK (board development kit)

Procedure:
1. Setup your Workstation (Ubuntu*)
2. Download the BDK
3. `brunch bsp download edison`
4. Structure your project/product
5. `brunch product build`
6. `brunch provision`
7. `brunch product tool fastboot reboot`
8. `brunch fastboot tool adb shell`
9. Register your device with the cloud

The `brunch` command is a python front-end for many of the product development tasks.

```
brunch product {build,gdb,provision,envsetup,tool,create} ...
  build   Build a product project from the current directory
  gdb     Run gdbclient.py for a given product
  provision Run provision-device for a given product
  envsetup Emit the environment for the product in the CWD
  tool    Run a host tool for a given product
  create  Create a product project in the current directory

brunch bsp {status,update,install,download,refresh,list} ...
  status  Get detailed information on the current status of a
           device BSP.
  update  Update an existing BSP, or Download a new BSP.
  install Alias, see brunch bsp update.
  download Alias, see brunch bsp update.
  refresh Refresh an existing BSP. Removes and re-downloads all
             packages for a specified device.
  list    List available BSPs, and their current installation
           status.
```
Brillo* BSP Structure and Contents

hardware/bsp/<vendor>

- SoC Definition .mk files
- init *.rc files for the SoC, HALs
- SELinux Policies
- HAL Source Code
- Tool Source Code
- Library Source Code
- Example Source Code
Brillo* BSP Structure and Contents

device/<vendor>/<board>

Board Specific Files - defines the lunch target
- Product Structuring *.mk files
- init *.rc scripts for the board
- Flashing scripts
- File System Mount Table
- Partitioning Table
- SELinux Policy files

/home/bjbeare/BOSP/device/intel_edison
    AndroidBoard.mk
    base_product
        AndroidProducts.mk
        devicesetup.sh
        edison.mk
        weaved.conf
        BoardConfig.mk
    build
        tasks
            flashfiles.mk
    flash_tools
        brillo-flashall-edison.bat
        brillo-flashall-edison.sh
        brillo-provision-edison.sh
        FlashEdison.json
        LICENSE.txt
        README
        fstab.device
        gpt.ini
        sepolicy
            file_contexts
            sensorservice.te
Brillo* BSP Structure and Contents

vendor/bsp/<vendor>

Prebuilt Binaries
- Downloaded via the Brunch command
- Not in an AOSP Repository
- Bootloader binaries
- Microcode/firmware
- BIOS/firmware

Include information on how to rebuild the firmware where applicable.
Brillo* Starter Boards from Intel

- Edison + Arduino*
- MinnowBoard*
- UP Board (CherryTrail)
- Edison + SparkFun*
The Intel® Edison Kit for Brillo*

• The Intel® Edison kit with the Arduino* Expansion Board was the first Brillo Starter Board
• Intel is providing the board support package
• MANY sensors and devices are available for prototyping with this board

• Brillo Starter Board
• Supported in AOSP since Nov ‘15.
Intel® Edison Board for SparkFun*

Compare to the Intel® Edison kit with Arduino:

• Same Edison CPU Module
• Simpler GPIO/Bus structure
• Stack on boards
• Smaller form factor
• Battery power available
• Many sensors available

• Brillo Starter Board
• Supported in AOSP with the February Brillo General Availability
Intel® Expansion Kit with MinnowBoard* (Baytrail-M)

- Similar Bus availability to Intel® Edison kits
- More powerful processor, Graphics and GPU enabled
- Larger RAM
- Additional storage options
- Add “lures” to expand to additional busses, sensors

- Brillo* Starter Board
- Supported in AOSP TIP
- Will be part in the BDK for General Availability
Aaeon UP Board

- Kickstarter UP-board is a likely AOSP Starter Board
- Adds MIPI/DSI, MIPI/CSI ports
- Adds USB-Peripheral (for fastboot, adb)
- CherryTrail Platform

http://up-board.org
Monitoring and Analytics

• Performance Analysis
• Crash Reporting
Monitoring: Registrations

New Weave device registrations per hour

Data source for slides in this section: my test account data
Monitoring: Registered Devices

Registered Weave devices by connection status

- online
- unknown
- registered

Days: Mar 9 to Mar 15
Monitoring: Product Versions

![Graph showing product version monitoring from March 9 to March 15, with different versions indicated by color codes.

- Orange: 0.0.0.1456584...
- Turquoise: 0.0.0.1456708...
- Red: 0.0.0.1456765...
- Green: 0.0.0.1456766082
- Purple: 0.0.0.1457724215]
Monitoring: Weave Commands Per Hour
Monitoring: Weave Commands by Name

- _ledflasher.set
- base.updateDev...
- _metrics.enable...
- _updater.check...
- base.identify

89.1%
Logging

- Crash Reporting
- CPU Usage
- Use Time
- Data Partition Use
- Memory Used
- Page Faults
- Storage Activity
Next Steps

Likely futures for Brillo:
- Java*
- Android* Framework
- Camera Support
- Display via surface flinger
- Bluetooth*
- Voice Command and Control
- Other connectivity options

Intel Exploring:
- JavaScript
- Node.js
- NW.js (for display)
What’s in Part 2 of this Talk?

• Hardware Access Layer
• Code Labs
  • Hello World Brillo IO Programming
  • Hello World Weave Integration
  • Enable Google Services and APIs
  • Web Service Development
Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Intel, the Intel logo and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

© 2016 Intel Corporation.