IoTivity Introduction

Martin Hsu

Intel Open Source Technology Center
Agenda

- Overview
- Architecture
- Base Layer & APIs
- Primitive Services & APIs
IoTivity Overview

- An open source software framework implementing OCF Standards
- Ensures seamless device-to-device and device-to-cloud connectivity to address emerging needs of IoT
- Licensed under Apache License Version 2.0
- Available on TIZEN, Android, Arduino, Linux(Ubuntu) Platforms
IoTivity – High Level Architecture

**Key Goals**

- Common Solution
- Established Protocols
- Security & Identity
- Standardized Profiles
- Interoperability
- Innovation Opportunities
- Necessary connectivity

**IoTivity Profiles**

**IoTivity Framework**

**IoTivity Connectivities**
IoTivity Base Layer & APIs
IoTivity – High Level Architecture

**Key Goals**

- Common Solution
- Established Protocols
- Security & Identity
- Standardized Profiles
- Interoperability
- Innovation Opportunities
- Necessary connectivity

**APIs**

- (C/C++/Java/JS)

**Service Layer**

- Device Management
- Low-Power Management
- Data Management

**Base Layer**

- Messaging
- Security
- Discovery

**Lite Device**

- Sensing/Control Application

- Resource Encapsulation
- Resource Container

**Rich Device**

- Consumer
- Enterprise
- Industrial
- Automotive
- Health

**IoTivity Profiles**

**IoTivity Framework**

**IoTivity Connectivities**
Discovery Subsystem

- **CoAP**: Constrained Application Protocol
- **IANA**: Internet Assigned Numbers Authority

### Internet Connectivity and Discovery Mechanism

<table>
<thead>
<tr>
<th>Connectivity</th>
<th>Discovery Mechanism</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFi &amp; Ethernet (over IP)</td>
<td>IP Multicast</td>
<td>CoAP Multicast Port: 5683 (Assigned by IANA) CoAP Secure Port: 5684</td>
</tr>
<tr>
<td></td>
<td>IP Unicast over UDP</td>
<td>Precondition: OIC Server Address &amp; Port are known</td>
</tr>
<tr>
<td>Bluetooth (EDR &amp; BLE)</td>
<td>Using Scan &amp; Advertise</td>
<td>OCF Specific Service UUID</td>
</tr>
</tbody>
</table>

### CoAP Features
- Open IETF Standard (RFC 7252)
- Compact 4 Byte Header
- UDP (Default), SMS, TCP Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-In Discovery

CoAP: Constrained Application Protocol
IANA: Internet Assigned Numbers Authority
Note platform hardening not part of the OCF Specs & IoTivity implementation.
Messaging - Connectivity Abstraction

- **CA Control Component**
  - Target network selection, interface control & monitoring
  - CoAP message serialization & parsing
  - Block-wise messaging flow control

- **Transport Adapter Component**
  - Data transmission over UDP, TCP, BLE(GATT), BT(SPP) & NFC
  - Secure data exchanging using DTLS

- **Platform Adapter Component**
  - Ubuntu, Wi-Fi, Ethernet and BLE
  - Android Wi-Fi, BLE and BT
  - Tizen Wi-Fi, BLE and BT
  - Arduino Wi-Fi, Ethernet and BLE
Messaging - Remote Access over XMPP

**Feature**

- Remote client discover & securely interface with resource servers when not on same subnet
- Adheres to access control policies
- End-to-End Secure

<table>
<thead>
<tr>
<th>Device Type</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light weight (LW) Device</td>
<td>Accessible within subnet. No RA, require GW/proxy device for access</td>
</tr>
<tr>
<td>Constrained RA (cRA) Endpoint</td>
<td>RA access for non latency-sensitive, low BW applications</td>
</tr>
<tr>
<td>RA Endpoint (RA)</td>
<td>Full RA access</td>
</tr>
</tbody>
</table>
IoTivity Primitive Services & APIs
IoTivity – High Level Architecture

Key Goals

- Common Solution
- Established Protocols
- Security & Identity
- Standardized Profiles
- Interoperability
- Innovation Opportunities
- Necessary connectivity
Purpose of Primitive Services

- Provides easier and simpler APIs for App developers (Heavy Lifting done by Framework)
- Mostly designed to run on Smart or Controller devices
- Uses the IoTivity Base APIs
Scene Manager

Helps Users to create a Scenario or Scene for controlling Multiple IoT devices & their functionality

e.g.
• Away Home – All Lights turned off, Doors locked
• Watching Movie – Living Room lights off, TV On, Speaker On
Simulator Service

Simulating different OCF resources

- OCF resources can be simulated, Using resource model definition (RAML) files.
- Manages creation, deletion, request handling and notifications for OCF resources.

Sending different requests to verify features supported by OCF resources

- Searching for different types of resources available in the network.
- Sending different types of requests both manual and automatically and displays the response payload received.

Feature

• Server
  - OCF resources can be simulated, Using resource model definition (RAML) files.
  - Manages creation, deletion, request handling and notifications for OCF resources.

• Client
  - Searching for different types of resources available in the network.
  - Sending different types of requests both manual and automatically and displays the response payload received.
MultiPhy Easy Setup

- Schematic illustrations

1. Out of box
2. Read QR code using Smartphone’s camera
3. The device and smartphone are connected via WiFi
4. Smartphone shows a list of WiFi APs to user and user selects one
5. The device gets connected to the desired WiFi AP
How many subscriptions thin device could support with its constrained system resource?

How many subscriptions thin device could support with its constrained system resource?

Solution

- Offloads request/data handling from remote clients
- Reduces the power consumption of resource constraint device

Thin Device enhances its lifetime delegating its resource subscriber to richer hosting device

- Hosting(Rich) device
- Thin(Light) device
Supporting Material
# Resource Encapsulation

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Broker</td>
<td>• Remote Resource Presence check (regardless of Remote Server supporting presence feature)</td>
</tr>
<tr>
<td></td>
<td>• Provide consistent reachability management for discovered resource of interest</td>
</tr>
<tr>
<td>Resource Cache</td>
<td>• Maintains last information of Remote Resource (regardless of Remote Server is observable)</td>
</tr>
<tr>
<td></td>
<td>• Data Centric API (Send/Recv Message Getter/Setter, Data Cache)</td>
</tr>
<tr>
<td>Server Builder</td>
<td>• Att. setter to provide easy way to create resource</td>
</tr>
<tr>
<td></td>
<td>• Changes “msg Handling” to “Data Setting” for users</td>
</tr>
<tr>
<td></td>
<td>• Monitors value of attributes so that notify-back for observation whenever attribute has changed</td>
</tr>
</tbody>
</table>

IoTivity Base
Low Power Management – Resource Directory

- Constrained device that needs to sleep and can not respond to multicast discovery queries
  - Discovery of RD server
  - Publish Resource to RD
  - Update / Delete Resource
Cloud-Native Architecture for IoT

- Devices can reach the cloud **directly**.
  - Devices can self-organize if the cloud is not accessible.
- Architecture and protocols don’t have to be replaced when device deployment changes from local-only to cloud-connected.
- Encourages end-to-end micro-services.
Onboarding & Provisioning Call Flow

[Diagram showing the call flow between Admin Device, Device ID: Admin0, Owned Device, Device ID: Li, Admin Device, Device ID: Light0, and Owned Device, Device ID: Door0.]

- **Owned Discovery**
  - GET /nl/sec/dxm?Owned="TRUE"
  - RSP [{..."DeviceID": "Light0"}, ...]

- **Provisioning ACL**
  - DTLS Handshake w/ OwnerPSK
  - POST /nl/sec/crd
    - ["subject": ":", "permissions": "CRUD", "owner": "Admin0"], ...
  - RSP 2.01

- **Create credential resource for Device1 and Device2.**
  - DTLS Handshake w/ OwnerPSK
  - POST /nl/sec/crd
    - ["credtype": 1, "pdata": "ps..."]
  - RSP 2.01
Protocol Bridge using Resource Container

- Integrates non-OCF resources (Bundle)
- Handles dynamic loading of resource bundles & dynamic creation of resources
- Supports C++ .so files & Java .jar files
- Common configuration for bundles and configured resources

- Designed to work devices with non OCF devices
- Enables control of legacy devices which are already in market with existing APIs using a OIC complaint device

1. Loads configuration
   - containerConfig.xml (resource instance specific configuration)

2. Loads resource bundles
   - hueToOCF.so

OCF bridge (with resource container)

Maps OCF to Hue light

OCF light interface

OCF light interface

OCF light interface
Interoperability …

• **Full interoperability** from the connectivity layer up to the service layer is the only way to truly guarantee a satisfactory UX

• Interoperability at the Connectivity and/or Platform layer only provides partial interoperability which can ultimately lead to fragmentation

![Diagram showing interoperability levels](image-url)
Consumer Radio-Based Standards

- Applications & Services
- Data & Control Points
- Profiles, Data & Resource Models
- Comms Protocols
- Transports

<table>
<thead>
<tr>
<th>Profiles, Data &amp; Resource Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensible</td>
</tr>
<tr>
<td>ZigBee</td>
</tr>
<tr>
<td>Thread</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comms Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth® Low Energy</td>
</tr>
<tr>
<td>Z-Wave</td>
</tr>
<tr>
<td>802.15.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transports</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP = 6LoWPAN</td>
</tr>
</tbody>
</table>

26
Security Building Blocks

Provisioning Manager
- Provisioning Database

Provisioning Manager
- Provisioning Database

Secure Resource Manager
- Resource Manager

Resources
- ACL
- DOXM
- PSTAT
- SVC
- AMACL
- CRED
- CRL

Credential Generator
- CK manager
- Ownership transfer
- Just Works
- Random PIN

Policy Engine
- Ownership transfer
- Just Works
- Random PIN

Just Works
- Ownership transfer
- Just Works
- Random PIN

Random PIN
- Ownership transfer
- Just Works
- Random PIN