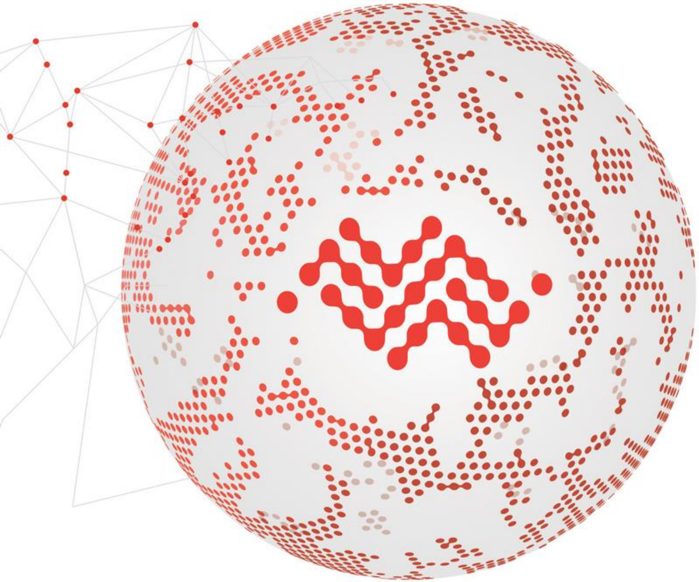
Hands-on Lightweight M2M



Sierra Wireless is building the Internet of Things.

SIERRA WIRELESS®

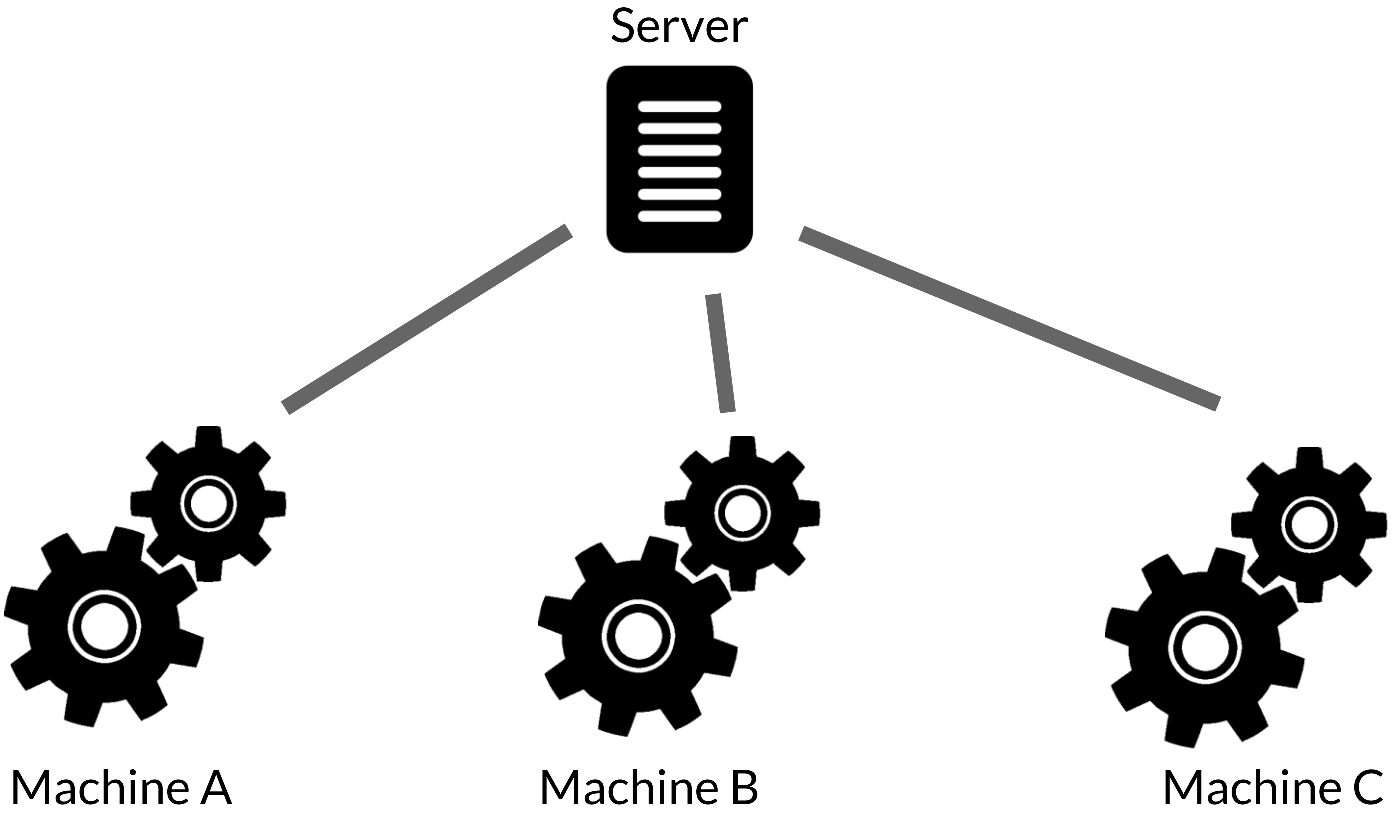
Agenda

- From M2M to Web-of -Things
- Device management 101
- Intro to CoAP
- Intro to Lightweight M2M
- Security with LwM2M
- Secrets & Access control
- Get started with Leshan & Wakaama

From M2M to Web-of-Things

SIERRA WIRELESS®

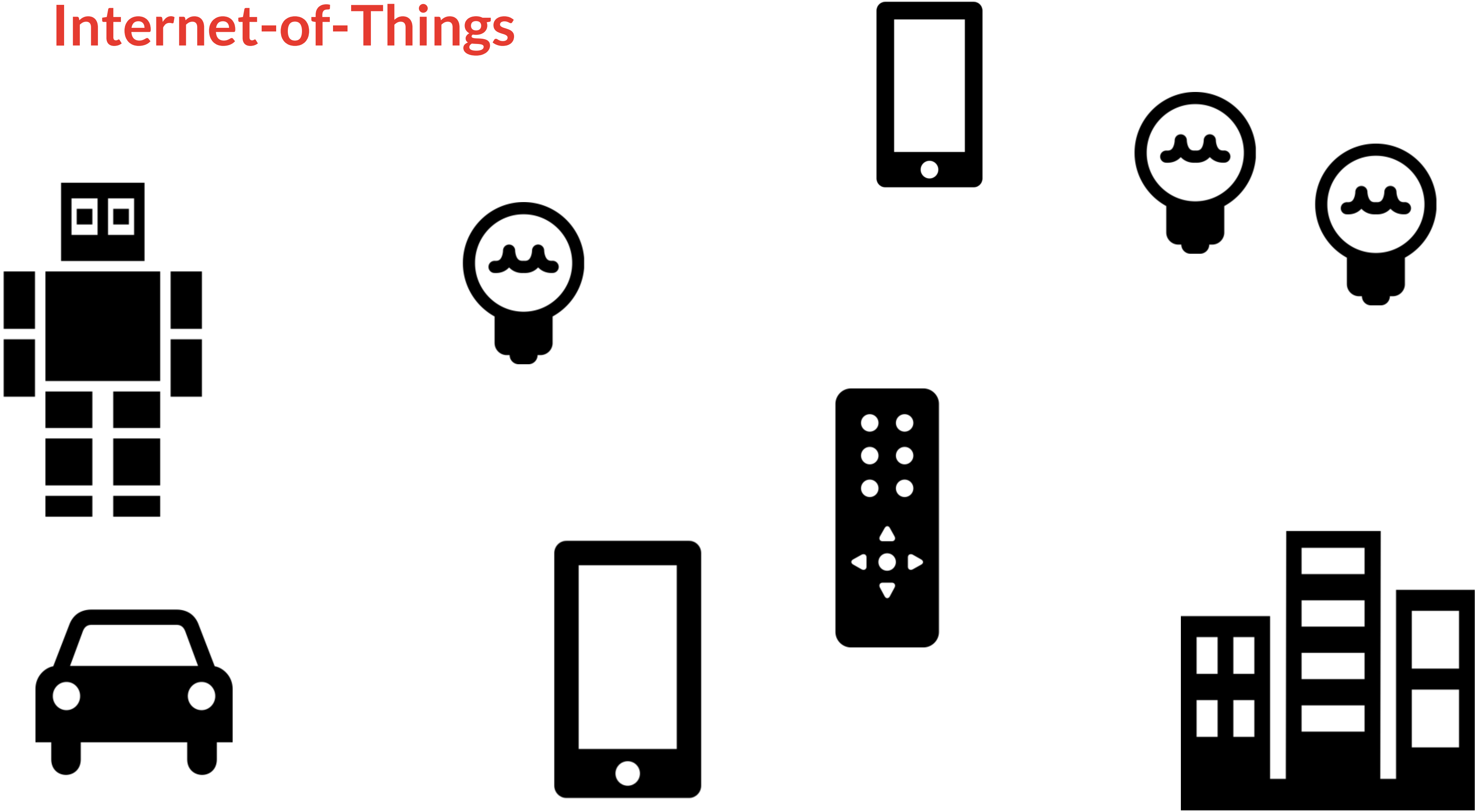
Machine-to-Machine



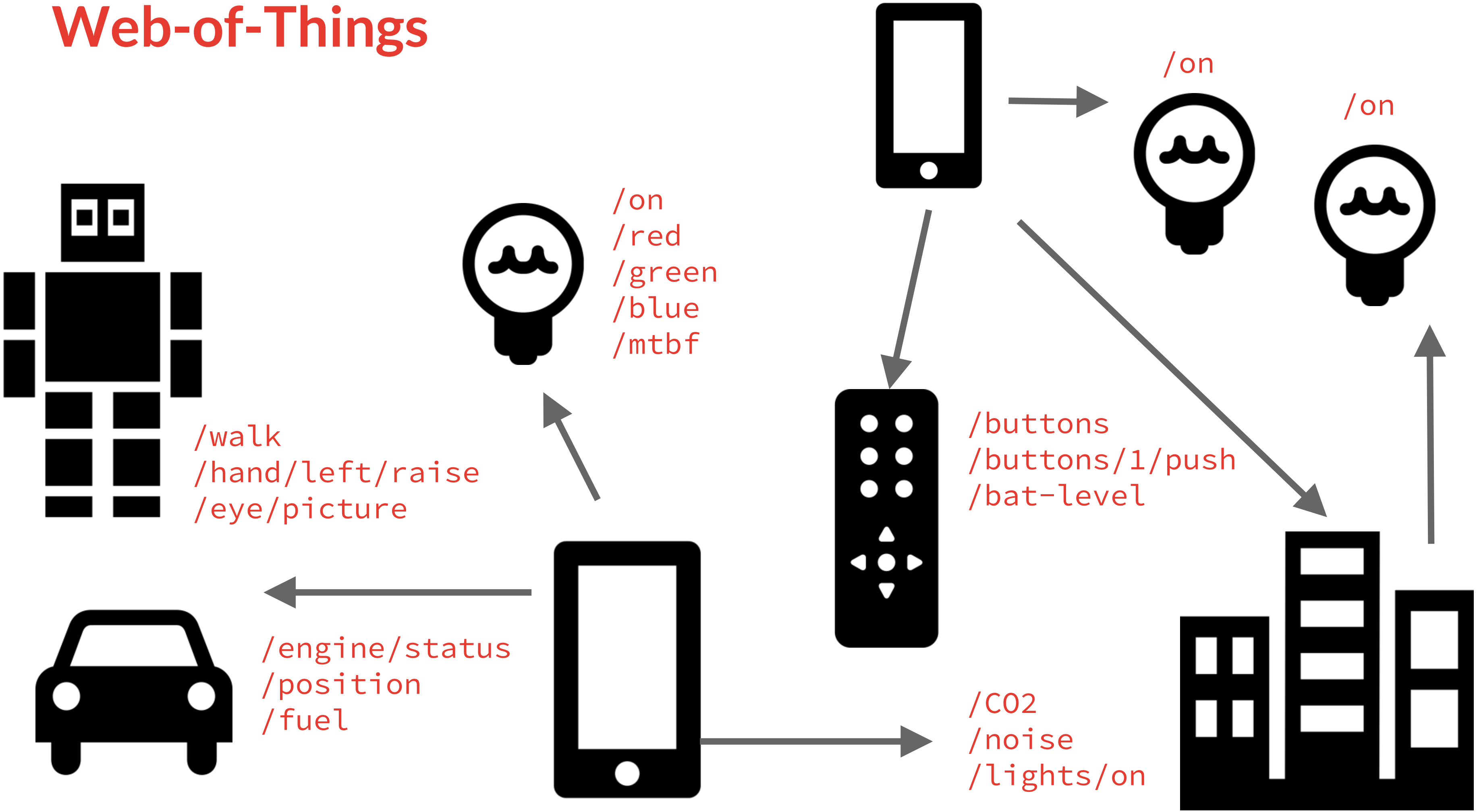
Conquering the last mile

- Low power networks plugged to the Internet
- 6LowPAN
- Bluetooth Smart 4.2
- Thread
- LWPA (LoraWAN, LTE-MTC,...)
- IPv6 MTU: 1280 bytes, 6LowPAN: ~100 bytes
- TCP, HTTP,MQTT doesn’t fit

Internet-of-Things



Web-of-Things

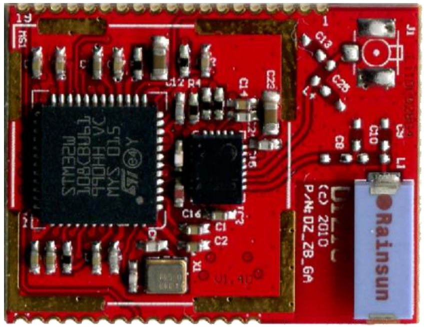


CoAP

Constrained Application Protocol

CoAP: A new protocol for the IoT

Class 1 devices
~100KiB Flash
~10KiB RAM
~$1



Low-power networks
<100Bytes packets

CoAP in a nutshell

RFC 7252: Constrained Application Protocol

- RESTful protocol designed from scratch
    - URIs, Internet Media Types
    - GET, POST, PUT, DELETE
- Transparent mapping to HTTP
- Additional features for M2M scenarios
    - Observe

CoAP: Constrained Application Protocol

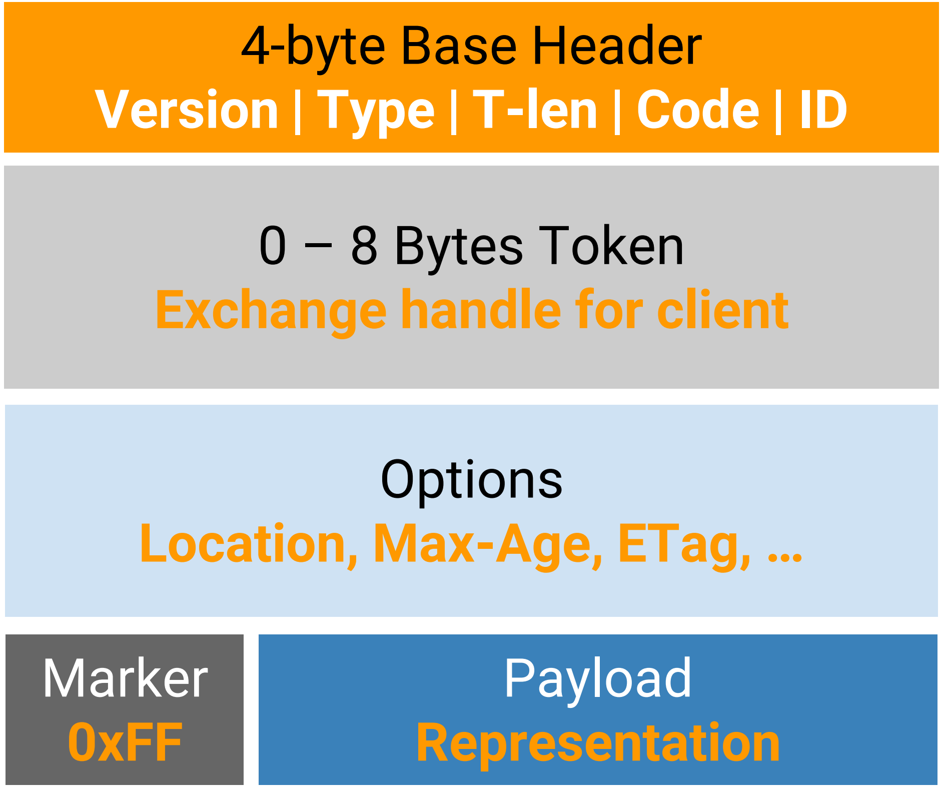
Binary protocol

- Low parsing complexity

- Small message size

Options

- Binary HTTP-like headers



4-byte Base Header

Version | Type | T-len | Code | ID

0 – 8 Bytes Token

Exchange handle for client

Options

Location, Max-Age, ETag, …

Marker

0xFF

Payload

Representation

Device Management

Operate, monitor, upgrade fleets

SIERRA WIRELESS®

Device Management

Secure, monitor, manage a fleet of devices

Configure the device

Update the firmware (and maybe the app)

Monitor and gather connectivity statistics

Device Management

You don't know yet what hardware will power your IoT projects on the field,

But you MUST be able to do device management in a consistent way without vendor lock

OMA Lightweight M2M

An API on top of CoAP

SIERRA WIRELESS®

Lightweight M2M

REST API for:

Security provisioning

Connectivity configuration, monitoring, statistics

Location

Firmware Upgrade

Software management

Error reporting

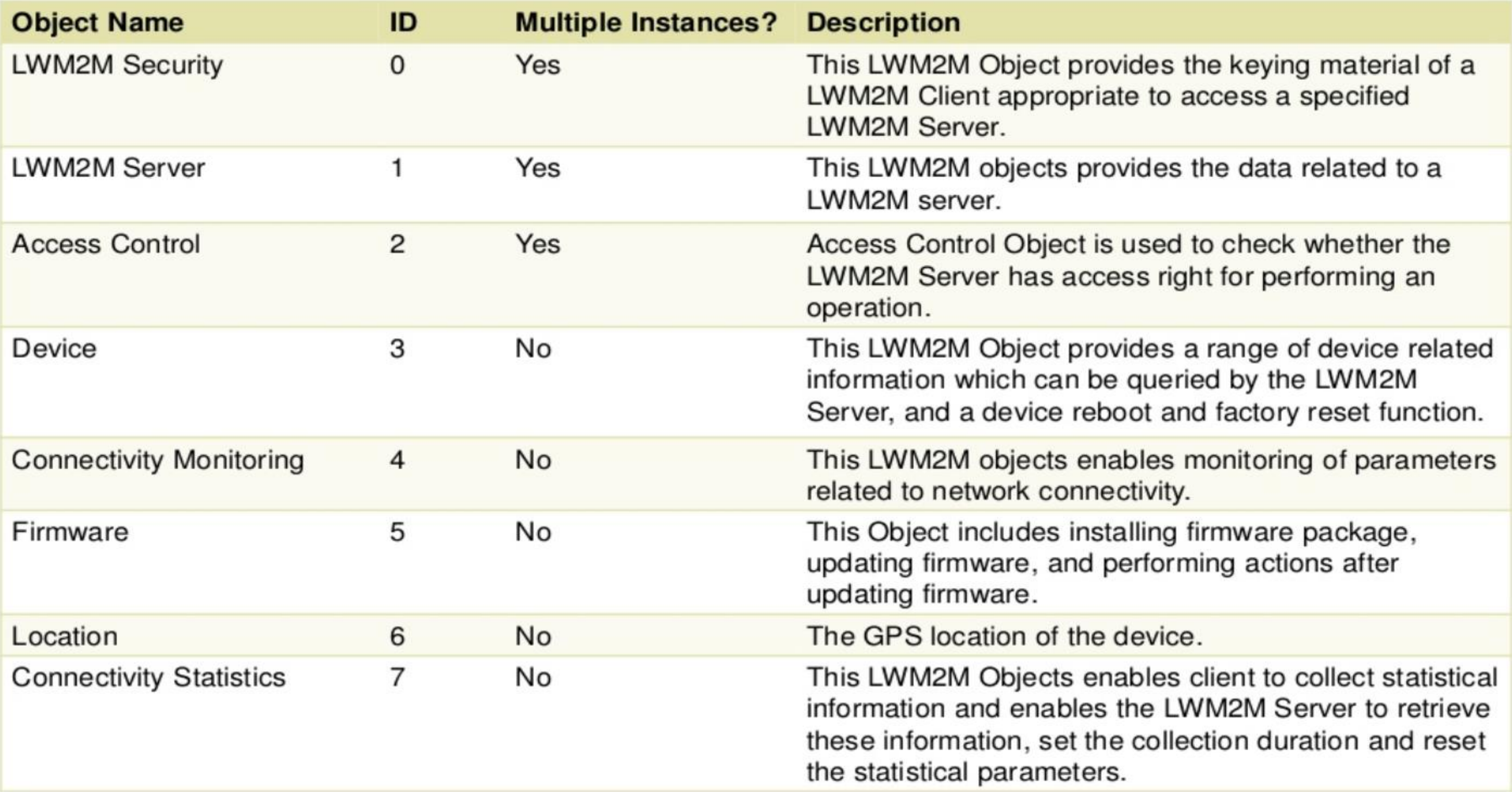
LwM2M API URLs

/{object}/{instance}/{resource}

Examples:

"/6/0" the whole location object (binary record)

"/6/0/1" only the longitude (degree)



Object Name	ID	Multiple Instances?	Description
LWM2M Security	0	Yes	This LWM2M Object provides the keying material of a LWM2M Client appropriate to access a specified LWM2M Server.
LWM2M Server	1	Yes	This LWM2M objects provides the data related to a LWM2M server.
Access Control	2	Yes	Access Control Object is used to check whether the LWM2M Server has access right for performing an operation.
Device	3	No	This LWM2M Object provides a range of device related information which can be queried by the LWM2M Server, and a device reboot and factory reset function.
Connectivity Monitoring	4	No	This LWM2M objects enables monitoring of parameters related to network connectivity.
Firmware	5	No	This Object includes installing firmware package, updating firmware, and performing actions after updating firmware.
Location	6	No	The GPS location of the device.
Connectivity Statistics	7	No	This LWM2M Objects enables client to collect statistical information and enables the LWM2M Server to retrieve these information, set the collection duration and reset the statistical parameters.

Standard objects

Example: Object Device

Manufacturer

Model number

Serial number

Firmware version

Reboot

Factory reset

Power sources

Power V/A

Battery level

Memory free

Error code

Current time

UTC offset

Timezone

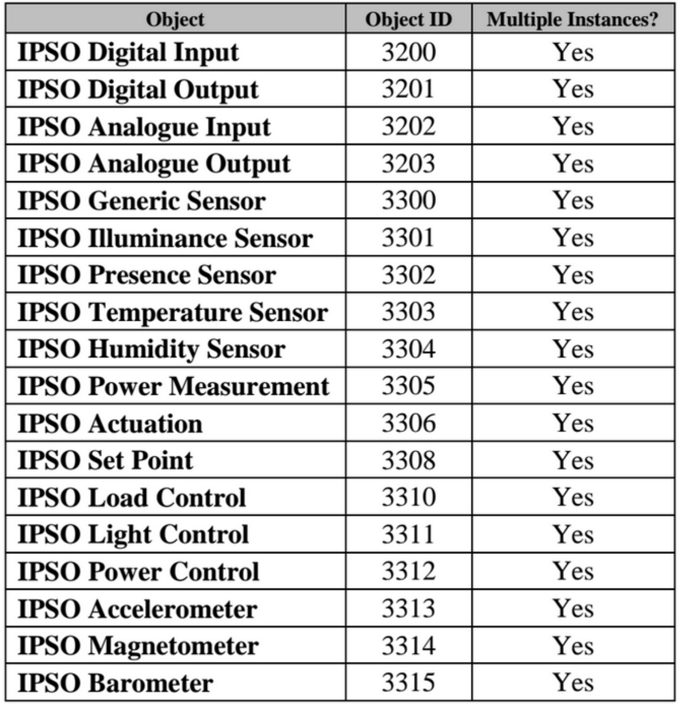
Custom objects

You can define your own objects and register with the OMA

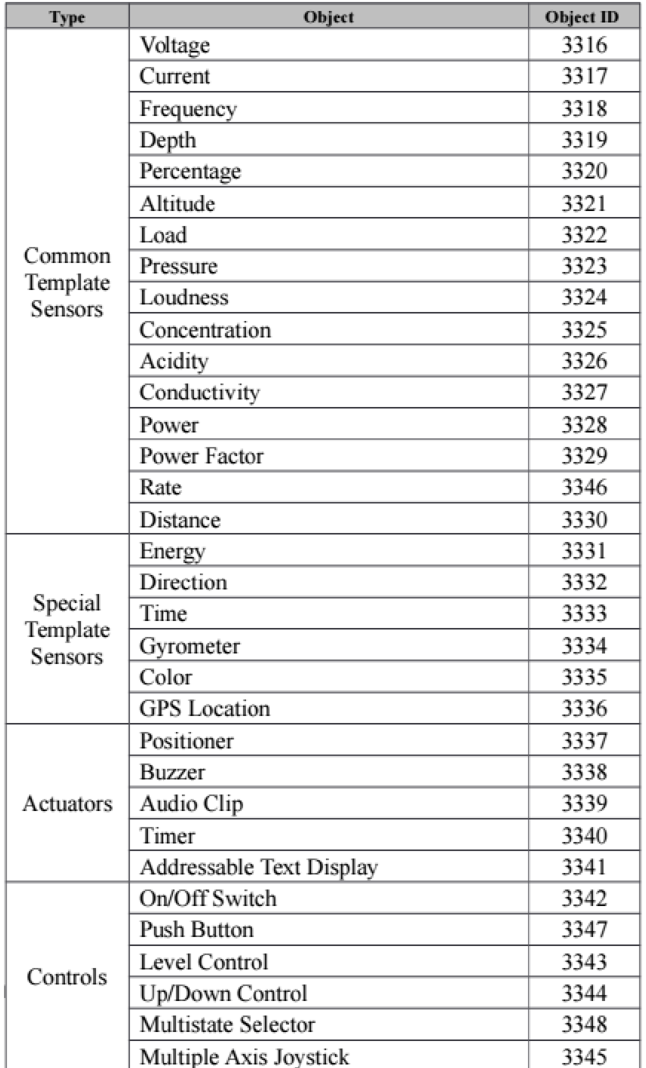
Discoverable using CoAP Link Format

IPSO Alliance Smart Objects:

- accelerometer, temperature, sensors,...



Object	Object ID	Multiple Instances?
IPSO Digital Input	3200	Yes
IPSO Digital Output	3201	Yes
IPSO Analogue Input	3202	Yes
IPSO Analogue Output	3203	Yes
IPSO Generic Sensor	3300	Yes
IPSO Illuminance Sensor	3301	Yes
IPSO Presence Sensor	3302	Yes
IPSO Temperature Sensor	3303	Yes
IPSO Humidity Sensor	3304	Yes
IPSO Power Measurement	3305	Yes
IPSO Actuation	3306	Yes
IPSO Set Point	3308	Yes
IPSO Load Control	3310	Yes
IPSO Light Control	3311	Yes
IPSO Power Control	3312	Yes
IPSO Accelerometer	3313	Yes
IPSO Magnetometer	3314	Yes
IPSO Barometer	3315	Yes



Type	Object	Object ID
Common Template Sensors	Voltage	3316
	Current	3317
	Frequency	3318
	Depth	3319
	Percentage	3320
	Altitude	3321
	Load	3322
	Pressure	3323
	Loudness	3324
	Concentration	3325
	Acidity	3326
	Conductivity	3327
	Power	3328
	Power Factor	3329
	Rate	3346
	Distance	3330
Special Template Sensors	Energy	3331
	Direction	3332
	Time	3333
	Gyrometer	3334
	Color	3335
	GPS Location	3336
Actuators	Positioner	3337
	Buzzer	3338
	Audio Clip	3339
	Timer	3340
	Addressable Text Display	3341
Controls	On/Off Switch	3342
	Push Button	3347
	Level Control	3343
	Up/Down Control	3344
	Multistate Selector	3348
	Multiple Axis Joystick	3345

Security with Lightweight M2M

DTLS and secret management

Authentication and encryption

Based on DTLS 1.2 (TLS for Datagrams)

Focus on AES & Elliptic Curve Cryptography (ECC)

AES Hardware acceleration in IoT oriented SoC

Works on Low Power networks (~100bytes MTU)

TLS_PSK_WITH_AES_128_CCM_8

Pre-Shared-Key:

password for session authentication

AES 128bits (or 256) - Counter CBC Mode:

encryption and integrity (AEAD cipher)

8 bytes for integrity in place of CCM usual 16

What? :)

PSK: No certificates, just password

CCM8: compactness

Full DTLS-PSK-CCM8 handshake in ~1030 bytes

Ex: HTTPS TLS handshake ~6000bytes

More security: TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8

ECDHE: Perfect Forward Secrecy (PFS)

Someone rob your private key: he can’t decrypt past communications

ECDSA: use public key in place of password

You can use X.509 certificates (like HTTPS)

At scale?

You will have a fleet of device

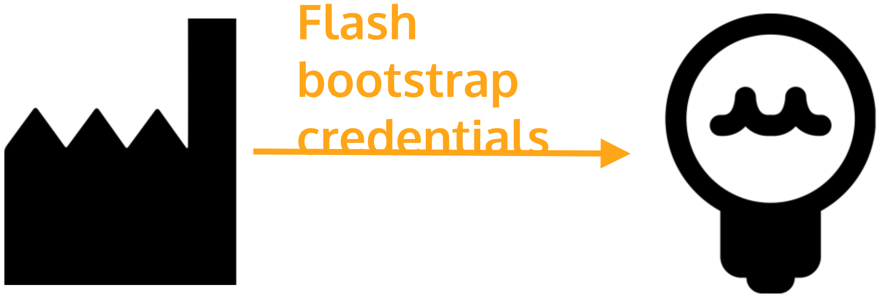
They need secrets (key, password, etc..)

Unique across devices

You need to be able to change those secrets

You will probably don’t trust your factory

Lightweight M2M Bootstrap

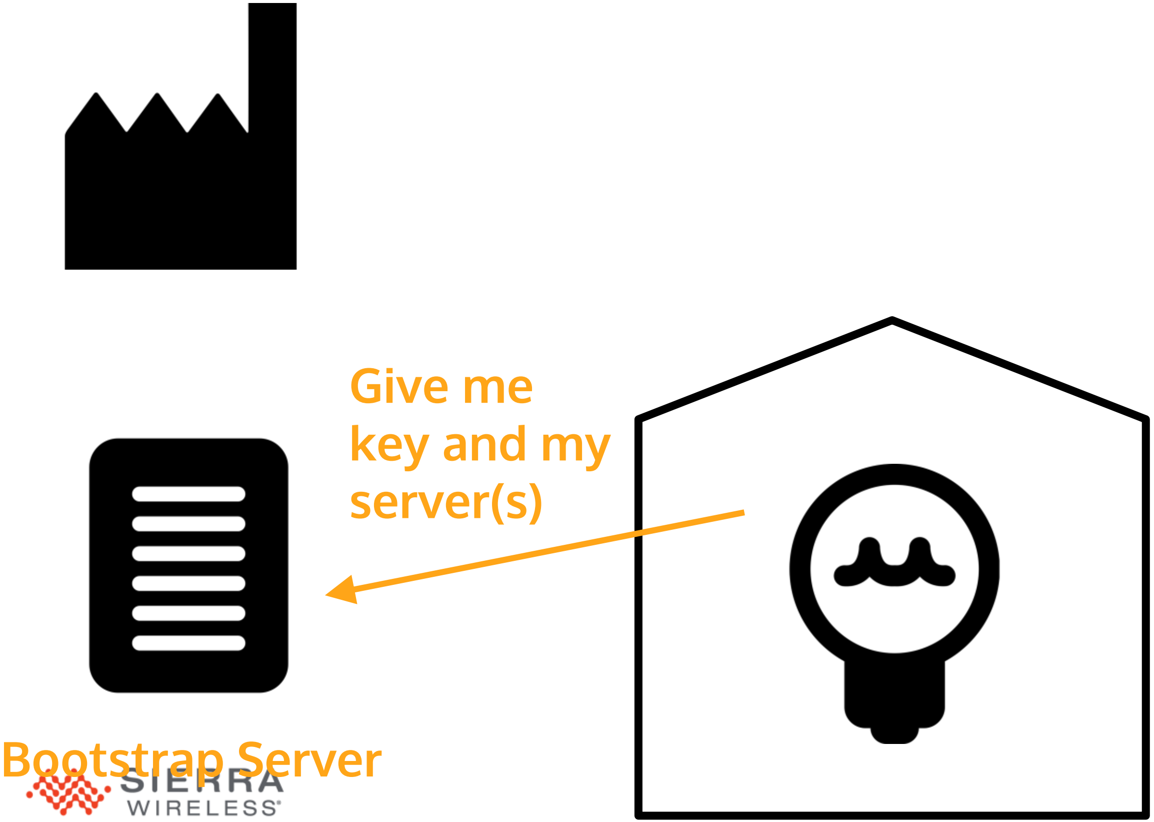


Flash bootstrap credentials

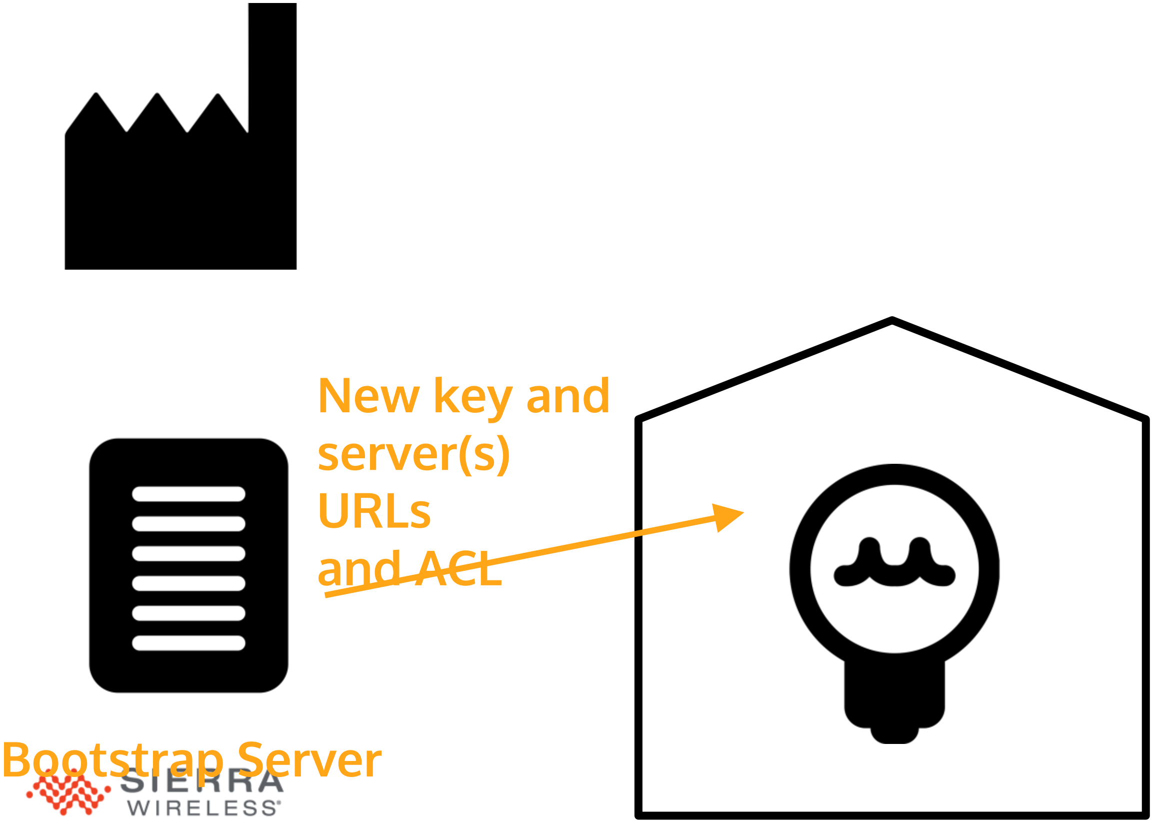
Lightweight M2M Bootstrap

I only have bootstrap credentials or I can’t reach final server

Lightweight M2M Bootstrap



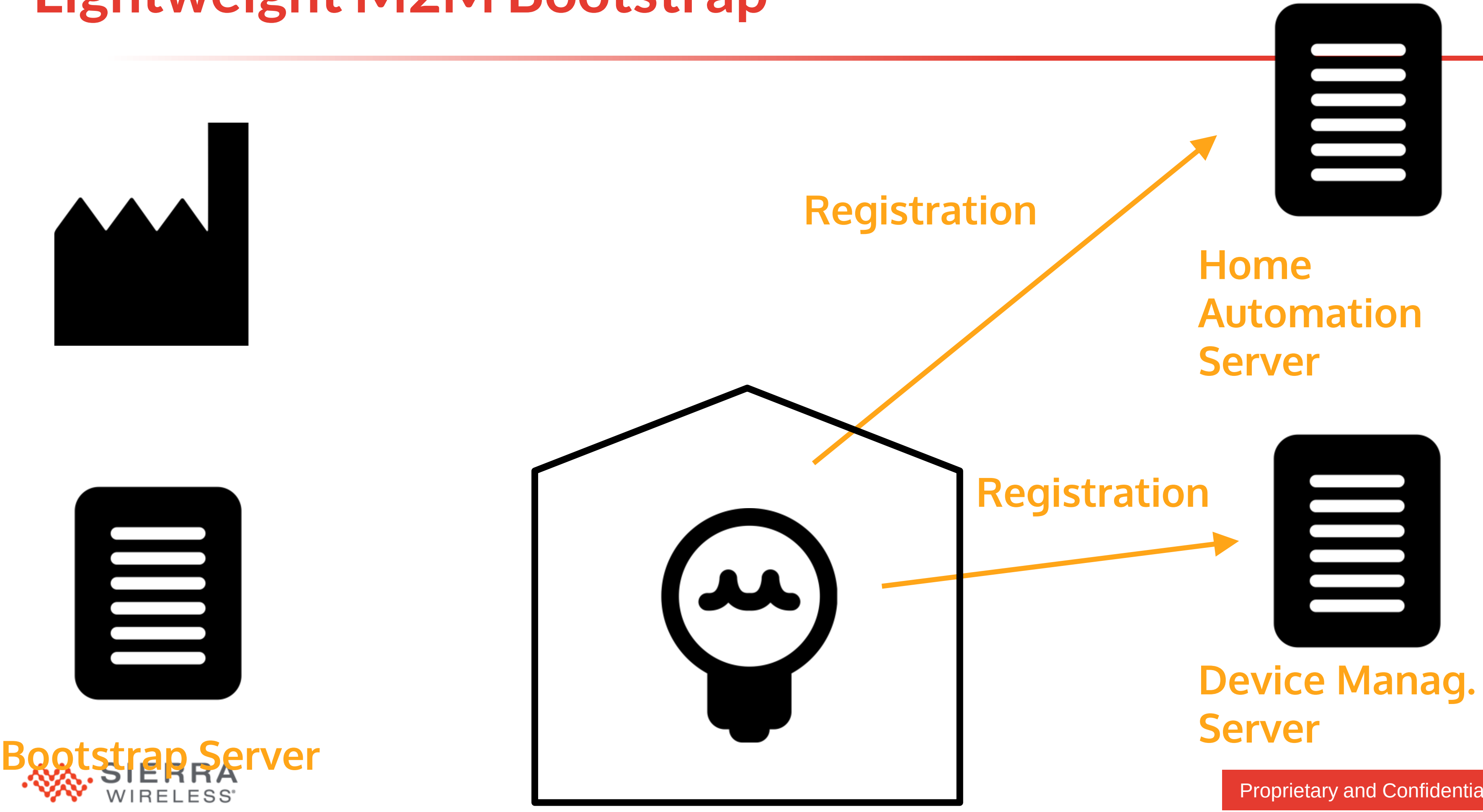
Lightweight M2M Bootstrap



New key and server(s) URLs and ACL

Bootstrap Server

Lightweight M2M Bootstrap



ACL: Access Control Lists

Define which operation on a given object for a given server

One server for Over-The-Air upgrade:

“/5/”+“/9/” read, write, exec

One server for application, maybe with:

“/5” read only

Hands-On!

Getting started with Leshan & Wakaama

SIERRA WIRELESS®

Leshan

Java library for implementing servers & clients

Friendly for any Java developer

Simple (no framework, few dependencies)

But also a Web UI for discovering and testing

Build using “mvn install”

Based on Californium and Scandium

http://eclipse.org/leshan

Public sandbox

http://leshan.eclipse.org

Bleeding edge: deployed on master commit

IPv4 and IPv6

Press “CoAP messages” for low-level traces

Wakaama

A C client and server implementation of LwM2M

Not a shared library (.so/.dll)

Embedded friendly but using malloc/free

Plug your own IP stack and DTLS implementation

http://eclipse.org/wakaama

http://github.com/eclipse/wakaama

Wakaama features

Register, registration update, deregister

Read, write resources

Read, write, create, delete object instances

TLV or plain text

Observe

Tinydtls

Eclipse Proposal

“Support session multiplexing in single-threaded applications and thus targets specifically on embedded systems.”

Examples for Linux, or Contiki OS

TLS_PSK_WITH_AES_128_CCM_8

TLS_ECDHE_ECDSA_WITH_AES128_CCM_8

http://sf.net/tinydtls

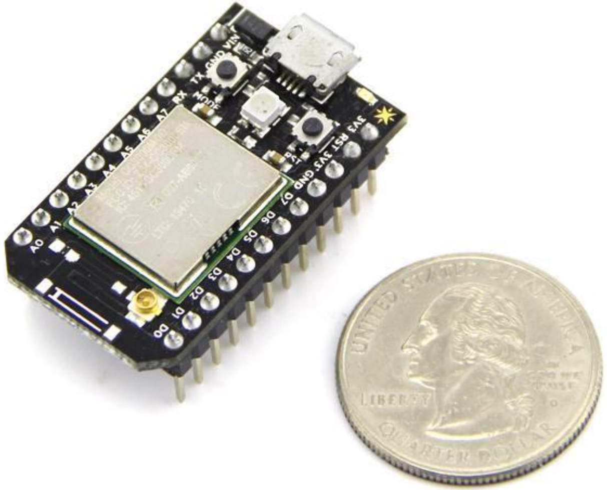
In real hardware?

Spark Core:

Cortex-M3 STM32,

RAM/ROM 20/128k, 72MHz

WiFi



Arduino Mega

AVR, **ATmega2560**,

RAM/ROM 8/256k, 16MHz

Ethernet