

Perspective on the WoT architecture and API in consideration of 4 characteristic implementation models

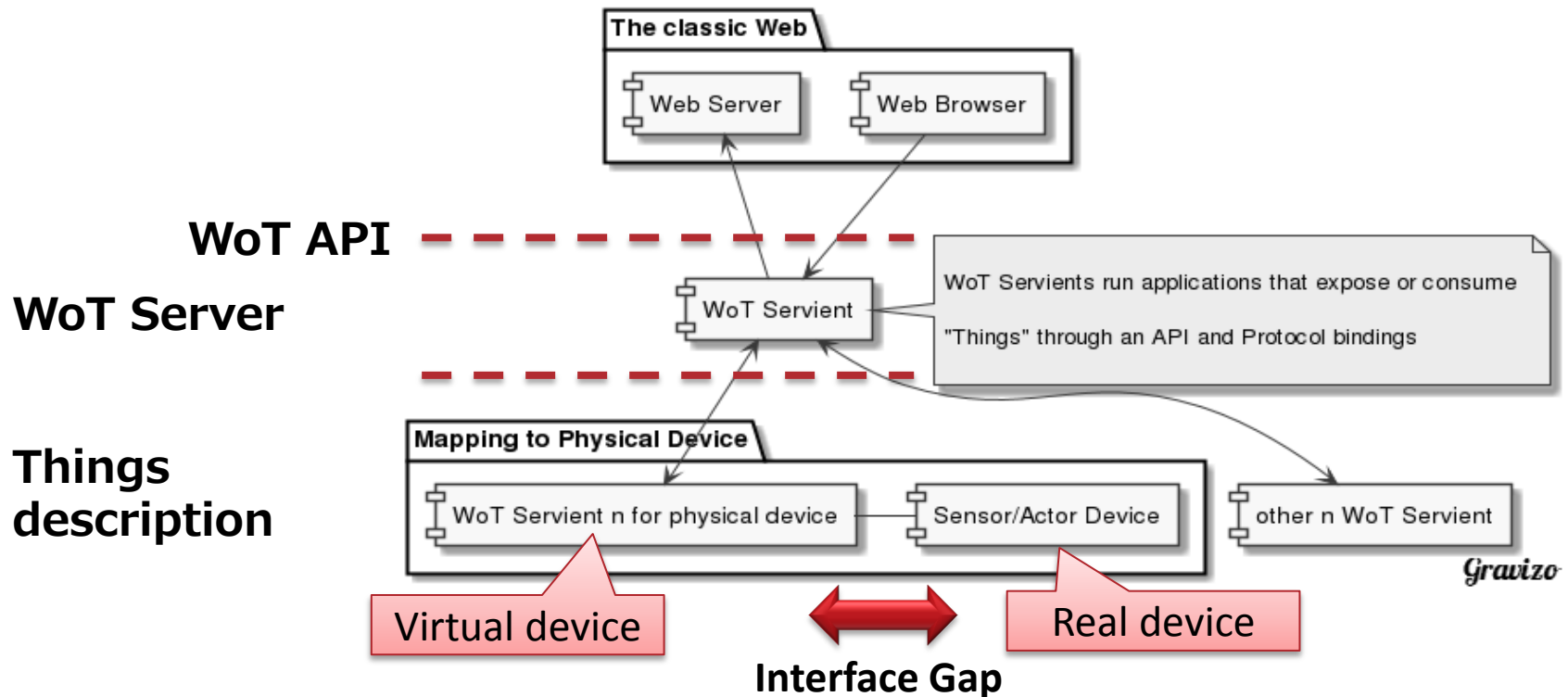
October 29-30, 2015

Fujitsu Laboratories / Fujitsu

Ryuichi Matsukura

WoT Skelton Architecture

- Model consists of two layers (i.e. WoT Servient and Virtual Device)
- WoT Servient exposes WoT-API
- Interface gap between Virtual device and Real device needs to be considered



- WoT-API discussion needs to be based on Real device interface
 - Support from Legacy-devices is a prerequisite for wide-spread adoption of Web of Things, at least initially.
 - Minimization of the gap between Virtual device and Real device is a substantial factor in designing Virtual device and Things Description Specification.
 - Discussion based on the following two is necessary to facilitate the support from Legacy devices.
 - Architecture involving WoT servient and Real devices.
 - Real Device Interfaces.
- Implementation Model (WoTServient ~ Real device)
 - All four models Kajimoto-san presented in Sunnyvale need to be considered.
 - The four models differ in arrangement in each implementation, however, are the same in functional models.

- Thing Description needs to correspond to Real device interface
 - Legacy device types that should be considered include the followings
 - Ones whose device interfaces are defined as wireless communication specifications, such as ZigBee/Bluetooth.
 - Standards for smart home and building: KNX/SEP2.0/ECHONET Lite
 - Sensor device interfaces for Automobiles.
 - Device interfaces for Factories and Industries.

WoT Architecture Summary

■ WoT Servient consists of:

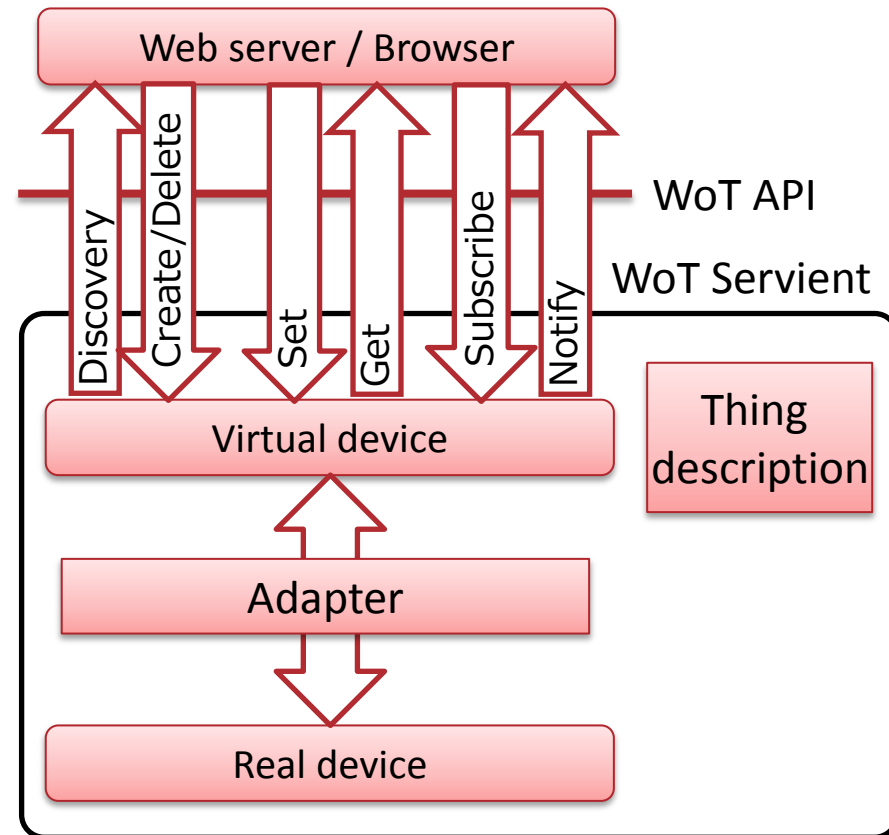
- Virtual device (VD)
- Real device
- Adapter
- Thing description

■ WoT Servient interface

- Discovery, Registration and De-Registration of VD
- Property access (set/get) to VD

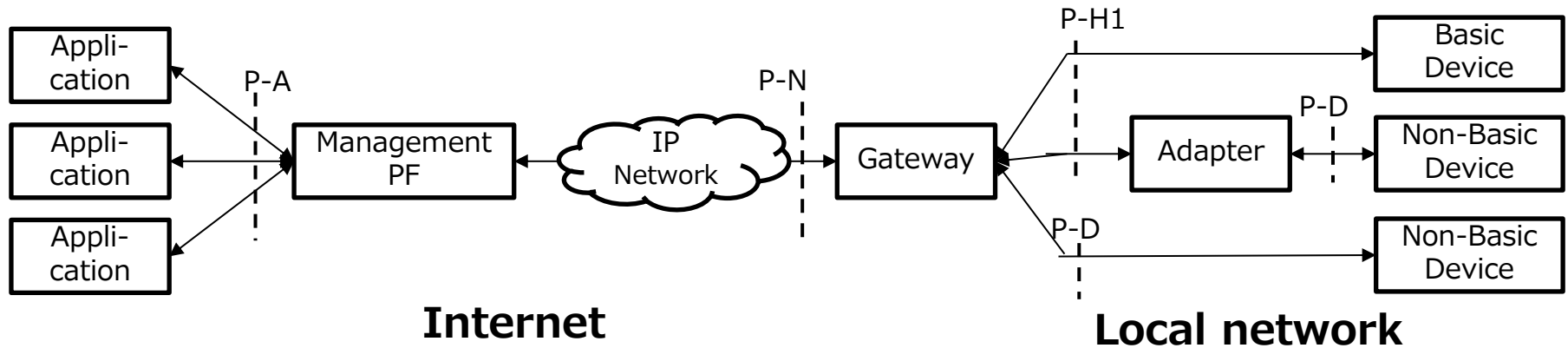
■ Adapter

- Adapter mediates the gap between virtual device and real device.



Legacy devices need adapters. In the case of WoT devices, Real devices themselves are Virtual devices.

■ ITU-T Y.2070 Architecture model

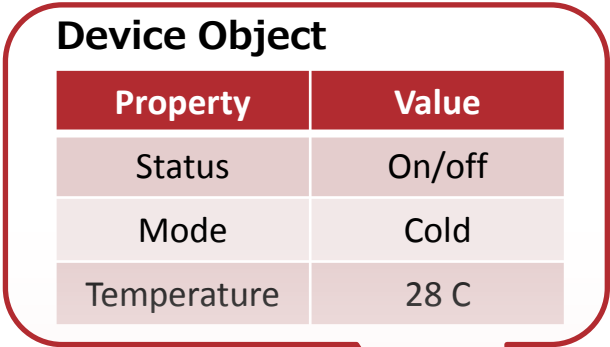


- Define how devices are connected to the IoT ecosystem
- Common APIs to diverse kinds of devices.
 - Sensors, Meters, Heater, Lights, TV, Washer/Dryer, etc.
- Facility to help identify the cause of troubles within local network.

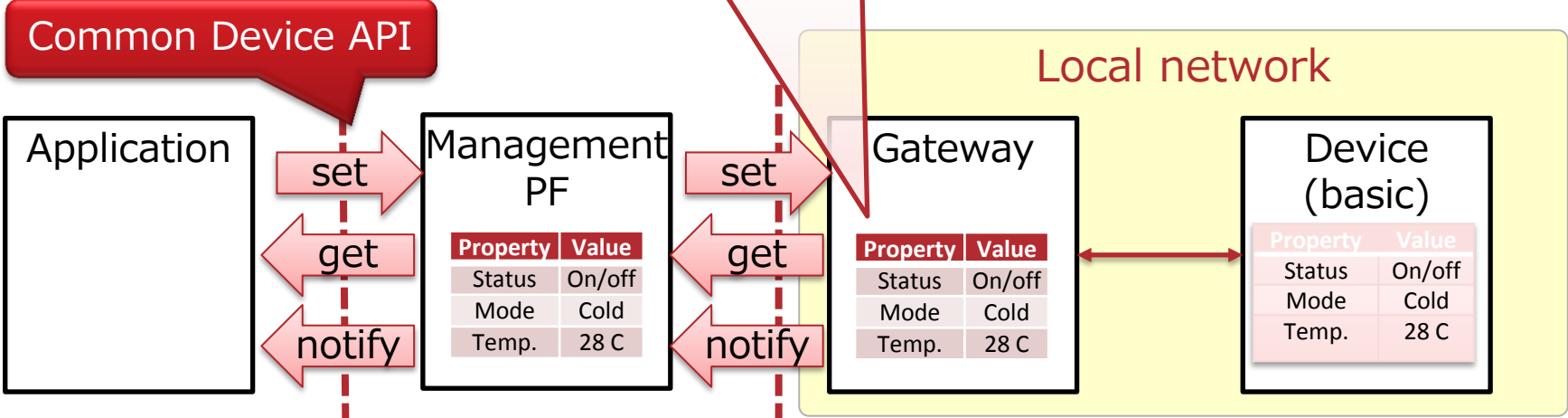
Y.2070 “Requirements and Architecture of Home energy management system and Home network services” was approved on Jan. 2015

■ Device Objects

- Representation of device functions and states.
- Set of properties and their values.
- Properties are defined for each device kind by respective industry or by platform vendors.

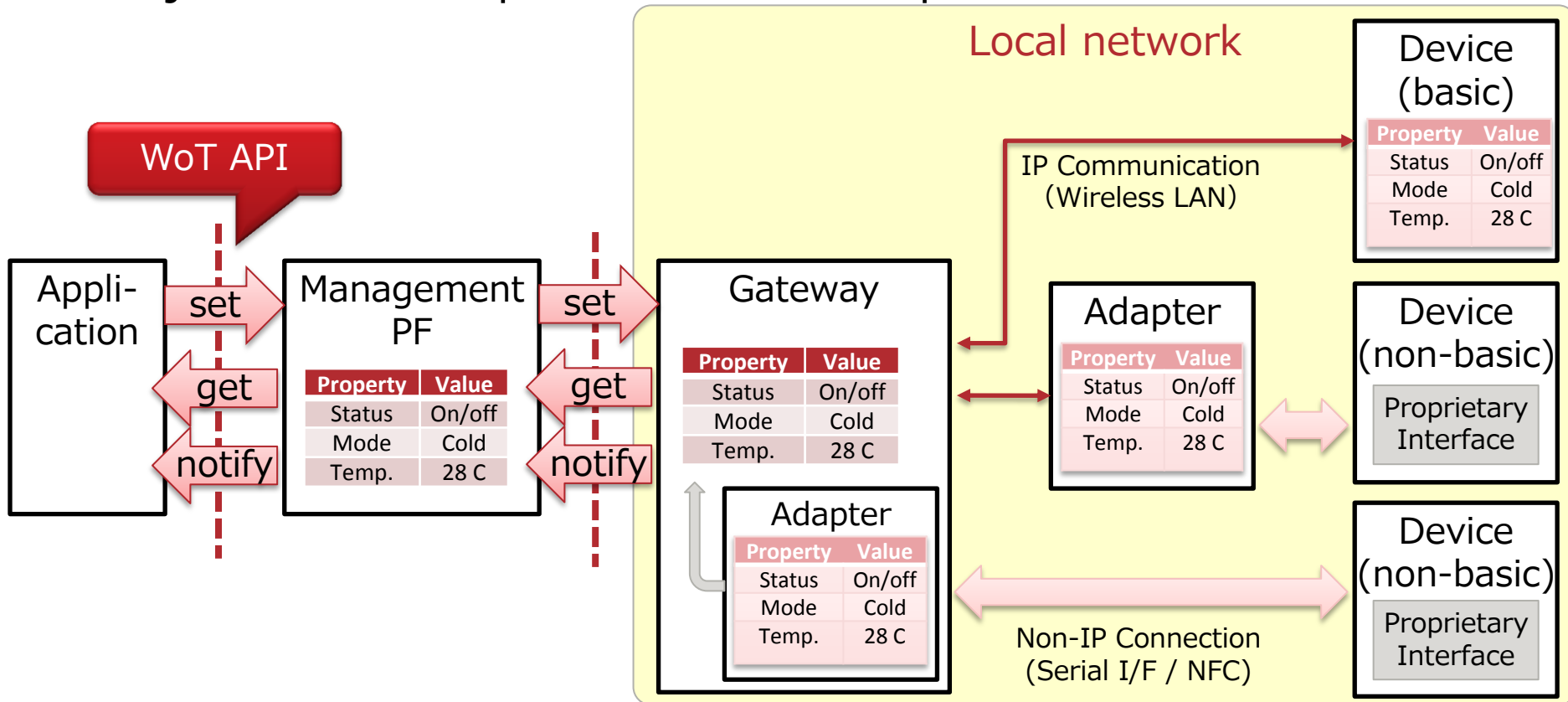


Device object is supported by standard for home automation devices (e.g. ECHONET Lite, KNX, and SEP)



Device adaptation – ITU-T Y.2070

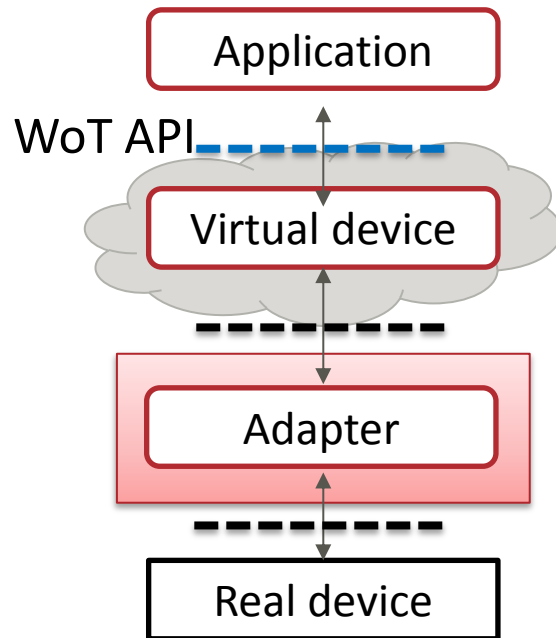
- Adapter devices, or adapter embedded in gateway
- Interface conversion between proprietary interface and function objects can be implemented in microprocessors.



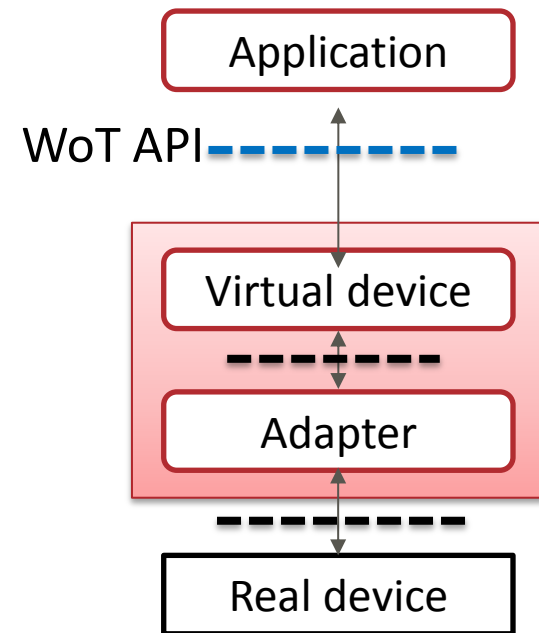
Basic devices themselves contain TD-like descriptions. Manipulation on Device objects in Management PF through Common device API results in operations on Real devices. Adapters implements TD-like description in the case of Non-basic devices.

- Function layout observation based on Kajimoto-san's four implementation models
 - Cloud centric and Hub Centric implementation models are defined in Y.2070.

Cloud centric

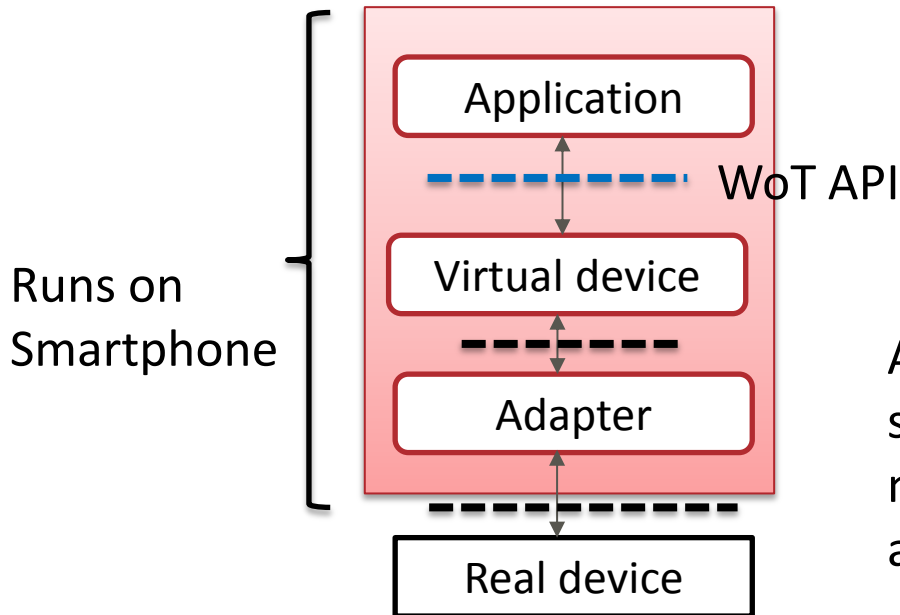


Hub Centric

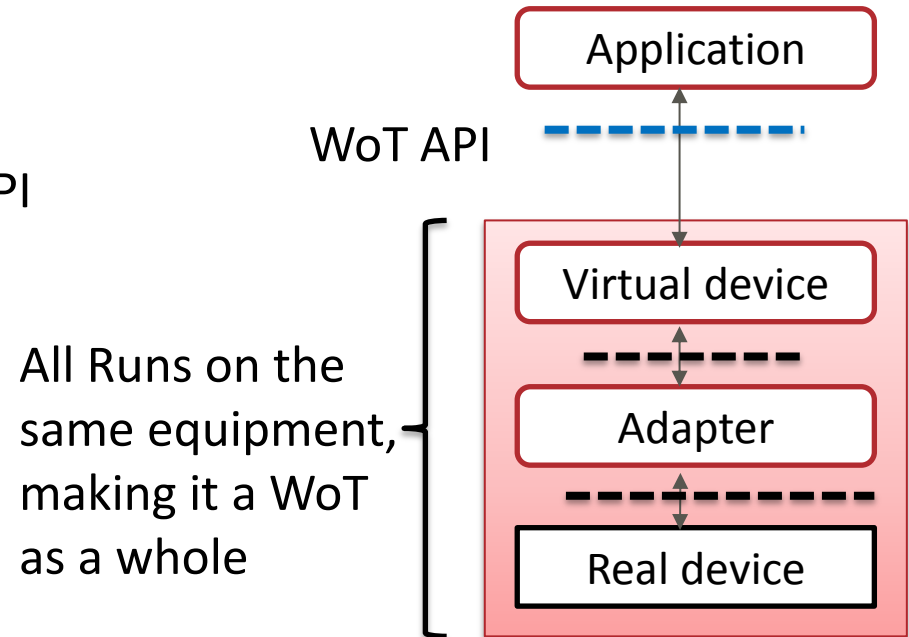


■ Observing Smartphone Centric, Web Centric models

Smartphone Centric



Web Centric



ECHONET Lite's Thing description (example)

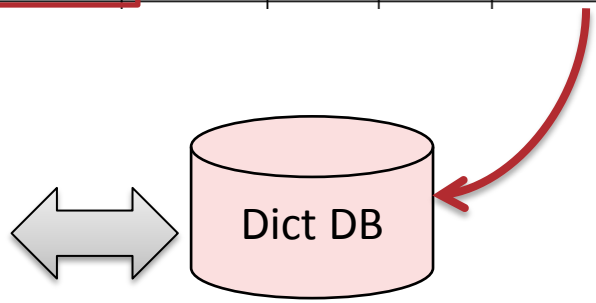
- Property is identified by its property name or EPC.
- Value holds property's contents.

Device Object Super-Class Definition (snippet)

APPENDIX Detailed Requirements for ECHONET Device objects, Release F

Property name	E P C	Contents of property		Data size	Data size (Byte)	Access rule	Mandatory Note2	Announcement at status change	Remark
		Value range (decimal notation)							
Operation status	0x80	This property indicates the ON/OFF status.	ON=0x30, OFF=0x31	unsigned char	1	Set		○	
						Get			
Installation location	0x81	This property indicates the installation location	See "2.2 'Installation location' property."	unsigned char	1 or 17	Set/Get	○ Note4	○	

property	value
status	On/off
mode	cooling
temperature	28 degree



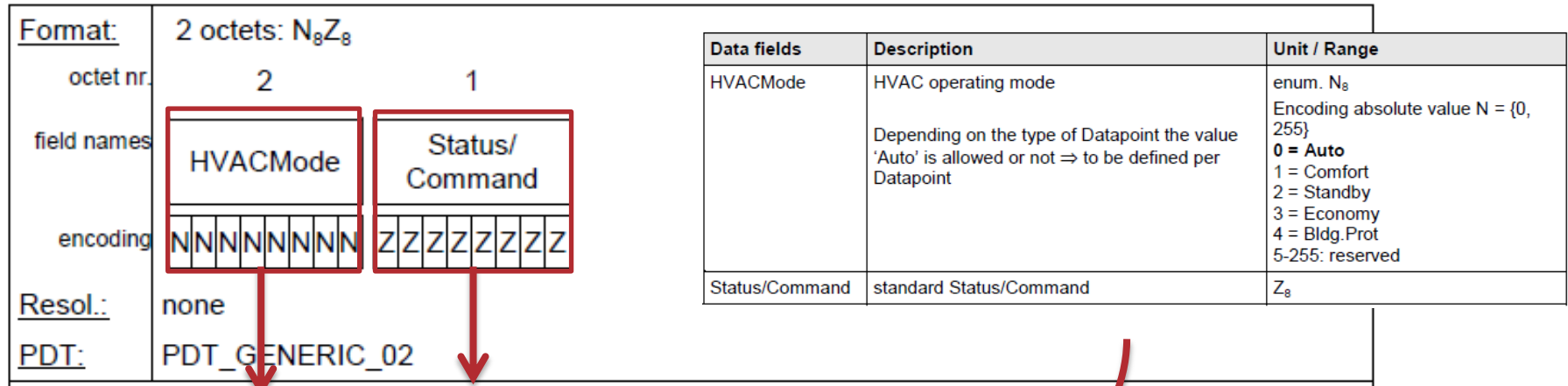
Dict DB manages each property's datatype. GW/Mgmt PF conduct type validation.

KNX's Thing description (example)

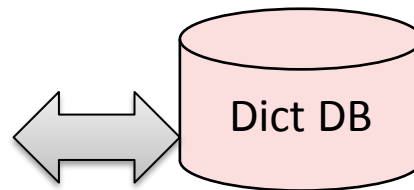
- Property is identified by its Datapoint Type.
- Value holds property's contents.

4.9.1 Datapoint Type "HVAC Operating Mode"

LTE: compound structure



property	value
status	On/off
mode	cooling
temperature	28 degree



Dict DB manages each property's datatype. GW/Mgmt PF conduct type validation.

■ Interaction type

- Property is the only construct necessary for ECHONET Lite and KNX
- Distinction read-only items from writable items is useful. Read-only items are typically ones for reporting in nature such as sensor values.
- Other constructs are for convenience, not based on real device's interface. For example, events keep applications from polling by notifying them instead (e.g. invoke Call back functions).

■ TD document


- TD should be defined in a manner each standard (e.g. ECHONET Lite, KNX) can define its mapping to TD.
- Virtually all items defined in ECHONET Lite and KNX can be mapped to TD.

■ Architecture model document

- This document should cover the points described in preceding slides. Of particular interests are the followings.
- Describe in detail about the relationship between WoT server, Virtual device, Real device
- Describe the four implementation models
- Describe requirements to TD based on architectural discussion, using ECHONET Lite and KNX as concrete cases.

■ Fujitsu demo

- A system based on Cloud Centric model. Applications on the cloud controls appliances (in this demo, they are lights.)
- TD are defined by mapping from ECHONET Lite.
- WoT-API is based on REST + XML(TD)



FUJITSU

shaping tomorrow with you