IoTivity: The Open Connectivity Foundation and the IoT Challenge

Thiago Macieira

Embedded Linux Conference / Open IoT Summit – Berlin, October 2016
Who am I?
About the Open Connectivity Foundation

**Specification**
Defines OCF framework including standard model for IoT devices, apps & services to interact

**IoTivity Open Source**
Delivers reference implementation of OCF framework & translation layers for non-OCF devices

**Certification**
Ensures interoperability via compliance and interop testing

Stop fragmentation and increase device orchestration by creating a common standard for IoT device connectivity

Ease developer burden through open source code availability and royalty-free license

Ensure interoperability through a formal testing and certification program
OCF Vision

Deliver an IoT connectivity standard that is...

- Open
- Free
- Seamless
- Technology Agnostic
- Fair & Accessible
- Cross Industry
- More Secure
- Structured
OCF Current members

Diamond
ARRIS
CableLabs
Canon
CISCO
Electrolux
GE Digital
Intel
Microsoft
Qualcomm
Samsung

Platinum
AFFINEGY
Atmel
CAICT
COMCAST
Dell
Honeywell
HP
IBM
MEDIATEK
neustar
SOMFY
TPVISION
twobulls
xped
ZTE

Liaisons
ATSC
CABA
CEA
DTG
DVB
dlna
EEBUS
enocean
industrial Internet® CONSORTIUM
IPSO Alliance
LONMARK® International
Thread
ULE Alliance
W3C®

For Gold, Basic and non-profit members, see openconnectivity.org
Where the stack sits

Applications & Services
Data & Control Points

OCF Comms Framework
(Single Resource & Data Model)

Translation Layers

* Extensible

- Wi-Fi
- 802.15.4
- IP over BLE
- Bluetooth Low Energy (BLE)
- Z-Wave
- ZigBee
- IP over BLE
- 802.15.4
- Bluetooth Low Energy (BLE)
- Z-Wave
- ZigBee
- 802.15.4
OCF Protocol Stack

Application Layer
- OCF client
- OCF intermediary
- OCF server
- Application-specific resources

Resource Layer
- Security Resource Manager (SRM)
- Security Resources

Connectivity Layer
- Session management (CoAP, DDS, XMPP, MQTT, etc)
- Session protection (e.g., DTLS)
- UDP, TCP, Bluetooth* profile
- IP over Wi-Fi*, IP over 802.15.4, Bluetooth, …

Security Enforcement

- Based on standard technologies
  - Does not require TCP (only UDP)
- Security built in from the start
  - “Security 2.0” will be end-to-end
- Hardening left as an exercise for the manufacturer
Core Protocol

- **OCF adopted RESTful APIs**

- **Core framework defines 2 logical roles that devices can take:**
  - OCF Server: A logical entity that exposes hosted resources
  - OCF Client: A logical entity that accesses resources on an OIC Server

- **OCF Client**
  1) Initiate an transaction (send a request)
  2) access an OCF Server to get a service

- **OCF Server**
  1) host a Resource
  2) send a response
  3) provide a service
Organisation of an OCF device

Device concept:

Physical Device e.g., lightbulb

OCF Device 1
- /oic/res
- /oic/d
- /oic/mnt

OCF Device 2
- /oic/res
- /oic/d
- /oic/prs

Resource URI: /oic/p
- rt: oic.wk.p
- if: oic.if.r
- n: homePlatform
- policy: bm:11
- pi: at1908
- mnmn: Samsung

Mandatory
Optional
Device Example: Light Device (oic.d.light)

- **Example overview**
  - Smart light device with i) binary switch & ii) brightness resource

- **Device type: Light device (oic.d.light)**

- **Associated resources**

<table>
<thead>
<tr>
<th>Device Title</th>
<th>Device Type</th>
<th>Associated Resource Type</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light</td>
<td>oic.d.light</td>
<td>/oic/res (oic.wk.core)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/oic/d (oic.d.light)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Binary switch (oic.r.switch.binary)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brightness (oic.r.light.brightness)</td>
<td>No</td>
</tr>
</tbody>
</table>
Ownership transfer and bootstrapping

Device Gets on the Network

OBT Discovers the Device

Device is Un-owned

Ownership Transfer

Bootstrapping / Provisioning (ACLs, certs)
OCF Data Models

• Starts with definition of individual elements
  - Built on generic description strategy (e.g., RAML, JSON schemas)
  - Starts with physical properties (e.g., temperature, mass, color …)

• Devices are comprised of collections of elements / properties
  - Including previously defined devices

• Abstract devices can also be defined
  - (e.g., Joe’s house, upstairs bedrooms …)
oneIoTa.org

- A crowd-sourced Integrated Development Environment (IDE)
  - RAML & JSON validated and syntax aware editors with shared editing

- Automatic support for derived models and multiple organizations

- Submission and approval process per organization
IoTivity Project Overview

• An Open Source Project, hosted by the Linux* Foundation
  - License: Apache Version 2.0

• Goal: implement the reference implementation of OCF specification

• Meritocratic, fair and open development process
IoTivity Main reference implementation

- An open source software framework implementing OCF Standards
- Available on Android*, Linux*, Tizen* and Windows*
- Notable features:
  - CoAP over TCP and over Bluetooth* LE
  - Bridge plugins to other ecosystems
  - Cloud integration
Other IoTivity reference implementations

**IoTivity for constrained devices**

- Designed from scratch for small devices (e.g., Intel® Quark™ family)
  - Static memory allocation
- Fully compatible with OIC 1.1 specification and main IoTivity
- Support for Linux* and Zephyr

**IoTivity for Node.js***

- API in JavaScript*, provided as an npm package
- “Feels” native for Node.js developers
- Easy to integrate with other Node.js packages for richer experience

See session on IoTivity Constrained
IoTivity for Node.js* API Sample

**Client**

 Promise findResources();
 Promise retrieve(id);
 Promise update(resource);
 Promise observe(id);
 Events:
   resourcefound

**Server**

 Promise<resource> register(data);
 Events:
   retrieverequest
   updaterequest
   observerequest

**Resource**

 Events:
   update
   delete
IoTivity for Node.js* Example Code

```javascript
var device = require("iotivity-node")();

device.configure({role: "client");

device.on("resourcefound", function(event) {
    console.log("client: resource found %s", event.resource.id.path);

    if (event.resource.id.path == "/a/light") {
        device.retrieveResource(event.resource.id)
            .then(function(resource) {
                resource.properties.on = !resource.properties.on; // toggle
                device.updateResource(resource).then(function() {
                    console.log("client: update OK");
                    process.exit(0);
                });
            });
    }
});

device.findResources();
```
Other IoTivity Projects

• Bridge to UPnP

• Bridge to AllJoyn*

• Testing tool, with network simulation
Get Involved!

• Participate in developing the reference implementation IoTivity (https://www.iotivity.org/get-involved)

• Participate in creating the specification & certification program OCF (http://openconnectivity.org/join)

• Participate in developing the OCF data models oneIoTa tool (https://www.oneiota.org)
Thiago Macieira
thiago.macieira@intel.com
http://google.com/+ThiagoMacieira