GENIVI+OCF Cooperation
May 11, 2017  |  Connect your car to SmartHome & Wearables

Ziran Sun for Philippe Coval
Samsung Open Source Group / SRUK
ziran.sun@samsung.com
philippe.coval@osg.samsung.com
Agenda

- IoT Vehicle demo
- Demo walk-through
  - OCF and IoTivity
  - GENIVI Development Platform (GDP)
  - OCF Automotive
- Example codes and demos
- IoTivity cloud and services
- Q&A
Smart Home+Automotive IoT ShowCase
[CES 2017] Smart Home+Automotive IoT ShowCase

https://www.youtube.com/watch?v=3d0uZE6lHvo&feature=youtu.be#smarthome-ces2017
Who are playing parts?

- **OCF and IoTivity**: IoT SW platform
- **Samsung/OSG**: SmartHome, personal devices (Tizen)
- **GENIVI**: GDP (Yocto based OS), Automotive
- **OCF Automotive**: IoT & Automotive
- **JRL, Intel, W3C…**
OCF and IoTivity
The Open Connectivity Foundation (OCF)

- An industry group with leading companies at all levels to develop **standards** and **certification** for IoT devices.
- Focus on **interoperability** and seamless connectivity between devices.
- Encourages users to collaboratively develop data models for any IoT device using online tool OneIoTA.
- Sponsor **Open Source** Implementation: IoTivity (under Apache 2.0)
Interoperability in IoT

• OCF addresses Interoperability challenge in IoT communications at all levels.
• Open specification that anyone can implement.
• “Brings Massive Scale to IoT Ecosystem” as quoted by press
OCF Specification and Certification

• Specifications
  ➢ Infrastructure including core framework, security and remote access etc.
  ➢ Resource Model
  ➢ Domain Specific Resource Specification
  ➢ Bridging between OCF devices and other ecosystems

• Certifications
  ➢ Conformance test - Each device proves conformance to specifications.
  ➢ Interoperability test - Each device proves interoperability with other devices.
OCF Resources Data Models: oneloTa

- Resource is identified by an URI
- Composed of properties
- Declared by a ResourceType
- Operations: CRUD+N
- Use existing known resource models from oneloTa.org repository or create new ones

```
oic.r.sensor.illuminance.json
/* … */ "definitions": {
   "oic.r.sensor.illuminance": {
      "properties": {
         "illuminance": {
            "type": "number",
            "readOnly": true,
            "description": "Sensed luminous flux in lux."
         }
      }
   }
} /* … */
```
IoTivity

- An open source communications framework for IoT, hosted by the Linux Foundation, and sponsored by the OCF.
- Implementation follows OCF specification using Common Protocols with RESTful architecture + Connectivity Abstraction + high level services
- Rely on existing standards (CoAP, TLS), or OSS libs
- Rules: No unimplemented features in Specification
IoTivity Architecture

Rich Device
API (C++/Java)

Service Layer
Device Management
Low Power Management
Data Management
Group Manager
Device conf.
Easy Setup
Resource Directory
Resource Hosting
Soft Sensor
Noti. Service
Protocol Bridge

Resource Encapsulation
Resource Container

Base Layer
Discovery
Messaging
Security
Resource Introspection
CoAP
BWT
Provisioning
Multicast
Presence
Messaging
Remote
DTLS
SRM

Lite Device
Sensing/Control Application

Base Layer
Messaging
Security
CoAP
DTLS
SRM

Discovery
Resource Introspection
Multicast/Scan

Discovery (Multicast)/ Messaging (CoAP)
Flow: Create, Read, Update, Delete, Notify

IoTivity Server

<table>
<thead>
<tr>
<th>Initialization as server</th>
<th>Registration of resource</th>
<th>Handling new clients + CoAP</th>
</tr>
</thead>
</table>

Local IP Network

IoTivity Client(s)

<table>
<thead>
<tr>
<th>Initialization as client</th>
<th>Discovery of resource</th>
<th>Set/Get/ing properties values</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Handling new requests</th>
<th>Notify updated resource</th>
<th>Observe resource change &amp; Handling properties</th>
</tr>
</thead>
</table>

UDP Multicast

POST/PUT/GET

OBSERVE

13 | May 11, 2017 | Samsung Open Source Group
IoTivity and GENIVI Integration
IoTivity and GENIVI Integration

• Yocto is Linux Foundation project to create custom GNU/Linux system
  o Open Embedded community provides many OSS sorted in layers (named meta-*):
  o Collections of recipes(package) to build package:
    o i.e: meta-oic layer is shipping IoTivity library
• GDP is GENIVI's Yocto based distro to provide full Automotive OS (IVI...)
  o Integrates meta-oic for iotivity (c/c++) and meta-iot-web for (for iotivity-node)
  o And supporting Board Support Packages (intel, renesas, raspberrypi, etc)
Timeline

2014-12-31: meta-oic 0.9.1 Initiated by Kishen Maloor (Intel) (with demo for edison)
2016-01-31: FOSDEM: Presented how to use meta-oic on Tizen Yocto (Tizen fan)
2016-04-27: GENIVI AMM : Presented demos (fan+map+wearables on 1.1.1), +RVI
2016-05-08: meta-oic 1.1.1 integrated in GENIVI
2016-05-27: AGLF2F meeting, “ocf-automotive” project introduced
2016-09-14: meta-oic 1.1.1 Philippe Coval (Samsung) new contributor
2016-09-21: meta-oic 1.1.1 integrated in AGL
2016-12-20: meta-oic 1.2.0 integrated in GENIVI and AGL
2017-01-05: CES2017, GENIVI+Smarthome+Wearables demos
2017-02-04: FOSDEM: Presented “streetlight+cloud” use case on AGL 3.0
2017-02-15: GENIVI announced partnership with Open Connectivity Foundation
2017-03-20: meta-oic 1.2.1+ : Samsung OSG keeps maintaining it
OCF Automotive profile's mission

● Provide OCF technology for connected cars, by proposing
  ○ A common definition of vehicle resources
  ○ A common way to interact with those (inside or outside vehicle)
  ○ Based on or bridging to existing standards

● Cooperative effort with existing FLOSS Automotive projects:
  Tizen, GENIVI, AGL, W3C, RVI ...
What is “meta-ocf-automotive”

- Playground for OCF and Automotive R&D experiments
  - connecting Automotive platforms such as GENIVI, AGL, Tizen etc
  - on various hardware: SBC (Raspberry Pi {0,1,2,3}, ARTIK10…)
  - to other products: SmartHome, Mobile, Wearable

- “Real world” integration/validation tests
  - for spotting issues before release (on different OS: Yocto, Tizen etc)

- Tutorial of demo codes to learn IoTivity, Yocto, Tizen

- Entry point: [https://wiki.iotivity.org/automotive](https://wiki.iotivity.org/automotive)
Interaction with **Tizen** products

- Tizen is an Operating System based on FLOSS
- Shipped into consumer electronics products
- Tizen and IoTivity
  - Tizen:3 contains as platform package (.rpm)
  - Tizen:2 can ship shared lib into native app (.tpk)
    - For Samsung Z{1,2,3} (Tizen:2.4:Mobile)
    - Samsung GearS2, S3 (Tizen:2.3.2:Wearable)
Example codes and demos
Check using samples apps

- Various examples are shipped from upstream:
  - ls /opt/iotivity*
- Playback smart light example scenario on loopback
- Open 2 sessions (hint: use GNU screen)
  - Server: cd /opt/iotivity/examples/resource/cpp/ && ./simpleserver
  - Client: cd /opt/iotivity/examples/resource/cpp/ && ./simpleclient
- More
  - https://wiki.iotivity.org/yocto
  - https://wiki.iotivity.org/examples
Base example: Resource discovery

- branch=example/master (src/example/master/README.md)
  - Server register a “dummy” resource under “/ExampleResURI” endpoint
  - Client discover and list all resources' endpoints served in local network
  - GNUmake is used to build it
  - Systemd service provided to start it once installed

- branch=example/packaging is based on previous one
  - Yocto Bitbake recipe
  - Tizen RPM spec file
  - Debian/Ubuntu packaging files too (more welcome)
Geolocation example: Observation

- Branch “geolocation/master” is based on “example/packaging” and adapted:
  - “/GeolocationResURI” endpoint
  - Uses OCF/OneIotA normalised resource type: oic.r.geolocation
  - Fake GPS that update position continuously (back and forth)
- ./bin/server: is updating “fake” position and notifying it
  - m_Representation.setValue(); OCPlatform::notifyAllObservers(…);
- ./bin/observer: observe changes (IoTObserver::onObserve)
  - geolocation: 48.1043, -1.6715
- ./bin/client : get value using GET (m_OCResource->get)
Derivate to Tizen app

- “geolocation/tizen/mobile/2.4/master”
- Port to tizen native app: support SDK build files, app manifest files
  - + GUI using EFL's Elementary map widget (inspired from SDK sample)
- Need to rebuild IoTivity’s shared lib (to be bundled in tpk):
  - Use helper script build rpm and unpack lib
  - ./tizen.mk ; ls lib/* .so
  - ./tizen.mk run # deploy on root device (ie TM1)
- More details: https://wiki.iotivity.org/tizen
Switch Example

- “switch/master” subproject
  - Shows actuator update using POST
- Port to Tizen devices
  - Add ELM GUI, Tizen SDK build files, tizen-helper script (that rely on CLI SDK)
  - To deploy on commercialized device use tizen-studio IDE SDK (+ cert eclipse plugin)
  - Tizen Z, TM1: Branch “sandbox/pcoval/tizen/mobile/2.4/latest”
  - GearS2 branch “sandbox/pcoval/tizen/wearable-2.3.1” (is based on previous one)
  - GearS3 branch “sandbox/pcoval/tizen/wearable-2.3.2” (is based on previous one)
Switch Example: Resource update

```
OCPlatform::Configure(...);
OCPlatform::registerResource(...);

handleEntity(OCResourceRequest) {
    switch entityHandlerRequest->method {
        case 'POST': // Update resource
            // update actuator resource (physically)
            ...
            OCPlatform::sendRespone(response);
        }
    }

OCResource::post(rep, callback);
OCPlatform::findResource(...);
OCPlatform::configure(...);
```

- Client controls actuator:
  - Set resource's value
- Server is handling request
  - and responding
Want more?
A Vehicle to Infrastructure notification service

```javascript
function handle(illuminance) {
    if (gThreshold > illuminance) {
        var data = {
            illuminance: illuminance,
            latitude: gGeo.latitude, longitude: gGeo.longitude
        };
        sender.send(data); // { ARTIK's client.post(url...); }
    }
}

client.on("resourcefound", function(resource) {
    if ("/IlluminanceResURI" === resource.resourcePath) {
        resource.on("update", handle);
    } else if ("/GeoLocationResURI" === resource.resourcePath) {
        resource.on("update", function(resource) {
            gGeo = resource.properties;
        });
    }
});
```

30 | May 11, 2017 | Samsung Open Source Group
https://youtu.be/3L6_DbMLJ1k#iotivity-artik-20170204rzr
IoTivity Cloud and Services
IoTivity Clouds

- Cloud Interface
- Authentication
  - OAuth2
- Message Queue
  - Publish
  - Subscribe
- Directory (RD)
IoTivity Services

- A common set of functionalities to application development.
  - Resource Container
  - Notification
  - Resource Encapsulation
  - Scene Manager
  - Easy setup
Summary

• OCF targets Interoperability challenge in IoT by specifications and certifications
• Open Source project IoTivity implements OCF specification
• meta-oic is the key for integrating IoTivity and it has currently been maintained by Samsung OSG
• OCF-automotive project has provided “meta-ocf-automotive” to encourage further R&D work in automotive domain
• IoTivity native cloud extends connectivity to global
• IoTivity Service make application development easier
References

● Entry points:
  - https://openconnectivity.org/industries/automotive
  - https://wiki.iotivity.org/automotive
  - http://elinux.org/Category:Automotive
  - https://wiki.tizen.org/wiki/Domains/Automotive

● Contributions
  - http://git.yoctoproject.org/cgit/cgit.cgi/meta-oic/log/?qt=author&q=coval
  - https://github.com/GENIVI/genivi-dev-platform/pull/54
Thank you!