

Open the Box

Open APIs for the Smart Home

Fred Rivard, PhD, IS2T
André Bottaro, PhD, Orange Labs

with the contribution of
François Bodet, Bouygues Telecom,
Jean Le Tutour, Delta Dore
and Serge Subiron, IJenko

Eclipse IoT Day,
St Martin D'Hères, February, 19th, 2014



Summary

- A **new world of applications** is about to emerge at home thanks to the growing variety of available sensors and actuators
- To unleash service delivery, the whole telecom infrastructure is to be **open to third party applications** through standard cloud and embedded APIs.
- Today technology status enables the openness to a set of trusted partners. **Remaining challenges are addressed by Open the Box project.**
- **Cost and reliability** matter. Open the Box specified an OSGi platform with a low hardware footprint and isolation between deployed applications.
- A **demonstration** shows application portability on IS2T Open the Box platform.

Smart Home

30 years after its official launch, a niche market likely to take off






- Seven application domains
 - Security, energy, comfort, health, wellness, multimedia content sharing, games
- Emergence of new products
 - Smart – Easy to install – Wireless
 - Smart Objects: Withings, NetAtmo, Goji, Kolibree, Nest, ...
 - Smart Appliances: LG ThinkQ, Miele@Home, Samsung, ..
 - Affordable self-install systems: Blyss, iControl, Ijenko, MyFox,, ...



A new world of applications will emerge on the long term from the variety of sensors, actuators, devices that become available

A business ecosystem showing many initiatives

Many service providers and solution integrators initiatives...

 Energy efficiency	 Security & energy monitoring	 energy efficiency	 Deutsche Telekom, E.ON, e-Q3, Miele	 Offer with iJenko  Smart Home by Orange	 Energy@Home with Electrolux, Enel, Indesit	 energy monitoring with remote on/off switch	 Digital Life  Miruene	 demand response	 Android@Home ?  HomeOS ? 
--	---	--	--	---	---	--	---	--	---

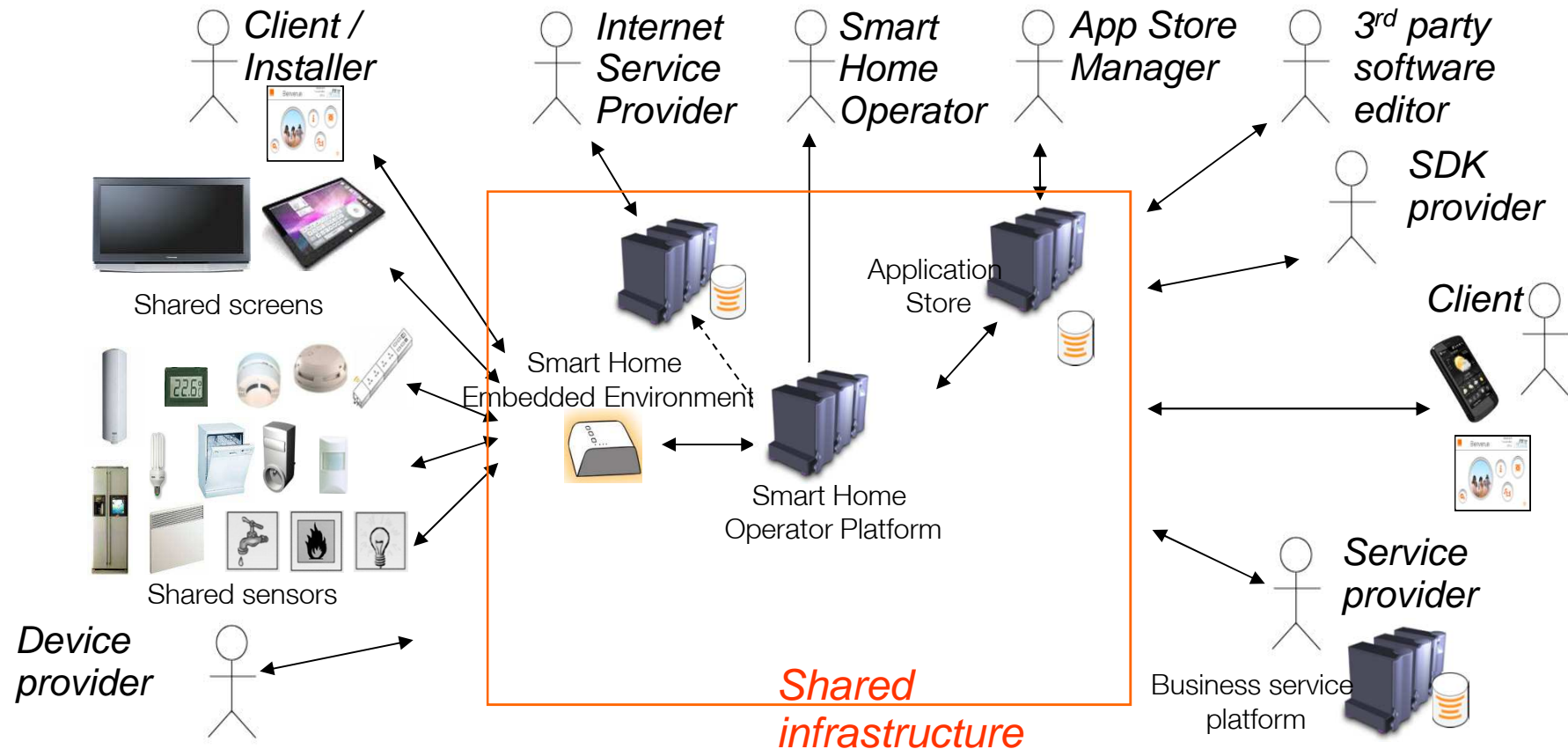
... and business actors that play both as service providers and product manufacturers



- The Smart Home Market is fragmented into niche markets
 - Proprietary protocols, APIs and solutions
 - Expensive solutions with a professional installation
 - Numerous partnerships on vertical applications
- ⇒ Many business initiatives arise
- ⇒ The ecosystem needs federation

The future ecosystem of the Smart Home market

Open the Box project use cases study



- Separated roles to let a wide set of business models emerge
 - Internet Service Provider, Smart Home Operator and App Store Manager.
 - Device provider and service provider. No silo.
 - SDK provider and 3rd party software editor

Technical challenges to open the architecture to 3rd party applications

Embedded service software platform

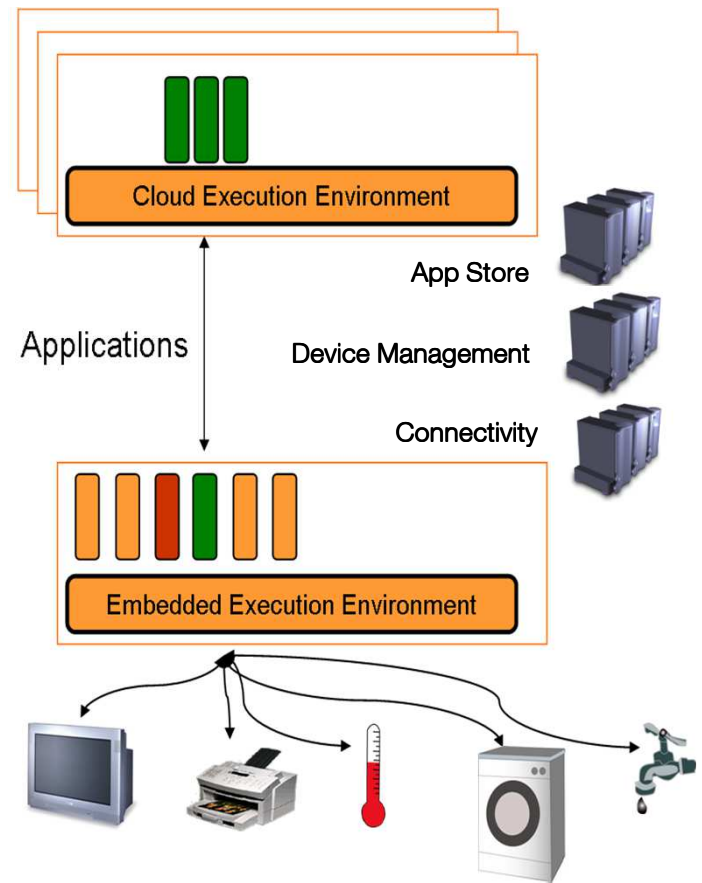
- Robustness, adaptation to constrained devices
- Resource management on a common embedded software platform
- Security – management of the access rights to applications, to hardware functions and to deployed sensors
- Device Representation Layer – A simple SDK to represent devices
- Dynamic application programming, data mediation infrastructure

Device and software management platform and application shops

- Openness of device management platforms and application shop tools to 3rd parties
- Sensor network and device management
- Modular application deployment and administration
- Simple remote access to heterogeneous device and sensor networks

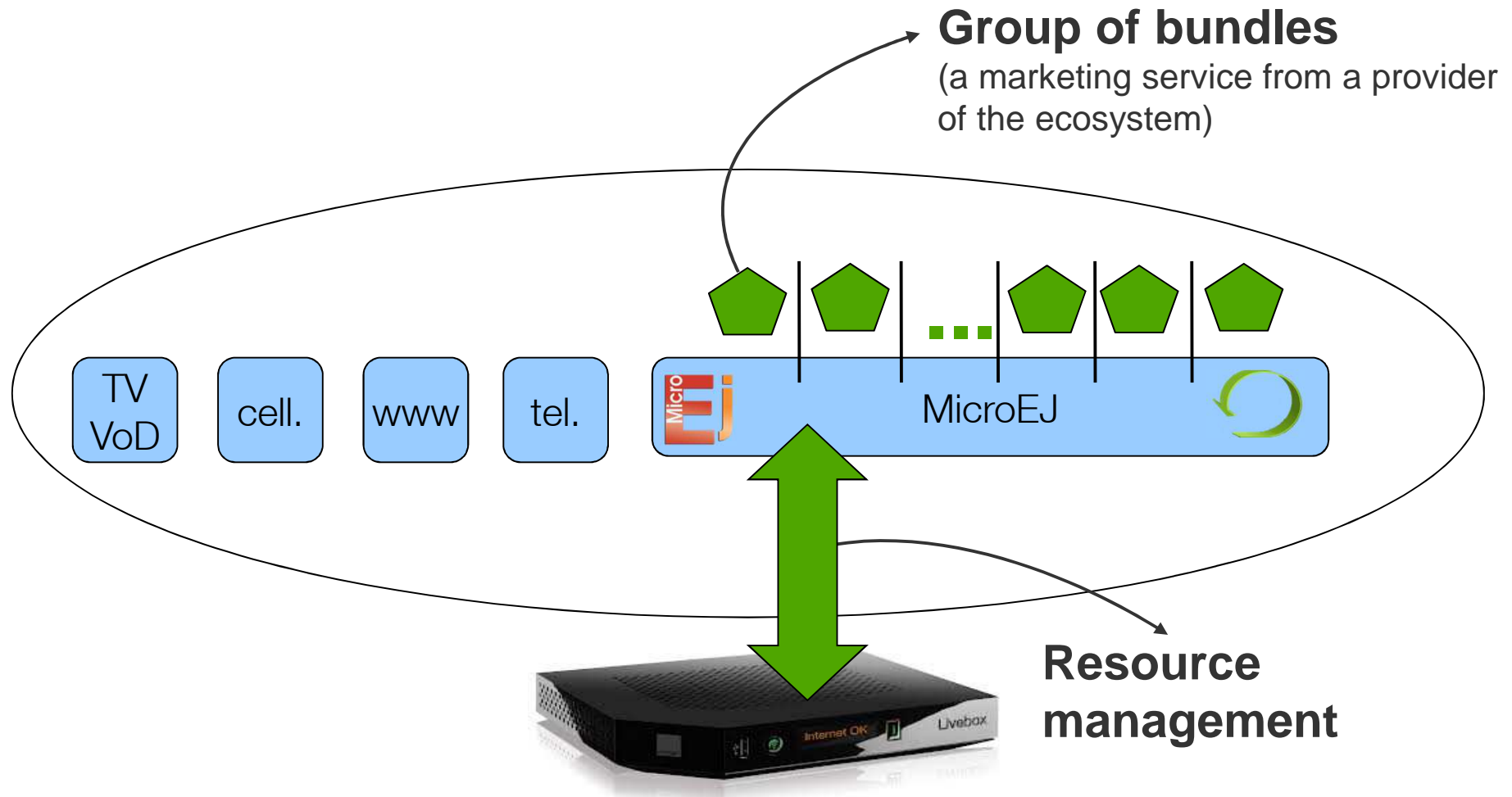
Hardware box platform and sensor networks

- Low power, home range, and low cost technology requirements
- Technology interoperability

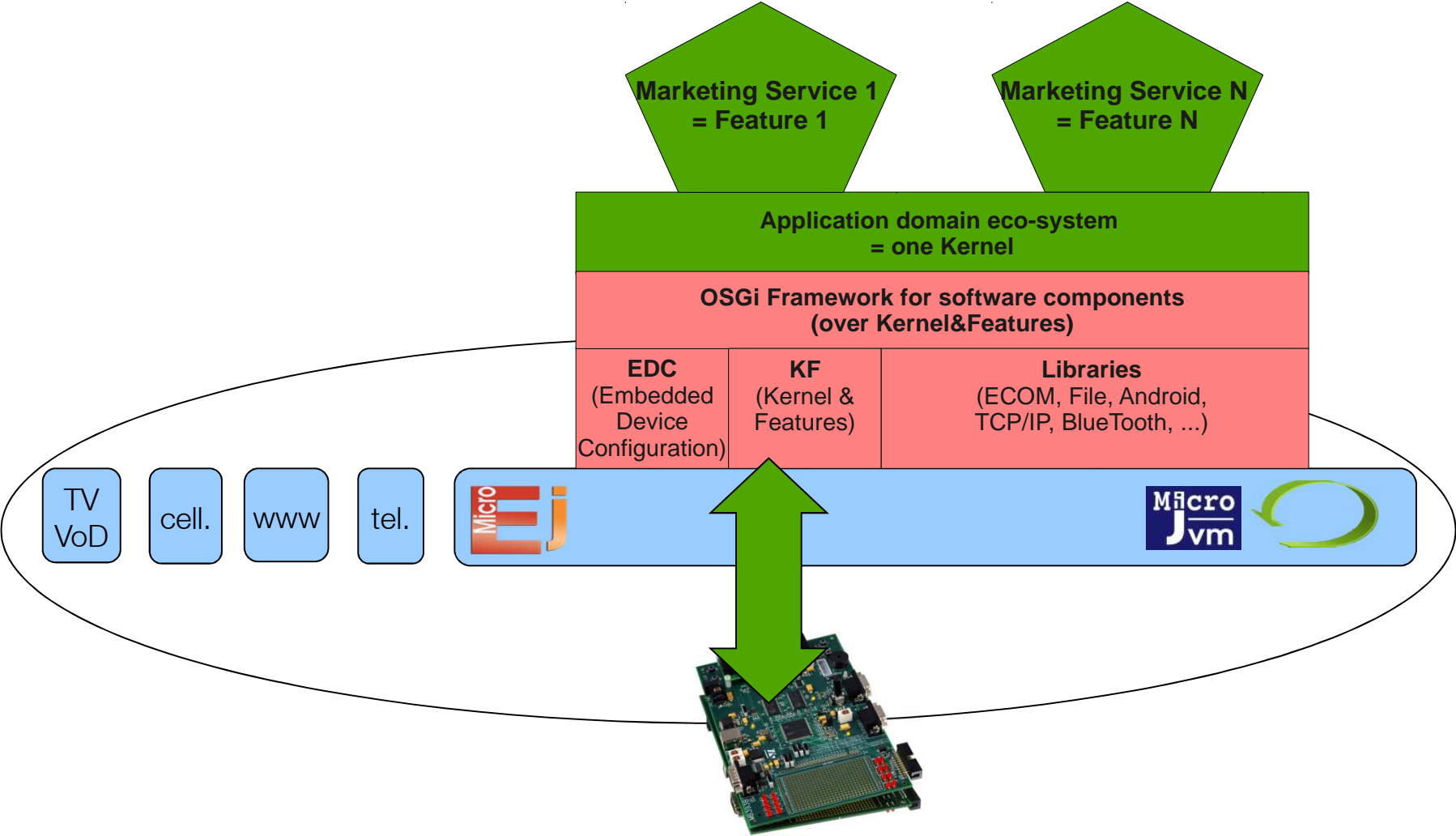


Open the Box embedded software platform

« Fifth Play » Smart Grid Ready Services



Embedded technical solution depicted



Embedded platform technical requirements

- **Low consumption & OS agnostic**
 - » 1 mega of RAM, 1 Mega of « flash » (code + JPF) : no bloatware !
 - » Must run THE SAME on any RTOS (VxWorks, Linux/Android, Win, RTX, uCOS, FreeRTOS, ...)
- **Resource management (OSGi RFC 200)**
 - » CPU, Memory, Storage, Input/Output streams
- **Reliable & Secure**
 - » Kill of a Feature (group of bundles) MUST be feasible at ANY time
 - Threads + objects + code killing
 - » No impact on other Features
 - No stale reference, no zombie threads, etc.
 - » No « back door »

Technical Solution

- **Principle : « group of bundles »**

- » One marketing service = one Feature = one tag for [Code, Threads, Objects]
- » 1 Kernel, N Features ==> ESR 020 « Kernel & Features »

- **Kernel**

- » Autonomous, does not rely on any feature
- » Controls the activity of the Features : life cycle, resources consumptions
- » Native code is permitted

- **Feature**

- » Relies only on the Kernel API
- » Cannot access to others features directly (code, objects, threads) : must go through kernel (proxy) object.
- » Fully controlled by the virtualization engine: no native code

Ownership

- **Type's owner**

- » Set when the type is loaded, cannot change during all type's life
- » Array of type: same owner as the type. Array of base types are from the Kernel.

```
ej.kf.Kernel.getOwner(Object)
```

- **Object's owner**

- » Set at object creation time to the execution context owner

- **Execution context's owner**

- » The first execution context owner is the thread object's owner.
- » Otherwise it is set to the caller's owner,
- » Except when the caller is executing in Kernel mode (execution context's owner is Kernel) and the receiver of the new context is an object owned by a feature (i.e. not the kernel), in that case the new execution context owner is set to the receiver's owner.
- » `Kernel.getContextOwner()` : fundamental API
- » `Kernel.enter()` : enter in Kernel mode
- » `Kernel.exit()` : if back to a Feature X, each local that points to an object owns by another Feature is set to null
- » Kernel APIs are only accessible from Kernel code, except `FeatureEntryPoint` interface (start, stop).

Features & Kernel definitions

- **Feature [K] = 1 [K].kf file**
 - » Entry point: implementation of the `ej.kf.FeatureEntryPoint`
 - » Version of the feature
 - » List of the embedded types (classes interfaces) : a.b.c, a.b.d.*
- **Identification : X509 certificate [F].cert**
 - » `ej.kf.Module.getProvider` returns the 6 first fields defined by RFC 2253:
CN (commonName), L(localityName), ST(stateOrProvinceName), O (organizationName), OU (organizationalUnitName), C (countryName).
- **Kernel**
 - » One `kernel.cert` X509 certificate
 - » One `kernel.kf` file (properties)
 - » By default, types are owned by the Kernel

Loading a Feature = a group of bundles

- **Kernel.load(InputStream)**

- » The content of the input stream is vm implementation dependent. IncompatibleFeatureException is thrown on error.
- » (1) read the bytes, link then
- » (2) create a new thread owned by the loaded Feature
- » (3) load returns

- **Asynchronous initialization inside the new thread**

- » Feature's all `<clinit>` are executed
- » Creation of the `ej.kf.FeatureEntrypoint` object
- » `FeatureEntryPoint.start()` is called

Unloading a Feature

- **Kernel.unload(Feature)**

- » Creation of a new thread owned by the Feature: execution of stop()
 - There is a timeout to execute this method
- » Once done, if there are still Feature's threads that are running, a DeadFeatureException is thrown in such threads.

- **On completion with true**

- » all threads owned by the Feature are stopped
- » its code has been unlinked from Kernel
- » memory has been reclaimed (objects and code)

- **On completion with false**

- » all threads are stopped
- » but the Kernel still « points » to some instances of some Feature classes.
- » The kernel may release these reference, and recall unload.

OSGi over Kernel & Features

- **Feature = Group of Bundles**

- » `FeatureEntryPoint.start() → Bundle[].start()`
- » `FeatureEntryPoint.stop() → Bundle[].stop()`

- **Kernel & Features insulation semantic**

- » `BundleContext` access is limited to Bundle owner
See [OSGi r4] section 4.4.16: *[...] it is intended to be used only by the bundle [...]*
- » A Feature can only see Bundles, Services, Events,... owned by itself or the Kernel

- **Framework Permissions**

- » `AdminPermission`
- » `BundlePermission`
- » `ServicePermission`

Security management

- **Embedded Device Configuration (EDC)**

- » Permission framework based on `java.lang.SecurityManager`

- **Runtime introspection**

- » Full insulation between Features, although communication is feasible using kernel objects (i.e. the kernel controls the inter-feature communication)
- » The identity of the owner is accessible
 - Owner of the current stack frame (the Kernel or a Feature)
 - Owner of the objects
 - Owner of the code (bundles' classes & interfaces)

- **Resources management Strategy**

- » Highly dependent from the application domain
- » Highly dependent from the eco-system the “Kernel owner” wants to create

Size & B.O.M cost

- **Java platform size**
 - » JPF (EDC + B-ON) : less than 50Kbytes
 - » KernelFeature extension: less than 10Kbytes
- **Finishing a lightweight embedded linker**
 - » Dynamic linker less than 20KBytes

Démo

Secured Execution Environment multi-operators

Services Provider



Management Layer

SmartHome APIs
Devices access
Application Store



Execution Platform



MicroEJ® HaaS

Hardware Platform



Livebox Pro v3



BBox Sensation



OTT Box

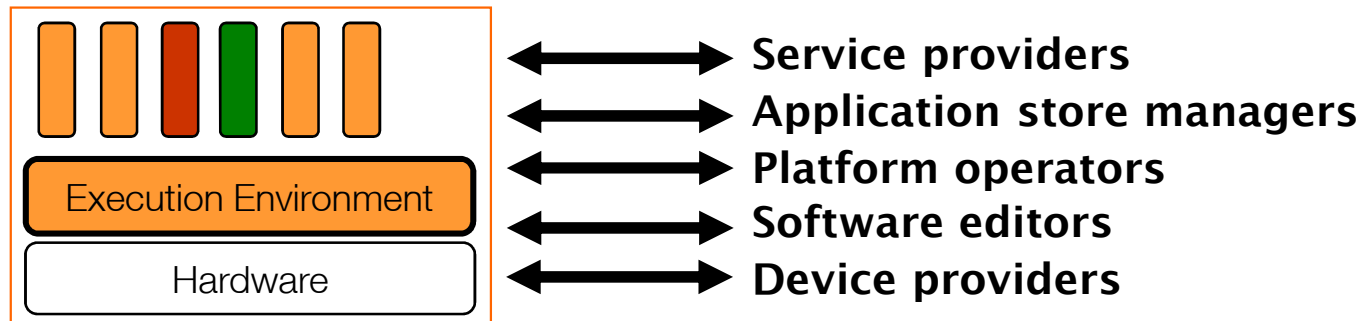
Sensors&Actuators
Network



Open the Box

Project ambition

- Integrate and create a standard execution environment
- to create a dynamic market of applications with Home players



- Publish a model and a set of APIs to the industry



Thanks