The OpenBMC Project
Chris Austen / Senior Software Manager / IBM
Devils in the Details
# BMC Development

<table>
<thead>
<tr>
<th>Feature</th>
<th>Outsourced</th>
<th>Home Grown</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control of features</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Fast bug fixes</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Control of security updates</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Cheaper to develop</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
</tr>
<tr>
<td>Simple infrastructure support</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
</tr>
<tr>
<td>Open Source Community Code Reviews</td>
<td>X</td>
<td>X</td>
<td>✔</td>
</tr>
<tr>
<td>Unique look</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Haven’t I heard of this before?
What is OpenBMC

Control the Functions in your OpenBMC...

• Built on Yocto-Linux

Simple infrastructure for you

• IPC via D-Bus
• Automatic REST API from D-Bus introspection
• Users via REST and IPMI
What is OpenBMC

Contributions back to Open Source

• Fully upstreamed U-Boot and Kernel
  – ASPEED AST2400 and AST2500

Struggling to keep up with latest technologies?

• U-Boot 2016
• Linux 4.10
• Yocto 2.1
Without D-Bus

- Code all IPC yourself
- Startup / Recover logic
- Multi-company collaboration?

© 2015 Javier Cantera - this work is under the Creative Commons Attribution ShareAlike 4.0 license
With D-Bus

- Rapid community collaboration
- Rapid Prototyping
  - Bindings for C/C++, Python, etc
- Compete IPC integration with systemd
OpenBMC Dbus Interface Examples

Attach an interface, get the benefits in Dbus and REST

xyz.openbmc_project.Sensor.Threshold.Warning
• Broadcast notification when value trips
• “Min”, “Max”

xyz.openbmc_project.Inventory.Item
• VPD Properties
• “Present”
Making ODM development easier

In 2 Ways...

XML Integration
- Manifest for your hardware
- Define wiring
- Define environmental limits
- IPMI entity IDs
XML Integration

```xml
<targetInstance>
  <id>MAX31785.hwmon2</id>
  <type>unit-hwmon-feature</type>
  ...
  <attribute>
    <id>HWMON_FEATURE</id>
    <default>
      <field><id>HWMON_NAME</id><value>fan1</value></field>
      <field><id>DESCRIPTIVE_NAME</id><value>fan0</value></field>
      <field><id>WARN_LOW</id><value>1000</value></field>
      <field><id>WARN_HIGH</id><value>80000</value></field>
      <field><id>CRIT_LOW</id><value></value></field>
      <field><id>CRIT_HIGH</id><value></value></field>
    </default>
  </attribute>
</targetInstance>
```
Making ODM development easier

In 2 Ways...

XML Integration
- Manifest for your hardware
- Define wiring
- Define environmental limits
- IPMI entity IDs

YAML definitions for REST
- Human readable properties
- Documentation driven code
**YAML, Code and Schema Documented**

**description:**
Implement to provide event/error entry attributes.
This interface should be instantiated for the `phosphor::logging` namespace.
This interface is a collection of objects, therefore it is required to implement `org.freedesktop.DBus.ObjectManager` on the logging namespace root. Optionally, implement `org.openbmc.Object.Delete` to allow the deletion of individual entries.

**properties:**
- **name:** Id
  **type:** `uint32`
  **description:**
  The error event entry id number.
- **name:** Timestamp
  **type:** `uint64`
  **description:**
  Commit timestamp of the error event entry in milliseconds since 1970.
Defined REST Schema

• /
• /list
• /enumerate
• /attr/<property>
• /action/<method>
• /schema
Let's see an example

curl -b cjar -k https://bmc/xyz/openbmc_project/inventory/

{
  "data": [
    "/xyz/openbmc_project/inventory/system"
  ],
  "message": "200 OK",
  "status": "ok"
}"
Let's see an example

curl -b cjar -k
https://bmc/xyz/openbmc_project/Inventory/system/chassis/motherboard/cpu0

{
    "BuildDate": "",
    "Manufacturer": "IBM",
    "Model": "",
    "PartNumber": "01HL322",
    "Present": 0,
    "PrettyName": "PROCESSOR MODULE",
    "SerialNumber": "YA3933741577",
    "Version": "EC:10"
}
Quick Test Drive?
QEMU == No Hardware Required
Automated Build Verification

- Dedicated Test Development Team
- Every commit... tested
- Enterprise quality
Covered Interfaces -- 2016

- REST
- BMC
- VGA
- SSH Console
- IPMI
- KVM
- Host
Covered Interfaces -- 2017

- REST
- IPMI
- HTTPS
- BMC
- VGA
- SSH Console
- IPMI

BMC

Host
Features Provided

- Power Control
- Remote Console
- IPMI
- Sensors
- LED Management
- Host Watchdog

- VPD Inventory
- Power and Cooling Management
- Event Logs
- Zeroconf discoverable
- Simulation
Features in development

• Enhanced Code Update
• Full IPMI 2.0 Compliance with DCMI IPMI
• User Interface looking for Sponsor Users
Features of the future... Join Us

- OpenCompute Redfish Compliance
- Remote KVM
- Remote USB
- Additional Board Support
- OpenStack Ironic Integration
- QEMU enhancements
- Quarantine Room for full Hardware CI
- Designs beyond the data center
Thank You

Code:
- https://github.com/openbmc

Continuous Testing
- https://openpower.xyz

Contact
- Mail: openbmc@lists.ozlabs.org
  IRC: #openbmc on freenode.net
- Riot: #openbmc:matrix.org